NASA Conference Publication 2032

Future Computer Requirements For Computational Aerodynamics

A workshop held at
Ames Research Center
Moffett Field, Calif.
October 4 – 6, 1977

(NASA-CP-2032) FUTURE COMPUTER REQUIREMENTS FOR COMPUTATIONAL AERODYNAMICS (NASA) 515 p HC A22/MF A01 CSCL 09B
N78-19778
THRU
N78-19819
Unclas
G3/59 06597

February 1978

NASA

1. Report No. NASA CP-2032	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle FUTURE COMPUTER REQUIREMENTS FOR COMPUTATIONAL AERODYNAMICS*		5. Report Date
		6. Performing Organization Code
7 Author(s)		8. Performing Organization Report No A-7291
9. Performing Organization Name and Address NASA Ames Research Center Moffett Field, California 94035		10. Work Unit No. 505-06-11
		11 Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		13. Type of Report and Period Covered Conference Proceedings
		14. Sponsoring Agency Code

15 Supplementary Notes

*A workshop held at NASA Ames Research Center, Moffett Field, California, October 4-6, 1977.

16. Abstract

This report is a compilation of papers presented at the NASA Workshop on Future Computer Requirements for Computational Aerodynamics. The Workshop was held in conjunction with preliminary studies for a Numerical Aerodynamic Simulation Facility that will have the capability to solve the equations of fluid dynamics at speeds two to three orders of magnitude faster than presently possible with general purpose computers. Summaries are presented of two contracted efforts to define processor architectures for a facility to be operational in the early 1980's.

17. Key Words (Suggested by Author(s)) Numerical analysis Computer sciences	18. Distribution Statement Unlimited STAR Category – 59
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified

*For sale by the National Technical Information Service, Springfield, Virginia 22161

U.S.GPO:1978-793-973/184

NASA Conference Proceedings 2032

FUTURE COMPUTER REQUIREMENTS

FOR COMPUTATIONAL AERODYNAMICS

A workshop held at NASA Ames Research Center
Moffett Field, Calif. 94035
October 4-6, 1977

PREFACE

The National Aeronautics and Space Administration is conducting preliminary studies of a Numerical Aerodynamic Simulation Facility that will serve as an engineering tool to enhance the Nation's aerodynamic design capability in the 1980's. This facility will provide computer simulations of aerodynamic flows at processing speeds several orders of magnitude faster than possible now with general purpose computers. The Workshop on Future Computer Requirements for Computational Aerodynamics was organized to elicit input from both computational aerodynamicists and computer scientists regarding the computer requirements for obtaining the desired solutions and the projected capabilities of general purpose computers and special purpose processors of the early 1980's.

The Workshop was opened with presentations outlining the motivations for the Numerical Aerodynamic Simulation Facility project and its potential benefits, supported by the recent advances being made in computational aerodynamics (Session 1). Subsequent sessions included invited presentations and panels. The invited presentations were comprised of projections of computing technology and computational aerodynamics in the 1980's (Session 2), results of two contracted efforts sponsored by Ames Research Center to define promising processor architectures for three-dimensional aerodynamic simulations (Session 3), and reports of two studies sponsored by the Air Force Office of Scientific Research (Session 8). The eight panels addressed a number of key issues pertinent to the future advancement of computational aerodynamics, including Compu~~tati~~onal Aerodynamics Requirements (Session 4), Viscous Flow Simulations (Session 5), Turbulence Modeling (Session 6), Grid Generation (Session 7), Computer Architecture and Technology (Session 9), Total System Issues (Session 10), Specialized Fluid Dynamics Computers (Session 11), and Supercomputer Development Experience (Session 12).

The Proceedings have been reproduced from manuscripts submitted by the participants and are intended to document the topics discussed at the Workshop. A list of attendees is appended at the end of this volume.

WORKSHOP COMMITTEE

V. L. Peterson, General Chairman
F. R. Bailey, Technical Program Chairman
M. Inouye, Arrangements Chairman & Proceedings Editor
A. W. Hathaway
W. P. Jones
K. G. Stevens, Jr.

CONTENTS

SESSION 5, Panel on VISCOUS FLOW SIMULATIONS
Robert W. MacCormack, Chairman

SESSION 6, Panel on TURBULENCE MODELING
Joel H. Ferziger, Chairman

SESSION 7, Panel on GRID GENERATION
Joe F. Thompson, Chairman

SESSION 8

SESSION 12, Panel on SUPERCOMPUTER DEVELOPMENT EXPERIENCE
S. Fernbach, Chairman

SESSION 1

F. R. Bailey, Chairman

OPENING REMARKS

Dean R. Chapman
Director of Astronautics
Ames Research Center, NASA

I note from the list of attendees at this workshop that we have representation from a very wide range of institutions--from computer hardware companies, software companies, universities, aircraft companies, the Air Force and other DOD organizations, private research groups, various NASA Centers and other government agencies--all with an interest in large scale scientific computations. In view of such diversity, and of the circumstance that many attendees are more indirectly than directly involved with the development of computational aerodynamics, it is appropriate to devote this introduction to outlining some of the driving motivations behind the development of computational aerodynamics. These motivations have not changed in the past decade, and we do not expect them to change in coming decades.

Two major motivations are (1) that of providing an important new technological capability and (2) economics. To illustrate the first, a comparative listing is made in Figure 1 of the fundamental limitations of wind tunnels and of numerical flow simulations. Every wind tunnel is limited, for example, by the size of model that can be put into it, by the flow velocity it can produce, and by the pressure it can be pumped up to. Thus wind tunnels have rarely been able to simulate the Reynolds number corresponding to the free flight of aircraft. The Wright Brothers, with their small box-size wind tunnel, were aware of the presence of "scale effects" in wind tunnel data, and the Reynolds number limitation of wind tunnels is still a problem today. Limitations on temperature and on the atmosphere that wind tunnels can utilize restrict their ability to provide simulations of earth atmosphere entry flights and of probes entering other planetary atmospheres in the solar system. Of particular importance to transonic aerodynamics are the limitations imposed by the interfering effects of the presence of wind tunnel walls and supports. Near a Mach number of one these severely restrict the accuracy of wind tunnel data. Aeroelastic distortions always present in flight are not simulated in wind tunnels; and the stream nonuniformities of wind tunnels have long been known to severely affect the laminar-turbulent transition data from wind tunnels. All these fundamental limitations have one thing in common; they limit the ability of wind tunnels to simulate free flight conditions.

In contrast, computer numerical flow simulations have none of these fundamental limitations, but have their own: computational speed and memory storage. Even though these latter limitations are fewer in number, they have been overall much more restrictive in the past than have been the limitations of wind tunnels. The reason for this is simply that the basic set of differential equations governing fluid flow, the Navier-Stokes equations, are of extreme mathematical complexity. This has required the theoretical aerodynamicist in the past to use highly truncated and approximate forms of the Navier-Stokes equations in making analyses. Only in the past three years has computer capability reached a stage where it is practical to conduct numerical simulations using the complete Navier-Stokes equations; and these simulations have been restricted to very simple aerodynamic configurations. It is important to note that the fundamental limitations of computational speed and memory are rapidly decreasing with time; whereas the fundamental limitations of wind tunnels are not. In essence, numerical simulations have the potential of mending the many ills of wind tunnel simulations, and providing thereby an important new technological capability for the aerospace industry.

The second major motivation, that of economics, has two essential contributing aspects: computer technology trends and numerical analysis trends. Although the cost of computers has risen with time, their computational power has increased at a much greater rate. Hence the net cost to conduct a given numerical simulation with a fixed algorithm is decreasing rapidly with time. This remarkable and well-known trend, illustrated in Figure 2, is expected to continue for some time. In addition, there has been another important trend that is not as widely known. The rate of improvement in the computational efficiency of numerical algorithms for a given computer has also been remarkable. This is illustrated in Figure 3 where the trends in relative computation cost due to computer improvements alone are compared to the corresponding trend due to algorithm improvements alone. The two trends have compounded to bring about an altogether extraordinary trend in the economics of computational aerodynamics.

An example may suffice to illustrate this. Numerical flow simulations for a two dimensional airfoil using the full time-averaged Navier-Stokes equations can be conducted on today's supercomputers (e.g., Illiac, Star, Cray, ASC Class) in roughly a half hour at roughly $1000 cost in computer time. Examples of such simulations are given in the subsequent presentation of Mr. Victor Peterson. If we had attempted just one such simulation twenty years ago in 1957 on computers of that time (IBM 704 Class) and with algorithms then known, the cost in computation time alone to complete just one such flow simulation would have amounted to roughly $10 million, and the results for that single flow simulation would not be available until 1987, ten years from now, since it would have taken about 30 years to complete.

So, by way of introduction I would like to leave you with the thought that the major driving motivations behind the development of computational aerodynamics are fundamentally sound, and that we certainly do not expect them to fade in importance in years to come.

WIND TUNNEL	NUMERICAL
MODEL SIZE	SPEED
VELOCITY	STORAGE
DENSITY	
TEMPERATURE	
WALL INTERFERENCE	
SUPPORT INTERFERENCE	
AEROELASTIC DISTORTIONS	
ATMOSPHERE	
STREAM UNIFORMITY	

Figure 1.- Comparison of analog and digital flow simulations - fundamental limitations.

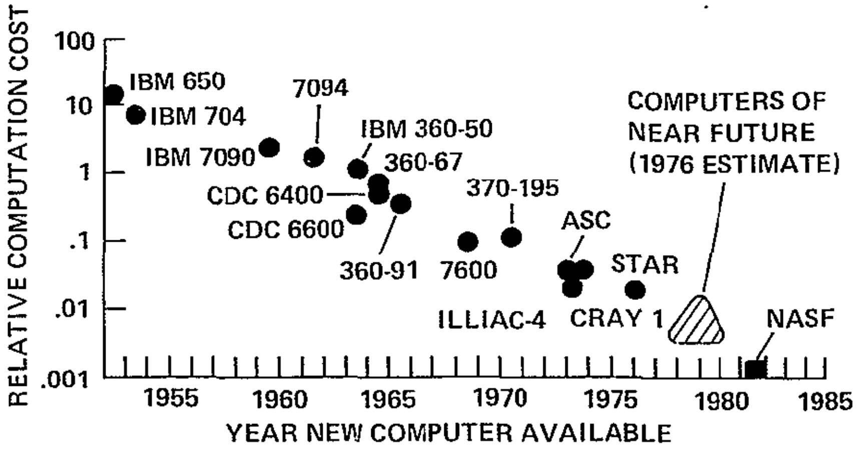


Figure 2.- Computation cost trend of computer simulation for a given flow and algorithm.

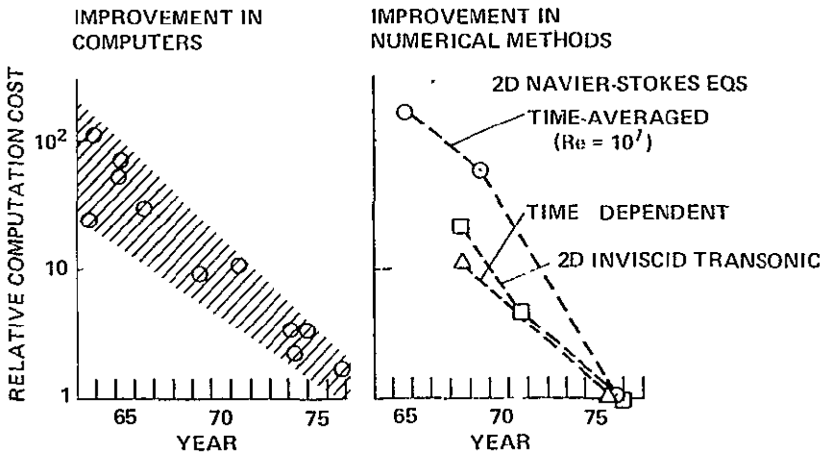


Figure 3.- Improvements in cost for computer simulation of a given flow.

KEYNOTE ADDRESS

COMPUTATIONAL AERODYNAMICS
AND THE
NUMERICAL AERODYNAMIC SIMULATION FACILITY

Victor L. Peterson

Ames Research Center, NASA

INTRODUCTION

The objective of computational aerodynamics is to simulate aerodynamic flow fields through the numerical solution of approximating sets of the fluid dynamic equations using high-speed computers. The discipline is characterized as being a composite of four other disciplines: aerodynamics, fluid physics, mathematics and computer science. Obviously aerodynamics is involved since the goal is to determine the motions of gases and their effects on bodies moving through them. Fluid physics comes into play in the course of modeling turbulent momentum and heat transport terms in the Navier-Stokes equations. Mathematics is drawn upon in the course of developing efficient algorithms for solving the governing equations with numerical methods. Finally, computer science has been involved in the development of new languages and compilers that permit more efficient coding of the equations for solution on computers having various architectures. Computer science now is playing an even more important role in the effort to define a machine that is optimized for solving the fluid-flow equations.

The discipline of computational aerodynamics, even in its early stages of development, is emerging as an important aerodynamic design tool. While it is not yet possible to rely solely upon computation to design a new aerospace vehicle, there are numerous examples of experimentally verified aerodynamic improvements to designs that have evolved from the application of computational methods. One such example will be discussed later.

There are both technical and economic reasons for accelerating the maturation of the discipline of computational aerodynamics. It is well known that the cost of conducting the experiments required to provide the empirical data base for new aeronautical vehicles is increasing rapidly with time. Two factors account for this increase. The cost of performing wind-tunnel and flight experiments is rising rapidly as the cost of labor and energy escalates. More importantly, however, the actual amount of experimentation required is increasing almost exponentially with each new generation of vehicle. It is also well known that the performance of new aerospace vehicles often is compromised by the over-design required because of the limitations in test facilities (Reynolds number, wall and support interferences, aeroelastic distortions, real-gas effects, etc.) for simulating the full-scale vehicle environment. On the other hand, the equations governing fluid flows are well known, the numerical methods for solving them are being continuously improved, and the cost of performing calculations is decreasing with time.

The purpose of this paper is to outline the goals and potential benefits, current status, and future prospects of computational aerodynamics. In addition, the computer requirements for advancing the discipline will be defined and an approach to satisfying these requirements will be presented. The paper will conclude with a discussion of a project that is underway to develop a Numerical Aerodynamic Simulation Facility that will enhance the nation's aerodynamic design capability.

GOALS AND POTENTIAL BENEFITS

The goals of computational aerodynamics and potential benefits to be derived from pursuing them are outlined in figure 1. The first of the goals shown on the left side of the figure is to provide a rapid and inexpensive means for simulating fluid flows. The aim is to develop a tool for use in aircraft design that will complement the wind tunnel by providing some of the needed data more quickly and at less cost. The second goal is to provide a more powerful combination of theory and experiment than is now available. The idea here is to be able to numerically simulate aerodynamic experiments conducted in test facilities and thereby provide the means for improved interpretation and understanding of observed phenomena. The third goal is to make possible an enhanced understanding of the influence of design variables on aircraft performance. This will follow from being able to explore far more combinations of the design variables on the computer than would be practical in the wind tunnel. The fourth goal is to have the means for simulating aerodynamic flows that are unaffected by the usual wind-tunnel constraints such as wall and support interference effects, aeroelastic distortions, and Mach- and Reynolds-number limitations. In addition to providing direct estimates of free-flight aircraft aerodynamics, the computational capability, of course, will permit the determination and elimination of the effects of the wind-tunnel constraints on measured data. The last goal, and perhaps the most important is associated with optimizing aerodynamic configurations. Powerful mathematical theories of optimization can be combined with the aerodynamic codes to permit optimum shapes subject to given constraints to be developed.

The potential benefits to be derived from the numerical simulation capability are listed on the right-hand-side of figure 1. Significantly improved preliminary designs will result from being able to economically search a large design space for the configuration best satisfying the desired mission profile while at the same time simulating the true free-flight situation. Increased efficiency of wind-tunnel testing will be made possible by being able to reduce the number of configurations that must be tested. If the most promising configurations are first identified computationally then a much reduced testing program can be conducted to verify and refine the resulting aerodynamic shapes. Clearly, this will help to make better use of the scarce and costly high Reynolds number test facilities. Together, all of the factors discussed lead to the fact that the application of computational aerodynamics will permit new aerospace vehicles to be designed in shorter periods of time, at less cost and with lowered risk of not meeting performance specifications.

Proof that improved aerodynamic design tools are needed is given in figure 2 wherein examples of inadequate simulation capabilities are presented. Many of the aircraft that have been designed and built were found to have aerodynamic deficiencies only after full-scale flight tests were performed. In some cases the problems were so severe that the loss of a flight-test vehicle occurred. In all cases, the problems that were uncovered in flight tests led to either costly modifications and/or reduced aircraft performance. Thus, there is a strong incentive to develop additional aerodynamic design tools.

One aspect of the complementary nature of computational aerodynamics and ground-based experiment is illustrated by the comparison of conventional and advanced design approaches in figure 3. The design of a vehicle requires the consideration of many interrelated parameters. The concept to be illustrated, however, can best be visualized by focusing on just two; for example, wing leading- and trailing-edge sweep for a given aspect ratio. The conventional design approach shown on the left side of the figure involves an experimental design space selected for investigation in wind-tunnel tests. The dimensions of the design space, or in other words the number of models that can be built and tested, are necessarily limited. Wind tunnel tests will uncover a good combination of the parameters within the design space considered but there is no assurance that a better design does not exist outside of the space investigated. In the advanced design approach shown on the right of the figure a much larger portion of the design space is first searched computationally for the best design. Then a much smaller portion of the space surrounding the computationally determined best design is selected for verification and refinement tests. Of course, this is a trivial example when only two design parameters are involved. In reality, when many parameters are involved, the n-dimensional space can only be adequately searched with the aid of computational methods and numerical optimization techniques. The quality of the results obtained depends only on the quality of the numerical simulation.

STATUS OF COMPUTATIONAL AERODYNAMICS

The discipline of computational aerodynamics originated in the 1950's when electronic digital computers first became available to aeronautical researchers. Only in the last decade, however, has the available computational power been sufficient to permit real advances in the state-of-the-art. The evolution of the discipline and some examples of current capabilities are discussed in this section.

Stages of Approximation to 3-D Numerical Aerodynamic Simulations

The four different stages of approximation applicable to computer simulations of aerodynamic flows are outlined in figure 4. Summarized for each stage are the nature of the approximations made to the governing equations and the class of computer required for practical three-dimensional engineering computations.

Stage I - The Past: Inviscid Linearized Equations. - This highly simplified approximation to the full governing equations has long been used as an aid in aircraft design. The linearized flow over lifting airfoils has been computed within this framework of approximation ever since the 1930's. With the development during the 1960's of computers of the IBM 360 and CDC 6600 class, it became practical to compute linearized inviscid flows about complete aircraft configurations. Considerable effort still is being expended to make the computer codes more efficient and to develop better methods for treating the boundary conditions on complex curved surfaces. Because the equations of motion neglect all viscous terms as well as inviscid nonlinear terms, such flow simulations provide only a minor complement to wind-tunnel simulations in the overall aerodynamic design process.

Stage II - The Present: Inviscid Nonlinear Equations. - Although neglecting only viscous terms in the Navier-Stokes equations, this approximation still imposes severe limitations on the usefulness of computed flow simulations. It has been less than a decade since the first useful solutions of these equations were obtained numerically. Despite the fact that great advances in numerical methods have been made since then, efficient routine computations still require a computer of the CDC 7600 class or better. Only problems in which viscous effects are not dominant can be treated adequately with these equations. Nevertheless, the ability to solve them has provided the designer with a new and valuable tool. In particular, results now can be obtained in some flow regimes where previously available theoretical methods were inadequate and where wind-tunnels have fundamental shortcomings. One example involves the hypersonic chemically reacting flow about the Space Shuttle Orbiter. The computer is providing results impossible to obtain in any known ground-based facility. Another example involves transonic flows where wind-tunnel data often are plagued by uncertainties due to wall interference.

Stage III - The Near Future: Viscous Reynolds Averaged Navier-Stokes Equations. - This approximation neglects no terms in the full, Reynolds averaged Navier-Stokes equations. Certain terms involving the turbulent momentum, energy and heat transport terms are modeled, however. The accuracy of the turbulence model limits this approximation and the development of improved models for separated as well as attached flows chiefly paces this type of flow simulation. Relatively large amounts of computer time are required using the stage III approximation. While two-dimensional flows take less than an hour on a CDC 7600 with current numerical methods, the routine computation of three-dimensional flows is not practical since they require roughly 100 times the computation of two-dimensional flows. With the improved turbulence models expected to be developed in the coming years, advanced computational capability at least 40 times that of current supercomputers should permit computations to provide a major complement to wind-tunnel simulations in the design of new aerospace vehicles.

Stage IV - The Far Future: Viscous Time Dependent Navier-Stokes Equations .- The final stage involves solving the complete time-dependent Navier-Stokes equations of viscous fluid motion. In essence, all of the turbulent eddies of significant size would be computed for a sufficiently long time period to yield both the time-averaged characteristics of the flow as well as its unsteady components. The significant-size turbulent eddies that transport the principal momentum and energy are relatively large, the order of 10 or more boundary-layer thicknesses in length. The subgrid-scale turbulent motion, of course, would be modeled in order to minimize the required computer time by permitting the use of the largest practical grid spacing. The grid spacing would be small enough, however, so that the end result would be insensitive to the particular subgrid-scale model employed. Under such conditions the computed results would involve essentially no empiricism. This fourth stage requires several orders of magnitude more computation than the third stage. Consequently, development of an advanced computer clearly is required for providing such simulations on a research basis for practical aerodynamic configurations.

Current 3-D Inviscid Capability

Important aircraft design problems associated with transonic flow now can be solved with currently available methods for processing the nonlinear three-dimensional inviscid equations. Of course, the particular problems chosen for solution cannot be dominated by separated flows since viscous effects are neglected.

Results of a recent application of computational aerodynamics to a practical design problem are shown in figure 5. The original design of the Highly Maneuverable Aircraft Technology (HiMAT) Remotely Piloted Research Vehicle (RPRV) was found through wind-tunnel tests to have excessive drag at the design conditions. The original design was based on linear inviscid theory since the newer codes including nonlinear terms that are important at transonic speeds were not available to the designers at that time. The failure of the original design to meet the performance goal was due to strong shock waves, that are not predictable with linear methods, forming on the upper surface of the wing at transonic speeds. Fiscal and time constraints precluded any chance for correcting the deficiencies by conducting an extensive wind-tunnel test program. Therefore, a decision was made to redesign the wing computationally using a new transonic code. The objective was to reshape the wing to obtain an improved surface pressure distribution. The goal was to decrease the drag by decreasing the strength of the shock wave on the upper wing surface while maintaining the same lift by increasing the loading on the forward portion of the wing. This was accomplished by making small changes to the airfoil shape and wing trailing-edge sweep.

Tests of a model of the computationally improved configuration produced results that came very close to meeting the design goal. About 10 iterations of the wing shape were required to evolve the new design. These required the use of about $6K of computer time compared to the estimated cost of $150K to obtain the same results experimentally. In addition, the redesign was accomplished in considerably less calendar time through the use of computational tools.

New 2-D Viscous Capability

The HiMAT example shows that computational methods, even in their relatively primitive state, can have a large impact on aircraft design. It is important, however, to recognize that currently available methods still have their limitations. They can be expected to work well only when applied to problems that can be treated within the framework of the approximations involved. The usefulness of the computational approach will be considerably enhanced when the computer power is available to routinely simulate flows with boundary-layer separation. This capability now is emerging for problems characterized by two-dimensional flows.

One example of a two-dimensional problem involving flow separation that has been investigated computationally using the Reynolds averaged Navier-Stokes equations is shown in figure 6. The problem illustrated in the upper left of the figure deals with the aerodynamic flow over the aft end of an idealized aircraft shape. A turbulent supersonic flow approaches a boattailed afterbody followed by a solid body representing an engine exhaust plume. The extent of separated flow was varied by changing the boattail angle over the range from 16- to 40-degrees. Computed and measured surface pressure coefficients are shown in the lower left of the figure for the relatively steep boattail angle of 34°. The agreement between the computations and measurements is remarkably good even though there is an extensive region of separated flow. Drag coefficients for a range of boattail angles are shown in the upper right of the figure. Again, the agreement between theory and experiment is excellent. Wind-tunnel results were obtained at only one Reynolds number. The computations, however, could easily be performed for a range of Reynolds numbers and results are shown in the lower right of the figure. The wind tunnel was limited to a unit Reynolds number of 14×16^6 per meter while a value representative of full-scale flight is about an order of magnitude higher. The excellent results obtained from the computations at the lower Reynolds number wind-tunnel conditions provide confidence in the computed results at flight conditions.

Another example involving the solution of the two-dimensional Reynolds averaged Navier-Stokes equations is shown in figure 7. The transonic performance of a supercritical airfoil has been computed and compared to wind-tunnel measurements. Here the computed results should not be expected to agree exactly with the measured results since the computations are representative of the free-flight situation while the measurements are influenced by wind-tunnel wall interference effects. Note that the onset of buffet near maximum lift coefficient and the increasing magnitude of unsteady forces with further increase in angle of attack is predicted by the computations. The measured buffet domain could not be determined in the experiment since it was not designed to acquire dynamic measurements. It is quite likely that free-flight characteristics of transonic airfoils now can be calculated as accurately as they can be measured in wind tunnels although further investigations are required for confirmation of this assertion.

A final example of two-dimensional viscous flow simulation concerns three types of separation of the flow about a thick circular-arc airfoil. The problem is illustrated in figure 8. At lower Mach numbers the flow about this airfoil is steady and separates from the trailing edge as shown in the sketch on the left of the figure. At higher transonic Mach numbers the shock wave appearing on the airfoil becomes strong enough to cause the steady flow to separate just downstream of the shock. This situation is illustrated by the sketch on the right of the figure. At intermediate Mach numbers the flow is violently unsteady and is characterized by asymmetric separation. At one instant of time the flow separates from the shock on the upper surface and from the trailing edge on the lower surface. At the next instant of time the pattern reverses as shown by the sketch in the center of the figure. Calculated and measured pressure coefficients for these three types of flow are shown in the lower part of the figure. The disagreement between measurements and calculations in regions of massively separated flow is due to inadequacies in the turbulence model; these results were obtained using a simple algebraic eddy viscosity model. More important, however, is the fact that the calculations correctly capture the general features of the flows in the three regimes including the approximate magnitude of the unsteady pressures at the intermediate Mach number.

Calculated contours of constant Mach number are shown in figure 9 at four different instants of time for the freestream Mach number where alternating flow separation was obtained. These results clearly show the unsteady nature of the flow and the asymmetric features of the shock waves and wake. Shadowgraph movies taken during the experiment bear a striking resemblance to these flow patterns.

Further results are presented in figure 10 wherein measured and calculated surface pressure time histories are shown for the circular-arc airfoil example. Both the frequency and wave forms of the unsteady pressures are reproduced reasonably well by the calculations.

FUTURE PROSPECTS FOR COMPUTATIONAL AERODYNAMICS

The next step in the development of the discipline of computational aerodynamics is to extend the viscous-flow simulation capability to three dimensions. This will provide the means for calculating the aerodynamic characteristics of complete aircraft configurations throughout the envelope of possible flight conditions. Pioneering efforts now are underway to build the three-dimensional codes and a few relatively simple problems already have been solved. There are no conceptual difficulties in taking this step.

Improvements in the utility of computational aerodynamics depend, in part, on reducing the cost of calculating complex flow fields and improving the accuracy of the results obtained. The cost is influenced by two factors: cost effectiveness of the computers themselves, and the efficiency of numerical methods. Accuracy is primarily dependent on the ability to properly model the turbulent momentum and heat transport terms in the Navier-Stokes equations. Based on past experience, future improvements can be expected in computer cost effectiveness, efficiency of numerical methods and accuracy through improved turbulence models. This past experience is discussed in the following paragraphs.

Computer Cost Effectiveness

The trend of computation cost for computer simulation of a given flow is shown in figure 11. These data were obtained by determining the cost of running a given code with a given algorithm on machines ranging from the IBM 650 to the current generation of supercomputers and then normalizing the results to the cost of performing the computations on an IBM 360-50. For over 20 years the relative computation cost has been decreasing. In fact, there has been a three order-of-magnitude reduction in cost over the span of two decades. Estimates for the next generation of general-purpose computers show that the trend toward reduced costs will continue. Of course, the reason for this well-established trend is that each new computer is much faster than its predecessor while being only a little more expensive to own and operate.

Efficiency of Numerical Methods

Dramatic improvements in the efficiency of numerical methods have been made in the last 10 years. This is illustrated in figure 12 wherein reductions in computation cost due to improvements in numerical methods are shown. Data on computer cost effectiveness from the previous figure also are shown for comparison. It is striking to note that improvements in numerical methods have kept pace with improvements in computers.

The efficiency of a numerical method for solving the governing flow equations depends on the number of mathematical operations required to obtain a solution. Early methods required thousands of iterations to obtain a converged solution for a high Reynolds number problem. New methods are being found to drastically reduce the number of iterations required as well as to reduce the number of operations per iteration. There still is much room for improvement. Intensive effort is being expended to achieve at least another factor of four in efficiency within the next five years. The prospects of meeting this goal are very good.

Improvements in Turbulence Models

The Navier-Stokes equations embody a complete description of turbulent as well as laminar flow. In the case of turbulent flow, however, the wide range of significant scales of fluid motion makes it impractical to solve the complete equations in the foreseeable future for all but the simplest flows. Thus, it will be necessary to continue the reliance on turbulence models for some time to come.

Fluid physicists have been trying to develop improved turbulence models for many years and the progress has been slow. The models depend on constants that must be evaluated experimentally. It is relatively easy to find a set of constants that will apply to one type of fluid flow but it is difficult to develop a single model that will apply universally to all types of flows.

The prospects of developing improved turbulence models in the future are much brighter than they were in the past. Two factors account for this optimism. First, advances are being made in the development of fluid-flow diagnostic equipment. A good example is the laser velocimeter which provides the means for unobtrusively measuring individual components of the mean and fluctuating velocities of small elements of a moving fluid. Other devices are being developed to measure instantaneous values of other quantities such as density and temperature, also without inserting large probes in a flow. The information obtained with these tools will provide a more complete description of the physics of turbulence than has been available in the past. Secondly, computers and numerical methods are now efficient enough to permit the governing equations to be routinely solved at least for two-dimensional flows. This will allow many more models to be tested over wider ranges of flows. In addition, turbulence now can be investigated computationally for a few simple flows by solving the time-dependent Navier-Stokes equations using subgrid scale models that employ fewer approximations. These solutions will provide an even more complete set of benchmark data for developing and testing models. Working hand-in-hand, with the new tools now available, the computational fluid dynamicist and the fluid physicist will be able to advance the understanding of turbulence at a much faster rate than heretofore possible.

There are numerous examples of recent advances in turbulence modeling. One case shown in figure 13 is for a high Reynolds number supersonic turbulent flow over a compression corner. Measured pressures and skin-friction coefficients are compared with computed results based on several different turbulence models. The calculated pressures are relatively insensitive to the choice of turbulence model and all models work equally well. The same is true for the skin friction upstream of the region of separated flow. Downstream of flow reattachment, however, the simpler algebraic eddy viscosity models do not give an accurate description of the skin friction. A newer model which uses a differential equation to describe the turbulent kinetic energy provides markedly improved results.

Additional evidence that advances in turbulence modeling are being made is presented in figure 14. This problem involves the interaction of a normal shock wave with a turbulent boundary layer. The situation is similar to that for the corner flow problem in that all models adequately describe the pressures throughout the flow and the skin friction upstream of the shock boundary-layer-interaction. Once again improved results have been obtained within the past year with a one-equation model. These results are particularly significant since data are available to show that the same model gives equally good results over a very large range of Reynolds number.

The overwhelming majority of past efforts in turbulence modeling have been focused on the understanding of two-dimensional flows. It is now time to place more attention on three-dimensional problems. Benchmark experiments must be defined and conducted to provide the necessary data base. If the effort is started now, models adequate for many applications should be available by the early 1980's.

FUTURE COMPUTER REQUIREMENTS

Available computational power is limiting the advancement of the discipline of computational aerodynamics. Much faster machines with considerably larger memories are required to solve the three dimensional Reynolds averaged Navier-Stokes equations in a short enough time to be practical for routine use in aircraft design.

A perspective on the amount of computational power that is required can be obtained by reviewing the historical use of computers for computational aerodynamics. Some pertinent data are given in figure 15. Experience has shown that computational methods are not routinely used in the aerodynamic development of an aircraft unless the time required to obtain a simulation for a given set of conditions is of the order of 10 minutes or less. Short computation times are required to make it practical to sort through many possible configurations early in the design cycle when aerodynamic factors can have the largest impact on the shape of a new aircraft. Simpler forms of the aerodynamic equations such as those for 2-D inviscid nonlinear flows or those for 3-D inviscid linearized flows can be solved in 10 minutes or less on machines of the IBM 360-65 or CDC 6600 class. The industry uses these forms of the equations extensively since access to computers of this class is readily available. When machines of this class were made available to the research community they were used to pioneer solution methods for the more sophisticated 3-D inviscid nonlinear equations. Then, as more powerful machines of the CDC 7600 class became available it became possible for industry designers to routinely use the 3-D inviscid nonlinear methods while the researchers moved on to develop methods for solving the next higher level of approximation to the governing flow equations. The current supercomputers are adequate to routinely solve the 2-D Reynolds averaged Navier-Stokes equations and to research extensions to three dimensions but they fall far short of making the 3-D viscous simulations practical for design work. A machine at least 40 times more powerful than the ILLIAC IV is required to take the next major step in computational aerodynamics.

The relationship between the time required to compute the flow about a wing-body combination using the Reynolds averaged Navier-Stokes equations and the speed of a computer is shown in figure 16. These results show that a computer must perform at least one billion floating point operations per second in order to simulate a flow in 10 minutes. This is the minimum required speed and it is arrived at by assuming that future numerical methods will have 4 times the efficiency of those available today. It is interesting to note that the solution of the same problem would take about a month on an IBM 360, a day on a CDC 7600 and many hours on a current supercomputer. These existing machines clearly are not adequate for the task at hand.

In addition to the speed of performing arithmetic operations, the other aspect of computational power that must be considered is memory or working storage. The memory requirement is developed with the data in figure 17 for both 2- and 3-dimensional problems. There are about 31 variables associated with each grid point in the 3-dimensional case since some of the quantities must be carried for two time steps. This number could be somewhat larger if complex turbulence models having more than two variables are required. It is estimated that a minimum of 10^6 grid points are needed to resolve a 3-dimensional flow field. This number is more than ample for optimizing aircraft components but might not be enough to resolve the flow about complete aircraft having complex shapes. Of course, problems requiring more grid points still can be solved but the time needed for solution will be greater than 10 minutes. Multiplying the number of variables per grid point by the number of grid points gives the amount of memory required. This amounts to slightly over 30 x 10^6 words for the 3-dimensional problems. This number is almost 300 times larger than the amount currently being used to solve two-dimensional problems with the Reynolds averaged Navier-Stokes equations.

General purpose computers are not likely to have the speed required for the next stage of development of computational aerodynamics for many years to come, if ever. The effective speed of general purpose computers has been increasing with time along the curve shown in figure 18. Although not shown, forecasts indicate that next generation machines for the 1980 time frame will continue to follow the trend indicated by the dashed line. The minimum requirement for computational aerodynamics is shown to be well above these forecasts.

There are several reasons for the leveling off of the growth in effective speed of general purpose computers. One of these is shown by the graph on the left side of figure 19 wherein data are presented for the speed of logic circuits. These data have been normalized to the state of the technology in 1965. It is seen that some increase in circuit speed is still possible but advances are difficult because of the close approach to the theoretical limit based on speed of light considerations. Economics also is a factor accounting for the growth trend of general purpose computers. The demand for more powerful scientific computers is dwarfed by the demand for more versatile business machines and electronic components for mass produced items such as hand-held calculators and wrist watches.

Even though general purpose computers are not likely to satisfy the computational aerodynamics requirement in the foreseeable future, it appears to be technically feasible to construct a special-purpose processor having the necessary capability. Micro-miniaturization is proceeding at a rapid rate as shown by the circuit-density data on the right side of figure 19. This means that enhanced computer capability can be obtained by matching machine architecture to the problem to be solved. That is, some degree of flexibility for solving all types of problems can be sacrificed for increased performance in working specific problems. Of course, an economic incentive must be provided to encourage the development of a system suitable for somewhat limited but important applications.

NUMERICAL AERODYNAMIC SIMULATION FACILITY PROJECT

A program to define a Numerical Aerodynamic Simulation Facility that will meet the needs of computational aerodynamics has been initiated by the Ames Research Center. The goal is to achieve at least a factor of 40 performance gain over current supercomputers in order to provide a new tool for simulating three-dimensional viscous flows about aerospace vehicles. Concept definition studies were carried out by two contractors in fiscal year 1977. Results of these studies are summarized elsewhere in these proceedings. One of the purposes of this Workshop is to provide a timely release of these concept definition study results.

Several design criteria have been adopted for the facility. The central processor must have a minimum effective speed of one billion floating-point operations per second when operating on the three-dimensional Reynolds averaged Navier-Stokes equations. Its working memory must accommodate at least 31 million words. The entire facility must be reliable and maintainable. Reliability applies both to the mean time to failure and to the capability to detect systematic errors when they occur. The development risk should be low since the goal of this project is not to develop new electronic technologies but rather to assemble existing technologies into a specialized architecture. The machine should be user oriented and easy to program. Finally, the performance of the facility should be comparable to the best general-purpose computers when used for processing the equations of other scientific disciplines.

A schematic diagram of some of the features of the simulation facility is shown in figure 20. The heart of the facility is the flow simulation processor capable of performing at least one billion floating point operations per second and containing a memory of over 30 million words. The processor, or Navier-Stokes solver is supported by a computer that assists with setting up the geometry of the problem to be solved and reducing the computed flowfield data to a usable form. The computer also controls the flow of data in and out of the facility. Users can actively interact with the facility from remote terminals through the user interface. Archival storage devices would be provided for information having long-term value along with graphics devices for displaying data to on-site users.

The estimated productivity of the numerical facility in terms of the number of data sets produced in a year is shown in figure 21. A data set corresponds to a complete description of the aerodynamic flow about one configuration at one set of flight conditions. Based on a conservative estimate of 6000 useful operating hours per year, the numerical facility will produce about 36,000 data sets per year. This compares favorably with the number of data points generated by a major NASA wind tunnel. Of course, a data set produced by the numerical facility contains far more information than a data set obtained in a wind-tunnel test. The numerically produced data set completely describes an aerodynamic flow while an experimentally determined data set normally is comprised of integrated forces and moments and/or a limited number of surface pressure or heat-transfer measurements.

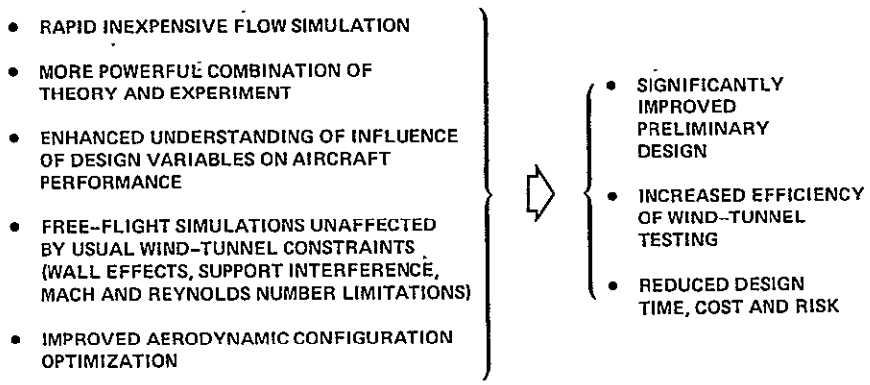
The estimated operational cost of the numerical facility in terms of dollars per data set is shown in figure 22. The cost defined in this manner is comparable to that for a major NASA wind tunnel which is less than one hundred dollars per data set. Again, the comparison with wind tunnels might not be very meaningful because of the vast differences in the information content of the respective data sets but it does provide some perspective. Finally, it should be noted that these data do not include the cost of designing and constructing models for the wind-tunnel tests.

Recent and near-term activities related to the Numerical Aerodynamic Simulation Facility project are summarized in figure 23. In addition to the previously mentioned concept definition studies, a number of briefings have been presented to the major aerospace companies and appropriate advisory committees. The purpose of these briefings was to inform the industry of current plans and to solicit additional views on all aspects of the project. A sample of the response to these briefings is given in figure 24. There was a strong general endorsement of the project along with some frequently asked questions concerning policy for allocating facility time, proprietary security of data and methods for numerically describing the geometry of complex three-dimensional shapes.

The feasibility of achieving project goals was confirmed in the concept definition studies. Therefore, two parallel preliminary design efforts will be initiated this fiscal year. The focus will be on developing a functional design of the flow simulation processor including a simulation of its performance. In addition, specifications will be prepared for all components of the facility. Contingent upon management approvals, final design could begin in fiscal year 1979 leading to operational check-out of the facility in the 1982-83 time frame.

SUMMARY

Computational aerodynamics is an emerging design tool. Even though the discipline is in its early stages of development it has already proven to be very useful and cost effective in the aerospace vehicle design process. Advances in the technologies upon which computational aerodynamics is based are occurring at a rapid rate. Computers are becoming more cost effective, more efficient numerical methods for solving the equations of fluid flow are being found and improved turbulence models are being defined. General purpose computers do not have the necessary capability for the next stage in the development of the discipline. Solution of the three-dimensional Reynolds averaged Navier-Stokes equations in a short enough time to be practical for design purposes will require about 40 times the power of current supercomputers. Even next generation general purpose machines will fall far short of having this capability. Results of feasibility studies show that it is possible, however, to assemble a special-purpose processor that will meet the requirements. Therefore, a project has been undertaken to develop a special-purpose Numerical Aerodynamic Simulation Facility to enhance the nation's aerodynamic design capability in the 1980's.



- RAPID INEXPENSIVE FLOW SIMULATION
- MORE POWERFUL COMBINATION OF THEORY AND EXPERIMENT
- ENHANCED UNDERSTANDING OF INFLUENCE OF DESIGN VARIABLES ON AIRCRAFT PERFORMANCE
- FREE-FLIGHT SIMULATIONS UNAFFECTED BY USUAL WIND-TUNNEL CONSTRAINTS (WALL EFFECTS, SUPPORT INTERFERENCE, MACH AND REYNOLDS NUMBER LIMITATIONS)
- IMPROVED AERODYNAMIC CONFIGURATION OPTIMIZATION

- SIGNIFICANTLY IMPROVED PRELIMINARY DESIGN
- INCREASED EFFICIENCY OF WIND-TUNNEL TESTING
- REDUCED DESIGN TIME, COST AND RISK

Figure 1.- Goals of computational aerodynamics and potential benefits to be derived from pursuing them.

AIRCRAFT	PROBLEMS DISCOVERED IN FLIGHT TEST	CONSEQUENCE
C-141	INCORRECTLY PREDICTED WING FLOW	COMPROMISED PERFORMANCE, COSTLY MODIFICATIONS
C-5A	INCORRECTLY PREDICTED DRAG-RISE MACH NUMBER	REDUCED WING FATIGUE LIFE
F-111	INCORRECTLY PREDICTED TRANSONIC AIRFRAME DRAG	COSTLY MODIFICATIONS
B-58 B-70 YF-12	INCORRECTLY PREDICTED TRANSONIC PERFORMANCE	REDUCED AIRCRAFT EFFECTIVENESS
F-102 F-106	INCORRECTLY PREDICTED TRANSONIC DRAG	REDUCED PERFORMANCE
2 CIVIL TRANSPORTS	INCORRECTLY PREDICTED NACELLE WING INTERFERENCE	REDESIGN REQUIRED

Figure 2.- Examples of inadequate aerodynamic simulation capability.

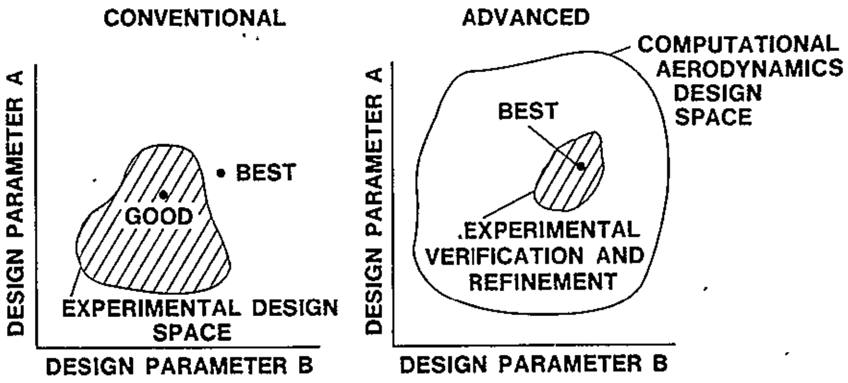

CONVENTIONAL
DESIGN PARAMETER A
BEST
GOOD
EXPERIMENTAL DESIGN SPACE
DESIGN PARAMETER B
ADVANCED
COMPUTATIONAL AERODYNAMICS DESIGN SPACE
DESIGN PARAMETER A
BEST
EXPERIMENTAL VERIFICATION AND REFINEMENT
DESIGN PARAMETER B

Figure 3.- Conventional and advanced aerodynamic design approaches.

STAGE		APPROXIMATION	COMPUTER CLASS FOR PRACTICAL 3D ENGINEERING COMPUTATIONS
I PAST	INVISCID LINEARIZED (1960's)	VISCOUS AND NONLINEAR INVISCID TERMS NEGLECTED	IBM 360 CDC 6600
II PRESENT	INVISCID NONLINEAR (1977)	VISCOUS TERMS NEGLECTED	CDC 7600 STAR CRAY ILLIAC IV
III NEXT STEP	REYNOLDS TIME-AVERAGED NAVIER-STOKES (EARLY 1980's)	NO TERMS NEGLECTED TURBULENT MOMENTUM AND HEAT TRANSPORT TERMS MODELED	AT LEAST 40 TIMES CURRENT SUPERCOMPUTERS
IV FAR FUTURE	FULL TIME-DEPENDENT NAVIER-STOKES (CIRCA 1990)	SUB-GRID SCALE TURBULENCE MODELED	AT LEAST 100 TIMES REQUIREMENT FOR STAGE III

Figure 4.- Stages of approximation to three-dimensional numerical aerodynamic simulations.

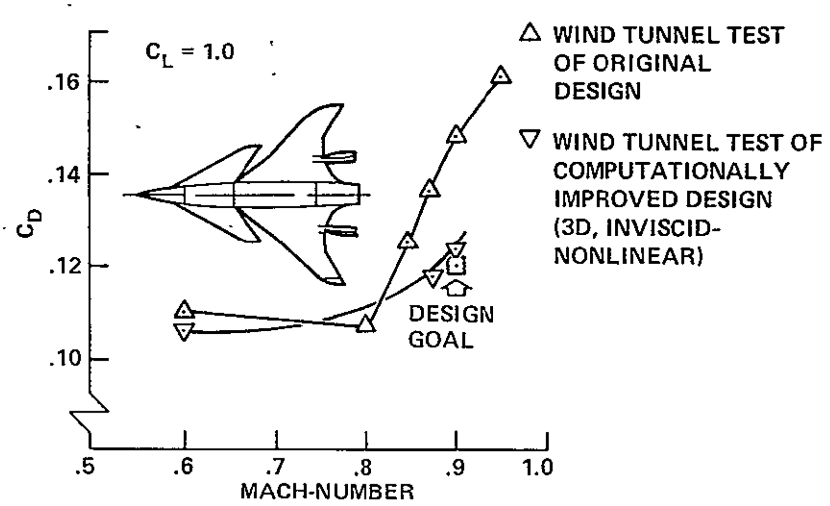


Figure 5.- Results of improving the design of the Highly Maneuver-Aircraft Technology (HiMat) Remotely Piloted Research Vehicle (RPRV) by the application of advanced computational methods.

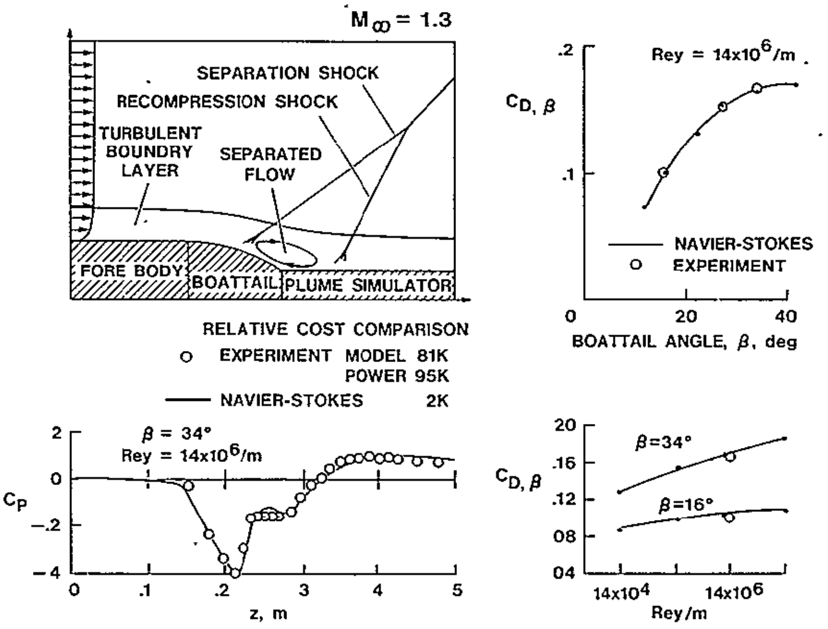


Figure 6.- Comparison of computed and measured pressures and drag coefficients for a boattailed afterbody.

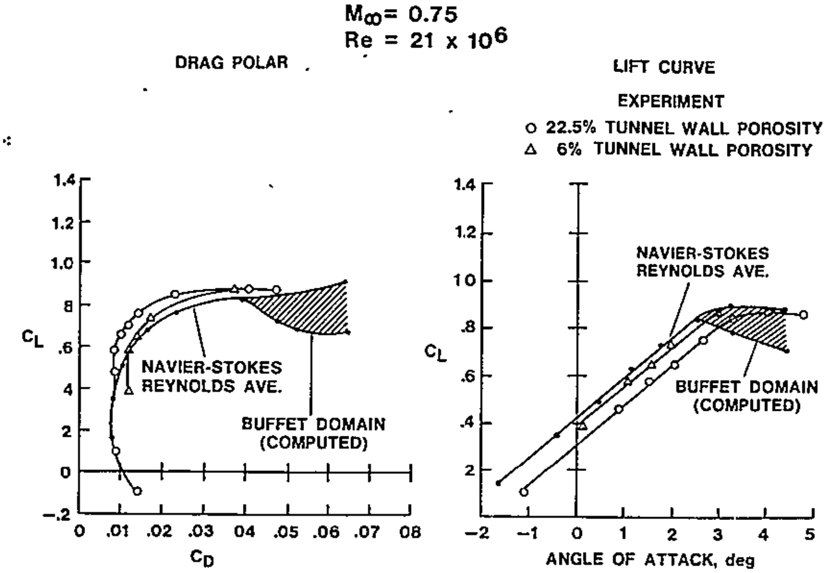


Figure 7.- Comparison of computed and measured drag and lift data for a supercritical airfoil.

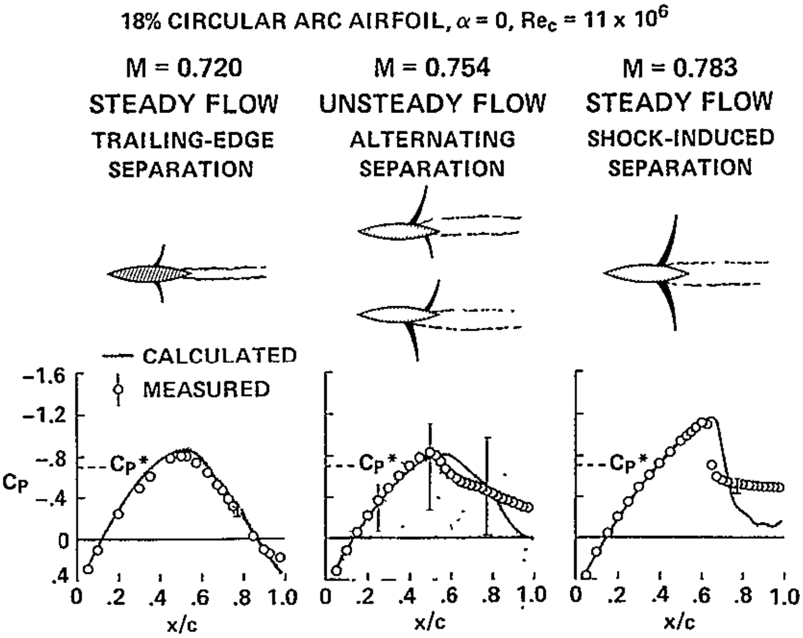


Figure 8.- Computed and measured results for three types of separation of the flow about a thick circular-arc airfoil.

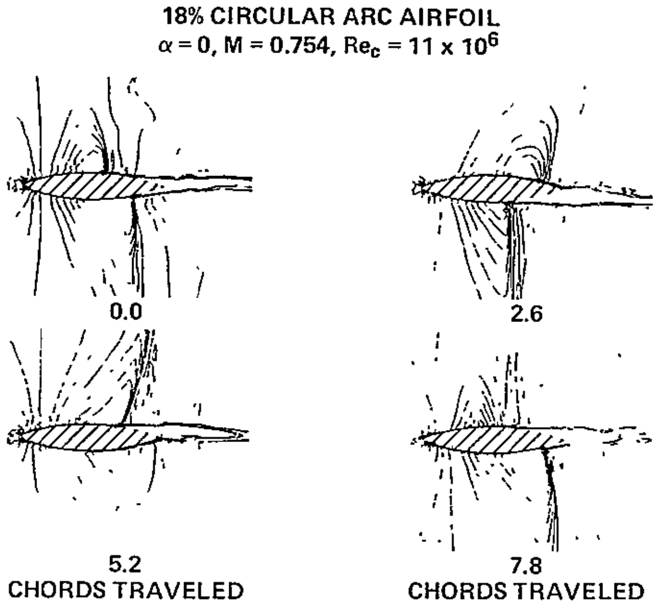


Figure 9.- Computed contours of constant Mach number in the unsteady separated flow about a thick circular-arc airfoil.

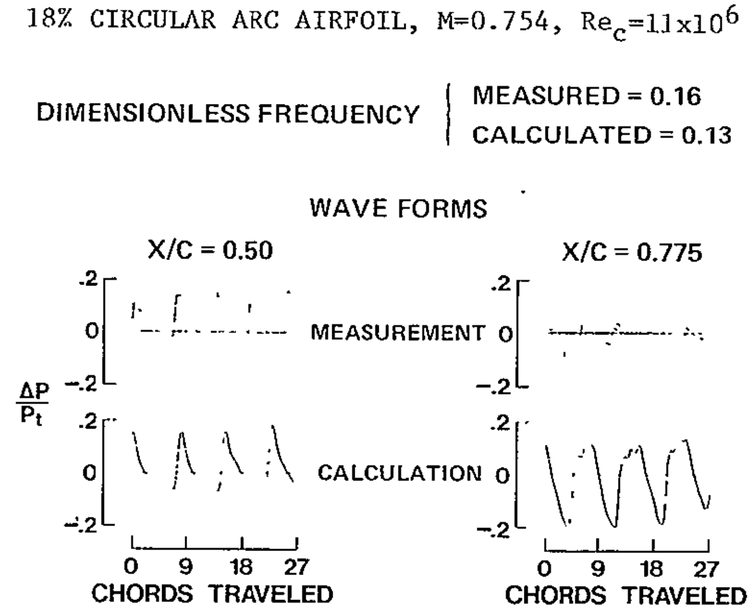


Figure 10.- Computed and measured time histories of surface pressure for a thick circular-arc airfoil experiencing unsteady flow separation.

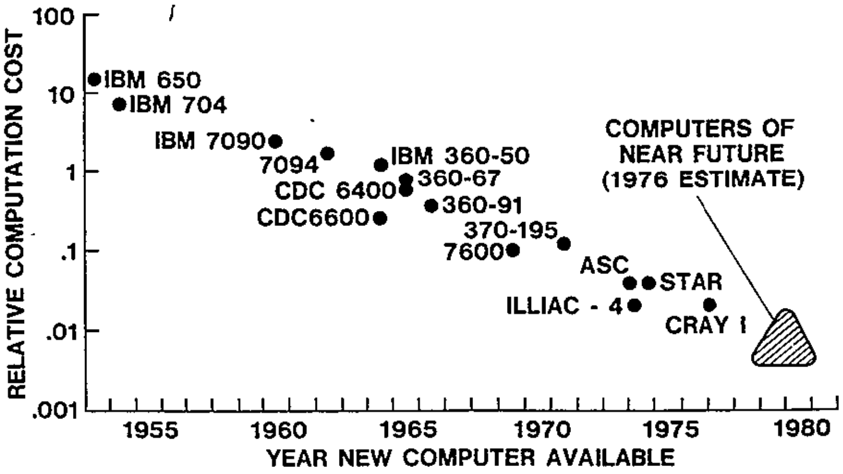


Figure 11.- Trend of computation cost for computer simulation of a given flow.

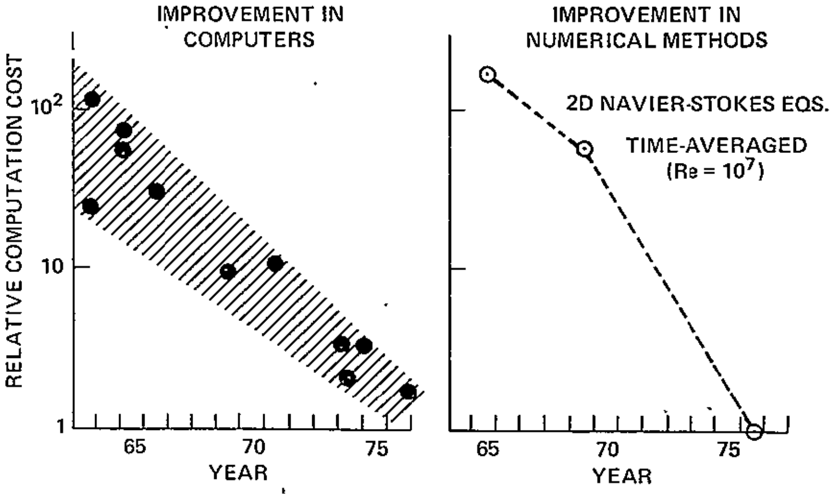


Figure 12.- Improvements in cost for computer simulation of a given flow.

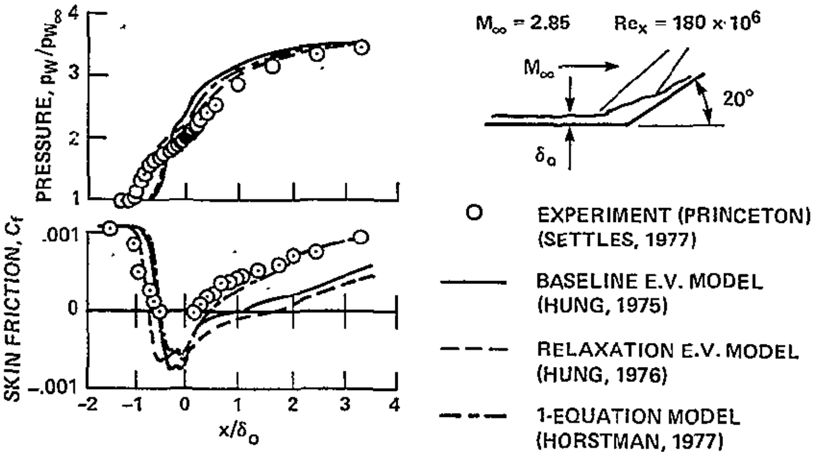


Figure 13.- Improvements in turbulence modeling for the separated turbulent flow over a compression corner.

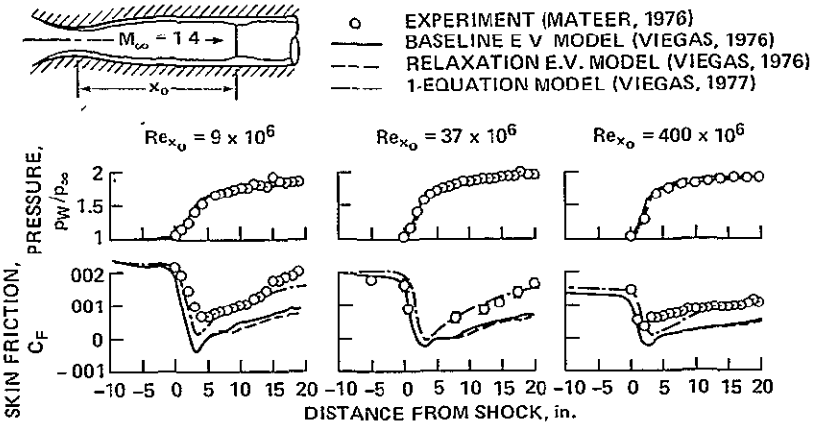


Figure 14.- Improvements in turbulence modeling for a flow having a normal shock wave interacting with a turbulent boundary layer.

COMPUTER CLASS	PRACTICAL ENGINEERING COMPUTATIONS (~10 min CPU TIME)	RESEARCH COMPUTATION (CODE DEVELOPMENT) (HRS CPU TIME)
IBM 360/65 CDC 6600	2D INVISCID NONLINEAR 3D INVISCID LINEARIZED	3D INVISCID NONLINEAR
CDC 7600	3D INVISCID NONLINEAR	2D REYNOLDS AVERAGED NAVIER-STOKES
STAR ILLIAC IV CRAY 1	2D REYNOLDS AVERAGED NAVIER-STOKES	3D REYNOLDS AVERAGED NAVIER-STOKES
40X ILLIAC IV	3D REYNOLDS AVERAGED NAVIER-STOKES	LARGE EDDY SIMULATION

Figure 15.- Computer requirements for computational aerodynamics.

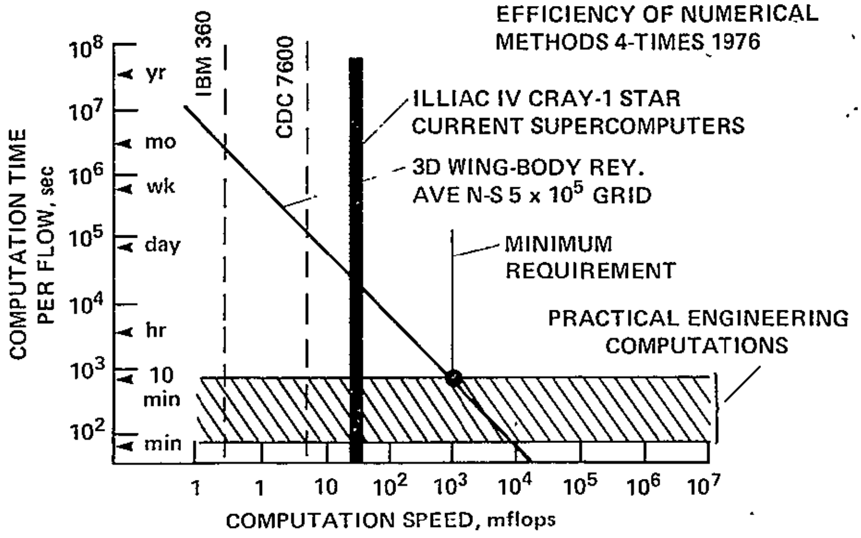


Figure 16.- Relationship between the time required to solve the three-dimensional Reynolds averaged Navier-Stokes equations and the speed of performing arithmetic operations.

	2-D	3-D
VARIABLES	NUMBER OF VARIABLES/ GRID POINT	
FLOW QUANTITIES	8	10
TURBULENCE MODEL	4	4
GRID METRICS	7	13
TEMPORARIES	4	4
TOTAL	23	31
GRID POINTS	5,000	1,000,000
TOTAL STORAGE	115,000	31,000,000

Figure 17.- Computer memory size required for processing the three-dimensional Reynolds averaged Navier-Stokes equations.

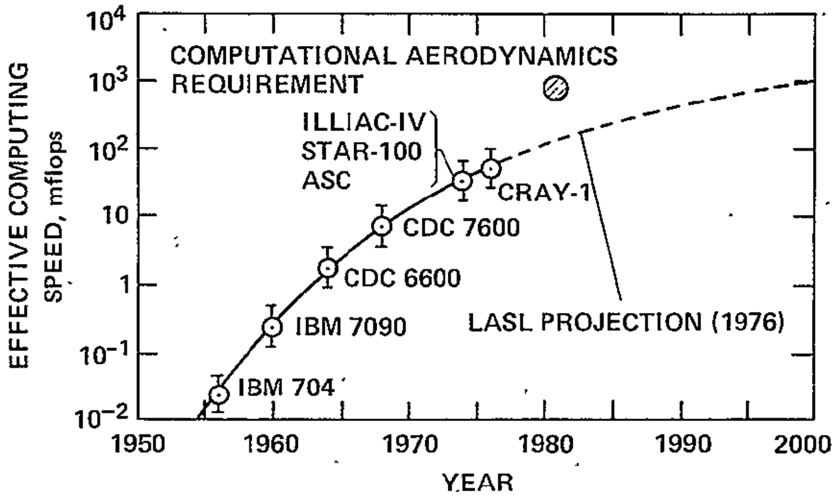


Figure 18.- Trend of effective speed of general-purpose computers.

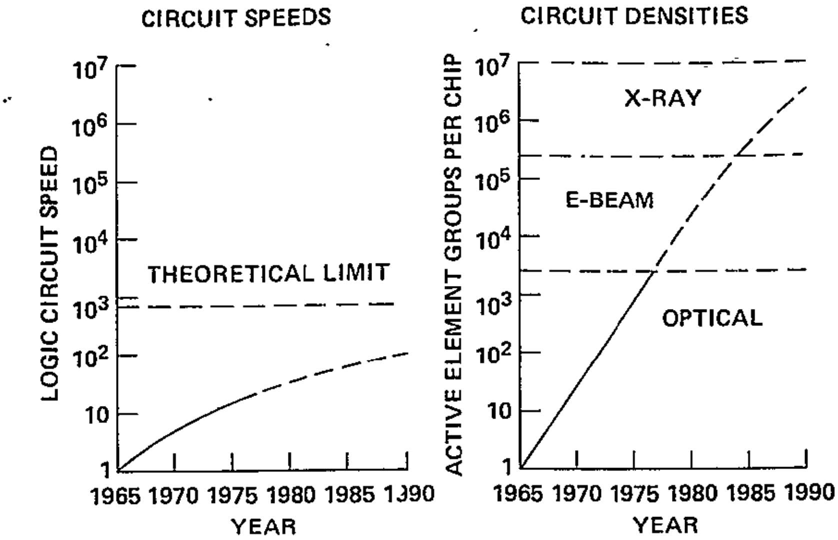


Figure 19.- Improvements in electronic technology relative to the state-of-the-art in 1965.

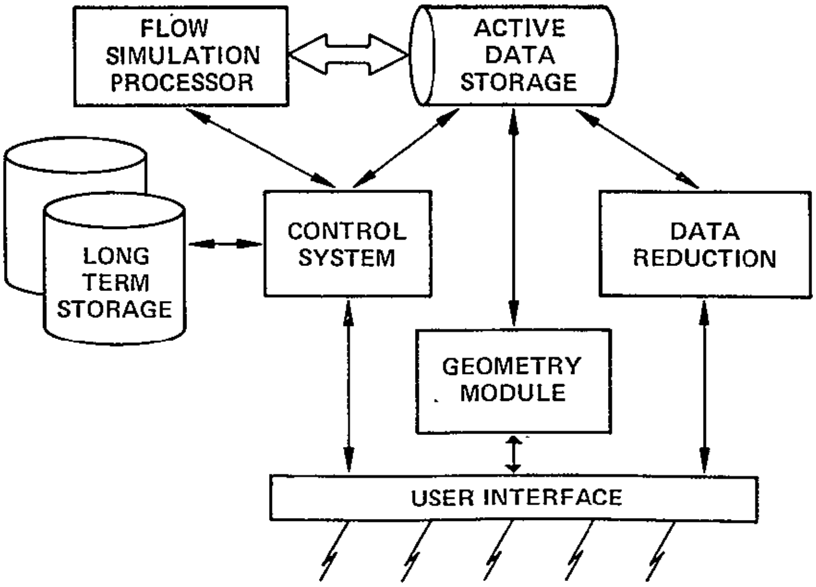


Figure 20.- Schematic diagram of the Numerical Aerodynamic Simulation Facility.

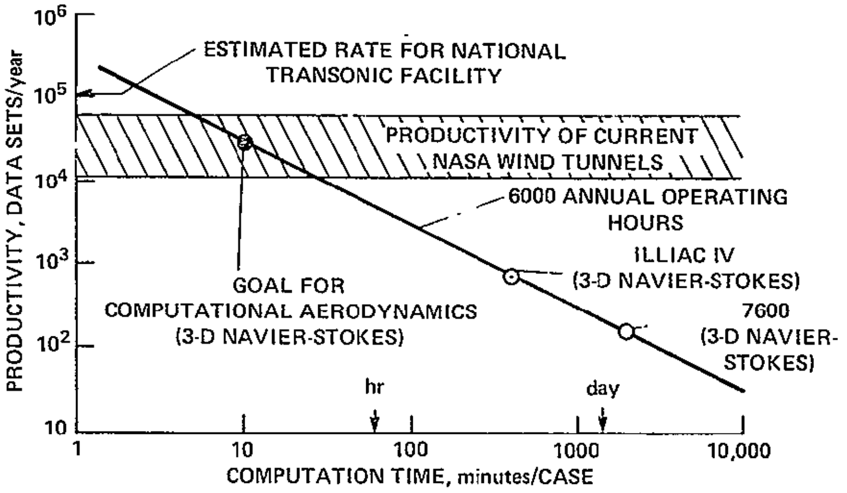


Figure 21.- Relationship between the productivity of numerical facilities and wind-tunnels.

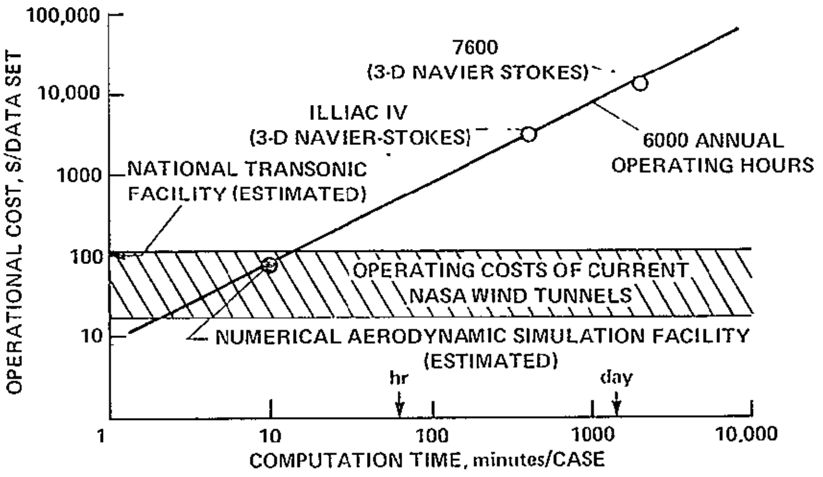


Figure 22.- Operational costs for numerical facilities and wind-tunnels.

FY-77

- CONCEPT DEFINITION – 2 CONTRACTORS
  - COMPUTER TECHNOLOGY SURVEY
  - PROCESSOR – FLOW MODEL MATCHING
  - PRELIMINARY FACILITY DESIGN
- AIRCRAFT INDUSTRY BRIEFINGS
- ADVISORY COMMITTEE BRIEFINGS (RTAC's, ASEB)

FY-78

- PRELIMINARY DESIGN – 2 CONTRACTORS
  - FUNCTIONAL DESIGN OF PROCESSOR
  - SYSTEM SPECIFICATION AND SIMULATION
  - COST/PERFORMANCE TRADE OFF FOR WEATHER/CLIMATE CODES
- ASSESSMENT OF NASF UTILITY FOR RELATED DISCIPLINES
- WORKSHOP

FY-79

- BEGIN FINAL DESIGN OF FACILITY

Figure 23.- Activities in the Numerical Aerodynamic Simulation Facility project.

STRONG GENERAL ENDORSEMENT:

- "SUCCESSFUL ACHIEVEMENT OF PROJECT GOALS WOULD HAVE A MAJOR IMPACT ON THE AERODYNAMIC DESIGN PROCESS OF FUTURE AIRCRAFT" (NORTHROP)
- "IT IS A PROPER ROLE FOR NASA TO PROMOTE THE ADVANCEMENT OF COMPUTATIONAL TECHNOLOGY IN THIS MANNER" (BOEING)
- "WE ENTHUSIASTICALLY SUPPORT YOUR PLANS FOR THE DEVELOPMENT OF SUCH A FACILITY AND YOUR CONTINUED EMPHASIS ON COMPUTATIONAL AERODYNAMICS" (LOCKHEED, GA.)
- "THE PLANNED FACILITY SHOULD PROVIDE A VALUABLE TOOL TO THE ENTIRE AEROSPACE INDUSTRY" (VOUGHT)
- "WE AGREE WITH THE GOAL OF EFFICIENT, COMPLEMENTARY USE OF COMPUTER AND WIND TUNNEL SIMULATION FACILITIES" (McDONNELL DOUGLAS)

PRINCIPAL CONCERNS:

- POLICY FOR USE OF NASF BY INDUSTRY
- PROPRIETARY SECURITY
- GEOMETRY MODULE

Figure 24.- Sample of aircraft industry response to briefings on the Numerical Aerodynamic Simulation Facility project.

SESSION 2

F. R. Bailey, Chairman

N78-19780

COMPUTING TECHNOLOGY IN THE 1980s

Harold S. Stone
Visiting Professor
University of California
Berkeley, California 94720

Advances in computing technology have continued to be led by consistently improving semiconductor technology. The semiconductor industry has turned out ever faster, smaller, and less expensive devices since transistorized computers were first introduced 20 years ago. For the next decade, there appear to be new advances possible, with the rate of introduction of improved devices at least equal to the historic trends. The implication of these projections is that computers will enter new markets and will truly be pervasive in business, home, and factory as their size and cost diminish and their computational power expands to new levels.

Perhaps the most innovative sector of computing technology is the microprocessor-based computer system. These are only just becoming available today. The phenomenon of the computer hobbyist and the computer store has created a temporary industry centered around home-built kits. In recent months the first introductions of home computers reached the market. These are completely assembled computers with keyboard, video display, cassette memory, monitor, and a BASIC software package offered at prices well under $1000. Projecting the advances in devices forward into the next decade indicates that a computer offered in the same physical package and at about the same price could easily have a 256K byte main memory and an auxiliary memory with several megabytes. The software offered could easily match that of a typical 1977 small business system costing in the range of $50,000.

While the possible applications for microprocessor-based systems with this kind of computational power are virtually unlimited, the rapidity with which the industry is changing may be the main damping factor with respect to new innovative products. A new idea can be marketed successfully only during the period of time before the idea becomes obsolete. Change comes so rapidly in the computer industry that the "window" for marketing some high-technology products may be shrinking from a few years to less than one year. If an idea is risky so that development efforts could potentially be delayed, the eventual appearance of that product may occur after the window is closed, and it becomes obsolete before its first announcement. We have seen these forces at work in hand calculators, digital watches, and video games. In the next decade, we shall encounter dozens more examples, and some innovations will simply go unpursued because of the risk.

The computer industry as we know it today will be greatly altered in the next decade, primarily because the raw computer system--the bulwark of the industry today--will give way to computer-based turn-key information and control systems. Even today it is possible to purchase an "automated office" with limited capability for filing, text preparation, accounting, and sales analysis. A decade of evolution of this device will make it an office fixture much like a typewriter and filing cabinet. The facets of the computer internal to such a device are relatively unimportant compared to the business oriented functions it performs. The user will probably not purchase the system by specifying such things as the size of memory, the power of the CPU, and the performance of the auxiliary memory. Rather he will specify whether or not he wants the inventory control package, the billing package, the electronic mail package, etc., as if these are simply extra keys on a keyboard. This places a burden on the present computer industry to provide fully functional integrated applications systems instead of raw computing power.

In terms of computer architecture, there are several major trends that will evolve over the next decade. Among these are:

1. very large main memories (8M bytes to 32M bytes), with much less importance given to management of memory as a precious resource,
2. diminished emphasis on time-sharing and multiprogramming to share resources with a corresponding rise in the use of the dedicated computer system,
3. wide proliferation of interconnected computers primarily to access common data bases, and
4. increased use of multiple processors within a single system with a tendency to dedicate particular processors to particular system tasks.

Because the costs of the hardware are predictably decreasing by a factor of 10 per decade (or sooner), truly great strides in the next decade will be hampered if the costs of software do not diminish at a comparable rate. Software technology has surely not matched hardware technology in terms of productivity increases. Software has actually tended to increase in cost in some sectors due to costs associated with larger and more ambitious projects being undertaken than have been undertaken previously. Nevertheless, high-level languages and, more recently, structured programming have improved programmer output to indicate that the potential for less expensive software development is there. But what is often overlooked is that the size of the software market is increasing so rapidly for the less expensive systems, that the cost of software to the user can be made negligible if he is willing to use software common to tens of thousands of other systems. So it is conceivable in some applications areas that for a few hundred dollars one could purchase an enormously powerful computational device plus all of the applications software required to solve a particular class of applications.

High-speed computing systems, unfortunately, lack the large base to share the cost of software development. Consequently, the next decade will find the super computers rather inexpensive in present terms, while support software will see very little change in cost. Programs will undoubtedly be hand-tailored for maximum efficiency then as they tend to be now. The net effect of technology advance for high-speed computing will thus be felt in the size of the problem attempted and in its running time, but the high cost of software development will probably lead to very little impact on the expenditure totals as compared to the rapidly decreasing expenditures experienced elsewhere in the computer industry.

D3

N78-19781

THREE-DIMENSIONAL COMPUTATIONAL AERODYNAMICS IN THE 1980's

Harvard Lomax

Ames Research Center, NASA

1. INTRODUCTION

The future requirements for constructing codes that can be used to compute three-dimensional flows about aerodynamic shapes should be assessed in light of the constraints imposed by future computer architectures and the reality of usable algorithms that can provide practical three-dimensional simulations. On the hardware side, it appears that vector processing is inevitable in order to meet the CPU speeds required. Furthermore, in order to cope with three-dimensional geometries, it appears that massive data bases with fetch/store conflicts and transposition problems are inevitable. On the software side, it is clear that we must be able to prepare codes that: (1) can be adapted to complex geometries, (2) can (at the very least) predict the location of laminar and turbulent boundary layer separation, and (3) will converge rapidly to sufficiently accurate solutions.

2. FUTURE SCIENTIFIC COMPUTERS

The approximate capabilities of several existing or possible scientific computers arelisted in Table 1.

Year	1968	1973	1978	1983	
MFLOPS*					
(Peak)	10	60	100	400	3000
(Expected)	3	20	30	130	1000
Computer	CDC 7600 IBM 360/195	ILLIAC IV STAR 100 ASC	CRAY I BSP STAR 100A	CRAY II BSP II STAR 100C	NASF

TABLE 1. Scientific computer CPU speeds.

*Million floating point operations per second.

The estimates for CRAY II and BSP II are what could be expected if the CRAY and BURROUGHS Corporations were to enter another generation of their present products. Shown in the second column from the right is what can be expected from "conventional" computers, and in the last column on the right is what we expect from a Numerical Aerodynamic Simulation Facility. The above addresses the raw computing speeds existing on present, and expected for future computers. Next we consider some other important aspects of expected conventional computers and contrast them with the requirements being considered for NASF.

The memories for conventional 1983 scientific computers are expected to be about 4 million words of "random" access and 400 million words of "rotating" backup. The NASF is expected to have about 8 million words of "random" access and up to 200 million words of "block addressable" backup. The block addressable aspect is explained below.

The operating systems of future conventional scientific computers are expected to be standard, multi-task systems with time-slice interrupts and a wall-to-clock execution ratio of around 1 to 8. On the other hand, the NASF is expected to have a simple operating system that runs single tasks to completion with a wall-to-clock execution ratio of 1 to 1.

The compilers for future conventional computers can be expected to interpret many forms of advanced, high-level, vector-extended, scientific programming languages. In contrast, the NASF (for cost and time constraints) will probably interpret only simple, vector-extended FORTRAN.

One can reasonably expect that the conventional 1983 scientific computers will be used extensively for many forms of two-dimensional, steady and unsteady, flow simulations, and on some forms of practical three-dimensional flow simulations. Examples of the latter would be three-dimensional transonic flows based on a velocity potential with some form of viscous interaction. However, three-dimensional flow simulations based on some form of the Navier-Stokes equations, with practical boundary conditions and sophisticated geometries would be computed much more effectively on the NASF. These latter simulations would involve compressible or incompressible flows at high Reynolds numbers with turbulence modeling at body surfaces and in the separated regions.

3. EXPECTED GEOMETRIES AND TURBULENCE MODELS

It is important that practical 3-D aerodynamic simulations accurately represent realistic geometries, such as complete wings and bodies, complete wing-body combinations, and complex component parts, such as body, nacelle, and jet-exhaust combinations. At the same time, production-type, user-oriented codes must be provided with algorithms that are reliable throughout the computational domain. It appears inevitable that the governing equations will be cast in a coordinate systems that transforms a complicated domain in Cartesian (x,y,z) space to a very simple (e.g., rectangular) domain in the (ξ,η,ζ) computational space. If we define the transformations

$$\xi = \xi(x,y,z) \qquad \eta = \eta(x,y,z) \qquad \zeta = \zeta(x,y,z) \tag{1}$$

and form the Jacobian (local mesh volume)

$$\mathcal{D} \equiv \det \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \tag{2}$$

the conservative form of the Euler equations including all effects of geometry can be written

(a) For the dependent variables

$$\hat{Q} = \begin{bmatrix} \rho/\mathcal{D} \\ \rho u/\mathcal{D} \\ \rho v/\mathcal{D} \\ \rho w/\mathcal{D} \\ e/\mathcal{D} \end{bmatrix} = \begin{bmatrix} \hat{\rho} \\ \widehat{\rho u} \\ \widehat{\rho v} \\ \widehat{\rho w} \\ \hat{e} \end{bmatrix} \tag{3}$$

(b) For the fluxes

$$\hat{E} = \begin{bmatrix} \hat{\rho} U \\ \widehat{\rho u} U + \hat{p}\xi_x \\ \widehat{\rho v} U + \hat{p}\xi_y \\ \widehat{\rho w} U + \hat{p}\xi_z \\ (\hat{e}+\hat{p})U - \hat{p}\xi_t \end{bmatrix} \qquad \text{where} \quad U \equiv \xi_t + u\xi_x + v\xi_y + w\xi_z \quad \text{and } \hat{F} \,\&\, \hat{G} \text{ have similar constructions.} \tag{4}$$

(c) For the conservation of mass, momentum, and energy

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} + \frac{\partial \hat{G}}{\partial \zeta} = 0 \tag{5}$$

These equations are relatively simple to code for both explicit and implicit algorithms, they permit easy application of boundary conditions for complicated geometries, and they are readily adaptable to clustered meshes in physical space that correspond to uniform meshes in computational space. The Euler equations (5) describe flows having both convection and pressure forces. To these equations one must add the process of diffusion brought about by viscosity and heat conduction or turbulent transport, if turbulence is modeled. The addition of such terms forms the Navier-Stokes equations or various approximations of them. It is important that practical 3-D aerodynamic simulations of high Reynolds number flows accurately predict the lines along which the turbulent boundary layer separates from the body surface.. Turbulence models that can do this are in various stages of development. How well the turbulence is modeled after separation occurs is argumentative and the importance of such modeling is problem dependent.

Conventionally, finite difference codes use highly stretched meshes throughout the thin turbulent boundary layer. This requires mesh clustering along surfaces nearly parallel to the entire aerodynamic shape. It is well known that this clustering, which is a part of the turbulence model itself, as well as the clustering brought about by geometry considerations, can lead to a numerical problem referred to as stiffness which is discussed next.

4. EXPECTED ALGORITHMS

Stiffness refers to a numerical problem caused entirely by the discretization of ordinary or partial differential equations. Among other things, it can be caused when very fine meshes are used in evaluating space differences. Stiffness occurs when a time (or iteration) step is forced, for numerical stability reasons, to be very small relative to the time (or iterative) variation of the solution itself. The conventional way to avoid stiffness is to employ implicit, rather than explicit, methods.

If we construct the numerical difference operators L and R and use n for the time index, the Navier-Stokes equations can be reduced to the symbolic form

$$(L_{\xi+\eta+\zeta})\hat{Q}^{n+1} = (R_{\xi+\eta+\zeta})\hat{Q}^{n} \tag{6}$$

The operators L and R are in fact very large banded matrices having a rank equal to the sum of all the points in the mesh. Now since the $\hat{Q}^n$ data are known over the entire mesh at step n, the operation $(R_{\xi+\eta+\zeta})\hat{Q}^n$ is an explicit calculation which can easily and efficiently be carried out in a variety of ways regardless of the rank or form of the matrix R. However, the evaluation of $(L_{\xi+\eta+\zeta})\hat{Q}^{n+1}$ requires the solution of simultaneous equations and, even if the matrix is sparse and banded, this evaluation is far from trivial for the rank given above.

At this point one is caught in a dilemma. To remove stiffness, we can employ implicit methods; but implicit methods can be very costly to evaluate. The dilemma is partially, if not entirely, removed by the process of factorization. One can show that reasonable (at least second-order) accuracy can be retained if the L operator is split into the product of three operators

$$(L_{\xi+\eta+\zeta}) \simeq (L_{\xi})(L_{\eta})(L_{\zeta}) \tag{7}$$

where the L_ξ, L_η and L_ζ are block tridiagonal matrices having individual ranks equal only to the number of points along the side of the rectangular computational mesh. This greatly simplifies the solution process as well as its cost, and makes the application of block or scalar implicit techniques quite practical. Fortunately, this practicality extends to future computer architectures since large vectors can readily be identified in factored forms of eq. (7) when these are used for three-dimensional flow simulations.

5. VECTOR IDENTIFICATION IN 3-D SIMULATIONS

The space-operator factorization shown in eq. (7) has the practical effect of reducing the solution process to three, successive, one-dimensional sweeps through the data base. Let us investigate a consequence of this in the NASF discussed in section 2. We mentioned that this might be composed of two types of memories connected by high band-width lines. Refer to these as extended core, randomly "block" addressable, and main core, randomly "word" addressable (both of which might have memory bank conflicts which are ignored here).

The extended core can be filled with consecutive strings of data each of which represents an N^3 block of data in the computational space. These blocks can be brought into the main core in column formation in any of the three ξ,η,ζ directions, two of which are shown in figure 1(a).

After arriving in main core, we refer to one of these columns as a pencil as shown in figure 1(b). Consider next N^2 planes of data in the pencil such as $\vec{Q}_5$ and $\vec{Q}_{20}$ in the figure. Data entering from extended core in one of the three orientations is sequentially aligned to represent the variables in each of these planes without further manipulation. This orientation is referred to as the natural direction. However, data brought from columns in the other two directions must be reordered to form sequential representations of the planes in the pencil. The nature of this reordering from a strictly serial storage is illustrated in figure 1(c). Once the data has been reordered, in each block of the pencil, the algorithms proceed as if the pencil had been formed from the natural direction.

After the data in the pencil has been properly aligned, the N^2 string of data lying in any given plane can form the critical "vector" in any of the various computer architectures that are expected for the future, either conventional or NASF. Since the factorization uncouples the three directions, all forms of scalar or block matrix operations can be carried out up and down the pencil, including the boundaries, making full use of the N^2 vector length in the computer hardware. It should be mentioned also that, when computing in a given direction, the order in which the pencils are brought in from, and returned to, extended core is immaterial.

6. CONCLUSIONS

It is expected that the computational problems which confront three-dimensional aerodynamic simulations in the oncoming future will involve complicated geometries and mesh generations, factored implicit algorithms, and sophisticated turbulence models.. The solutions of these problems are constrained to techniques that work efficiently in vector processors. So far there is every reason to believe that future hardware and software can be made to work together to solve many practical aerodynamic problems.

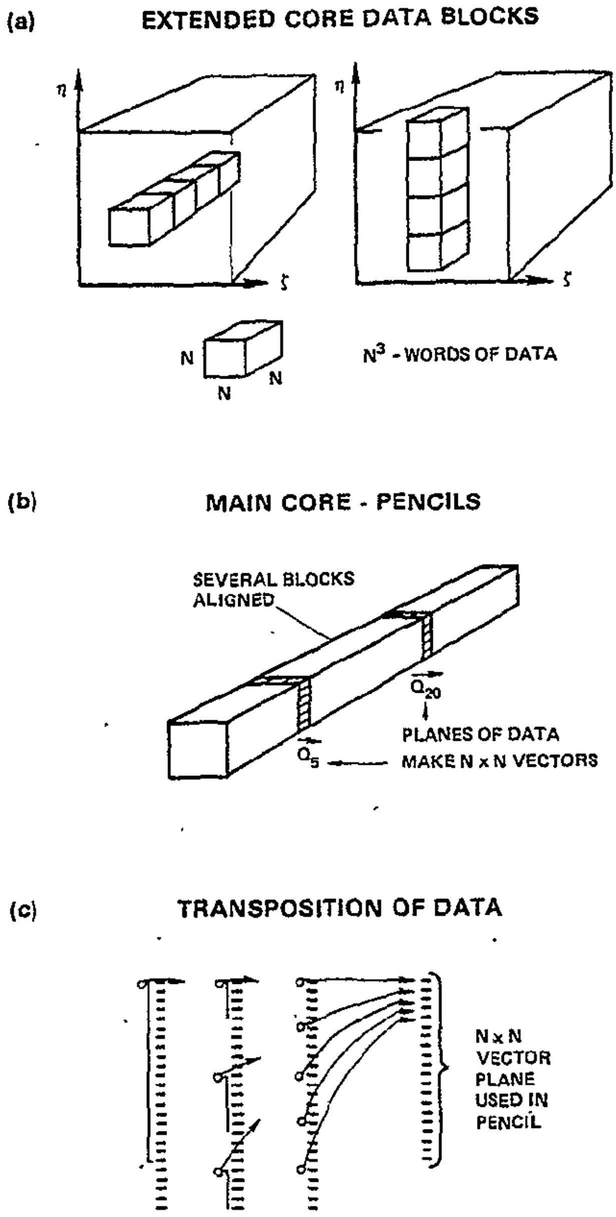


Figure 1.- Vector identification in three-dimensional data sets.

SESSION 3

SUMMARY REPORTS OF PRELIMINARY STUDY FOR A NUMERICAL AERODYNAMIC SIMULATION FACILITY

F. R. Bailey, Chairman

FINAL REPORT

NUMERICAL AERODYNAMIC SIMULATION FACILITY

PRELIMINARY STUDY

October 1977

Distribution of this report is provided in the interest of information
exchange. Responsibility for the contents resides
in the author or organization that prepared it.

EXECUTIVE SUMMARY

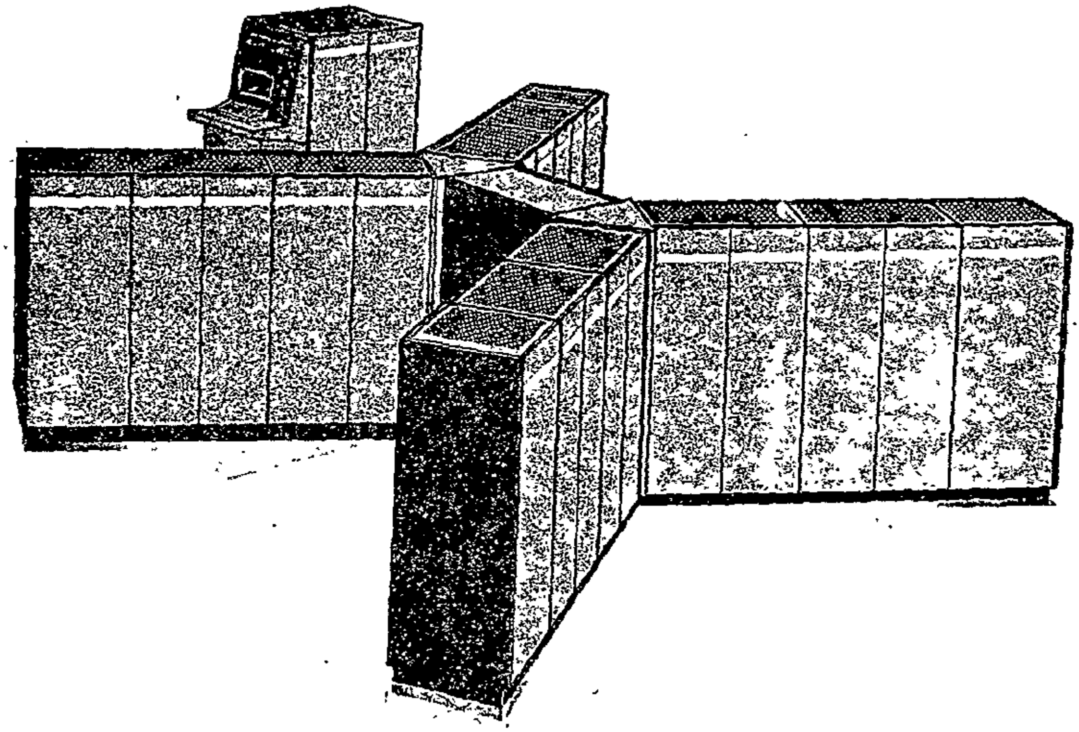
Prepared under Contract No. NAS2-9456 by
Burroughs Corporation
Paoli, Pa.

for

AMES RESEARCH CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

39N

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



NAVIER-STOKES SOLVER

INTRODUCTION

Burroughs Corporation is pleased to submit this executive summary of the findings of the Numerical Aerodynamic Simulation Facility (NASF) Preliminary Study. This report presents a unique solution to the problem of numeric aerodynamic simulation. The solution consists of a computing system designed to meet the stated objective of providing an effective throughput of one billion floating point operations per second for three-dimensional Navier-Stokes codes. Burroughs presents this design with full confidence that it is feasible to complete the detailed design and construction of this machine within the required time-frame. This high level of confidence is based on Burroughs' extensive and continuing experience in the design and development of very high performance computer systems. It is Burroughs' belief that the computer industry will not produce a commercial general purpose machine with the required performance by the early 1980's. Consequently, we feel that the design and construction of a relatively specialized system is not only feasible, but necessary to the achievement of NASF objectives.

This view is based on two business judgements. First, projections of both computing-power and cost of performance of commercial computers for the 1980 to 1985 time-frame do not include a machine of this capacity or price. That is, a generation gap will exist between any NASF implementation and concurrent commercial products. Second, market trends indicate that an insufficient market exists to sustain development of a machine with two orders of magnitude speed increase on a commercial basis.

In summary, we believe that the system presented in this report constitutes the best approach to meeting the NASF goals in a timely and cost-effective manner, and that NASA has an opportunity to maintain a "forefront" position in the scientific community while achieving these goals.

The results of this study have produced a unique solution to the problem of numerical aerodynamic simulation of three-dimensional Navier-Stokes equations. In order to fully appreciate the design, its features, and subtleties, the methodology of the study which evolved this solution must be understood. This executive summary is intended to explain that methodology. First, the problem and solution, in brief, will be presented, then basics of the study approach will be explained. Next, a description of each of three substudies follows with emphasis on specifically what was examined and why. Finally, the results of the substudies are merged to highlight their impact on the processor architecture evolution, and show how the "baseline design" for NASF was selected. The final report chapters will discuss details of that design.

STUDY OBJECTIVES

The Numerical Aerodynamic Simulation Facility Preliminary Study Objectives were to determine the feasibility of designing a system delivering one billion floating point operations per second effective throughput for three-dimensional Navier-Stokes codes by 1982. If feasible, a processor architecture and functional design definition were to be developed, supporting that assertion, with attendant requirements of power, size, cost, schedule, etc.

NASF OVERVIEW

The basic structure of the candidate baseline NASF system is shown in Fig. 1. The major elements are:

- The Host, a Burroughs B7800 multiprocessing system
- The Navier-Stokes Solver (NSS) . . . the high throughput work-horse of the system
- File Memory
- An Archival Storage system.

THE HOST COMPUTER

The Host, a Burroughs B7800 system, acts as the system manager and support facility. It provides the user interface, schedules and dispatches NSS tasks, and executes supporting functions such as compilation, data reduction, and output generation.

THE NAVIER-STOKES SOLVER (NSS)

The NSS is the high throughput computational element. It is a highly parallel processing array, designed to provide the required computational throughput on three-dimensional Navier-Stokes programs. The Data Base Memory (DBM) of the NSS provides the interface between the NSS and other system elements. The program and data files are loaded to the DBM by the Host. The NSS with the DBM constitute a high speed "computational envelope," allowing the NSS to run at maximum speed essentially without outside interruption or dependence until job completion.

THE ARCHIVE MEMORY

The Archive provides a very large storage capability for long term retention of programs and data bases. It consists of a commercially available mass memory system, which is managed by the Host.

THE FILE MEMORY (FM)

The FM provides for short term file retention, staging and buffering between the Host, the Archive, and the DBM. It consists of a standard disk pack sub-system, and is also managed by the Host.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

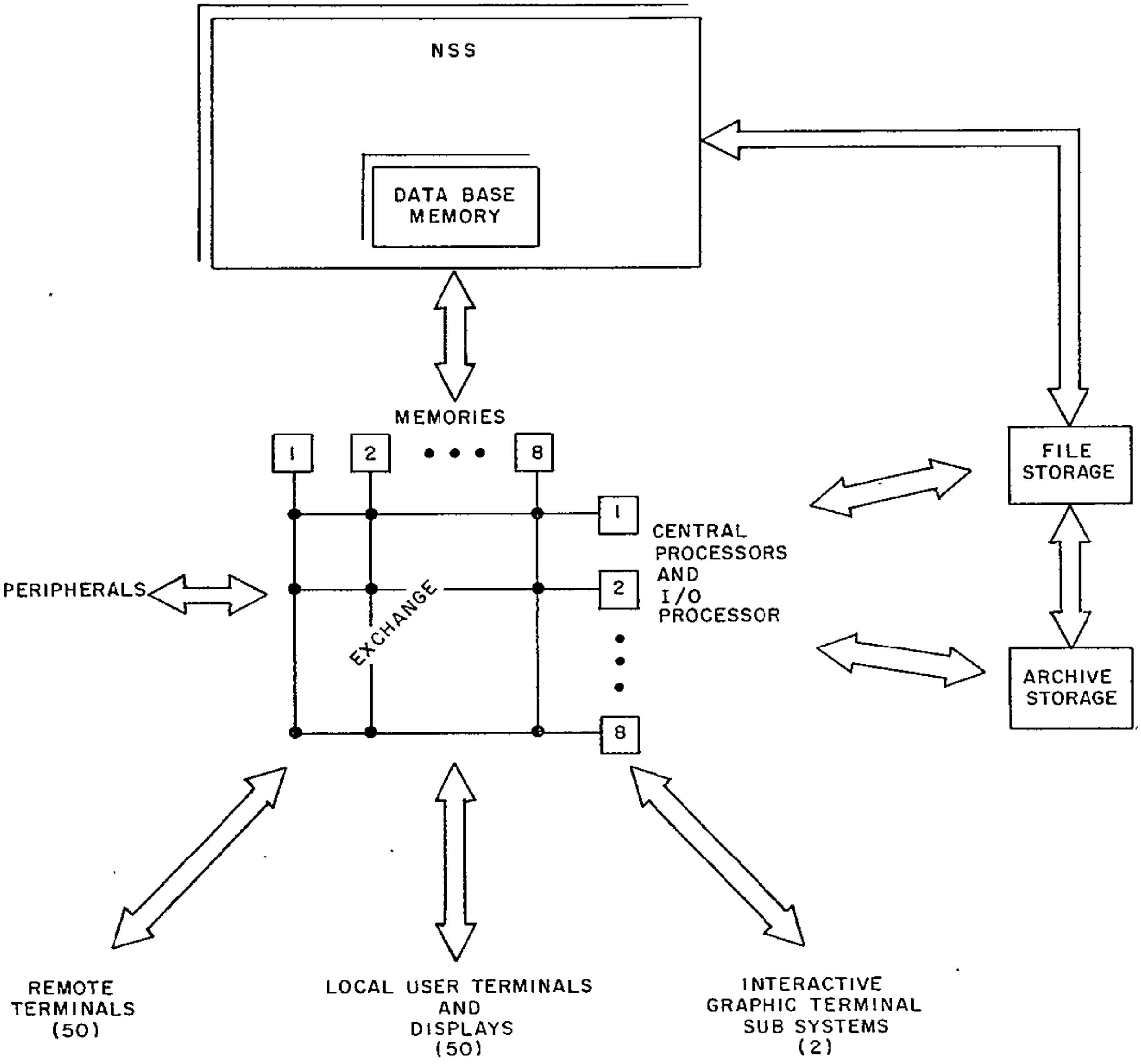

NSS
DATA BASE MEMORY
MEMORIES
1
2
8
1
2
8
CENTRAL PROCESSORS AND I/O PROCESSOR
EXCHANGE
PERIPHERALS
FILE STORAGE
ARCHIVE STORAGE
REMOTE TERMINALS (50)
LOCAL USER TERMINALS AND DISPLAYS (50)
INTERACTIVE GRAPHIC TERMINAL SUB SYSTEMS (2)

Figure 1 NASF System Block Diagram

MAJOR ELEMENTS OF THE NAVIER-STOKES SOLVER

The principal innovation in the NASF system is the NSS. The organization of the NSS is shown in Figure 2, and its characteristics are summarized in Table 1. The major features of this processing array are:

- Highly Parallel Architecture
  The NSS consists of 512 computational processors, each with its own local data program memories. These are coordinated by a single control unit, and connected via a transposition network to 521 modules of extended memory.

- Synchronizable Operation
  This feature of the NSS suggests the name we have given to the computational array, the Synchronizable Array Machine, or SAM. Previous processor arrays have operated in "LOCKSTEP," essentially synchronizing on every instruction cycle. The computational array of the NSS is synchronized explicitly by the code stream only when necessary. Between synchronization points, the individual processing elements may operate asynchronously, allowing them a degree of freedom in scheduling instruction sequences.

- Conflict Free Memory Access
  The transposition network between the processing elements and extended memory allows conflict free access to vectors in any dimension at full memory bandwidth. This eliminates the non-productive time which would otherwise be consumed by reordering or transposition of data before processing.

- Large Second Level Store
  The Data Base Memory (DBM) in the NSS provides an interface between NSS and Host that allows each to process independently of the other. NSS processing need never be held up waiting for some response from the Host.

- System Balance
  All transfer rates and execution speeds are tuned to one another in concert with the requirements of the application. This provides for high efficiency by balancing the utilization of system elements.

- Ease of Use
  A high level user language, complemented by an instruction set oriented to efficient implementation of high level language programs, allows ready access to the computational power of the NSS, without encumbering the user with assembly language programming or implementation details.

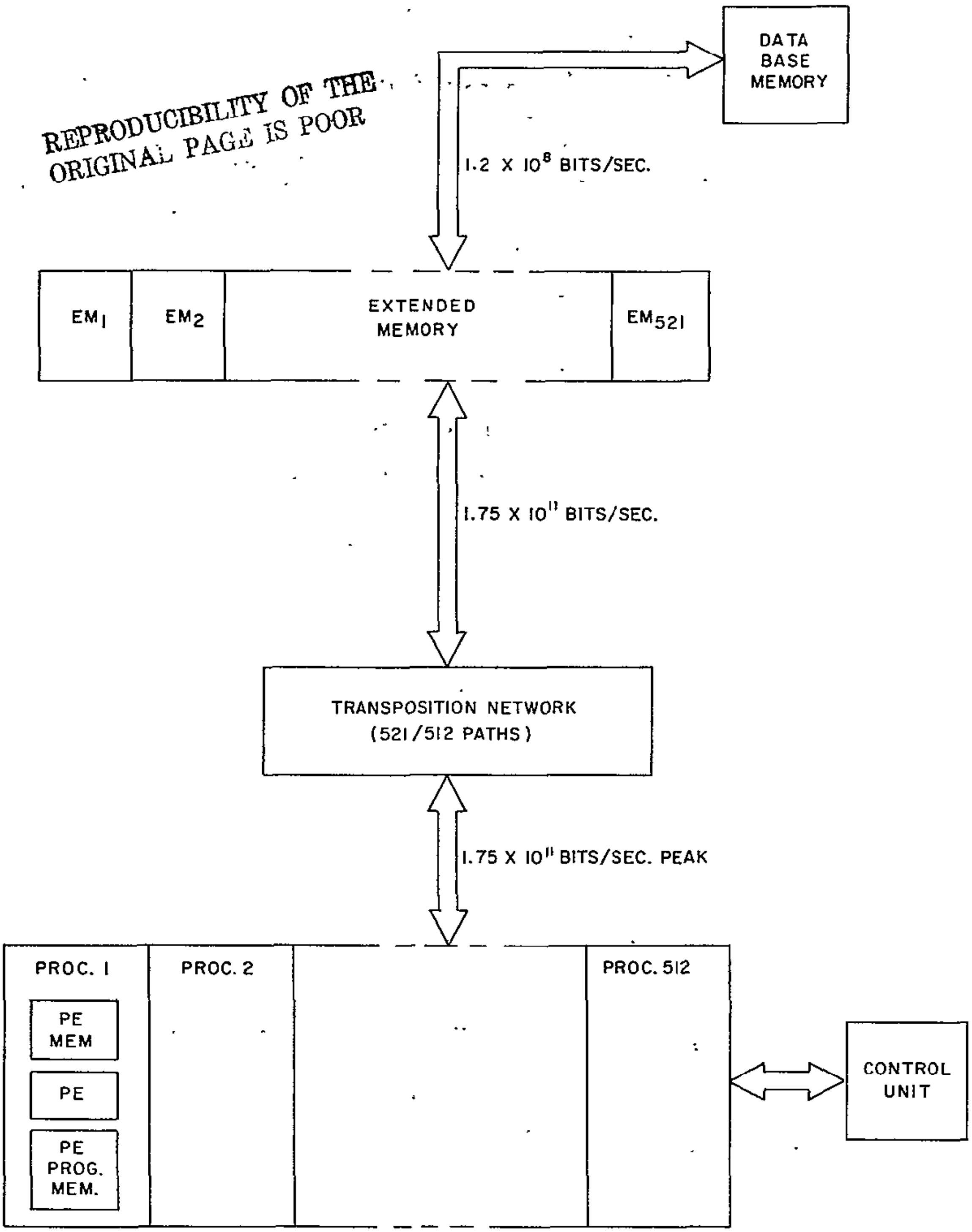


Figure 2 SAM Block Diagram

TABLE 1 NSS CHARACTERISTICS

Computational Capacity (On instruction mix)	1.7×10^9 floating operations/sec.		
Number of Processing Elements Number of Extended Memory Modules	512 521		
Memory capacities (total)			
Extended memory	34 million words		
Processing element memories	8 million words		
Processing element program memories	4 million words		
Transfer rates (bits/sec)	per path	no. paths	total
PE - PEM	490×10^6	512	2.5×10^{11}
PE - PEPM	490×10^6	512	2.5×10^{11}
PE - (PEM+PEPM)	10^9	512	5×10^{11}
EM - via TN - PEM			
streaming mode	4×10^8	512	2×10^{11}
1 word/transfer	1×10^8	512	5.5×10^{10}
EM - DBM	—	—	1.4×10^8
Program loading to all PE's simultaneously			4×10^8 per PE
Clock, synchronous throughout the NSS	50 MHz minor cycles 25 MHz major cycles		
Total No. of IC packages, including memory (almost all LSI)	200,000		
Word Size:	48 Bits		

STUDY METHODOLOGY

Experience in the design and manufacture of data processing equipment, especially very-high-performance computer systems, leaves many lessons behind. In addition to knowing what a design team should do, there are some lessons about what should **not** be done.

The Burroughs study team took care to avoid a serious problem that often traps those aiming at maximum speed — namely pushing the state of the art on too many frontiers. One could rely on significant advances in

- Architecture,
- Hardware Technology, or
- Software Technology.

For increased performance Burroughs chose Advanced Architecture taking care to build on mature or developed software whenever possible. In addition hardware implementation will be conservative, consistent with performance goals, and will not rely on imposing inordinate speed requirements or new, untried technologies.

Selecting architectural elegance as the new frontier, the study concentrated on matching the architecture to the problem. Existing computer structures were not integrated to force-fit a "super-structure" of these units to the problem. The reasons were:

- Lack of Architectural Flexibility
- Inefficient and Not-Cost-Effective.

Although performance requirements may be met in this fashion, the lack of architectural freedom with the structures implies that many hardware and software elements are not utilized, others must be customized, resulting in a machine that has some "dead-wood."

The NASF system presented here was developed by evolution from careful analysis of the problem characteristics to insure a genuine fit. Top-down design fundamentals were practiced so that on each of the several design iterations, results could be traced to assumptions. Traceability of this sort allows bottlenecks or errors found to be identified at their origin where viable alternatives could be reexamined.

SUB-STUDIES

Specifically, three sub-studies were executed simultaneously as required by the original contract statement of work.

- THE TECHNOLOGY STUDY developed a data base of logic and memory technologies by literature searches, vendor interviews and conferences, etc. Trends of critical issues and parameters of these technologies were studied and a technology forecast developed for the 1980-1985 time-frame.

- THE MATCHING STUDY analysed the flow models and their characteristics and matched them against candidate processor architectures.

- THE FACILITY STUDY established metrics for the total facility and, at a more detailed level, the facility issues addressing the "buildability" of the final system.

Each sub-study was executed with two objectives as shown in Figure 3.

- How do results affect processor architecture choice?
- How do results affect specific design choices in the baseline design?

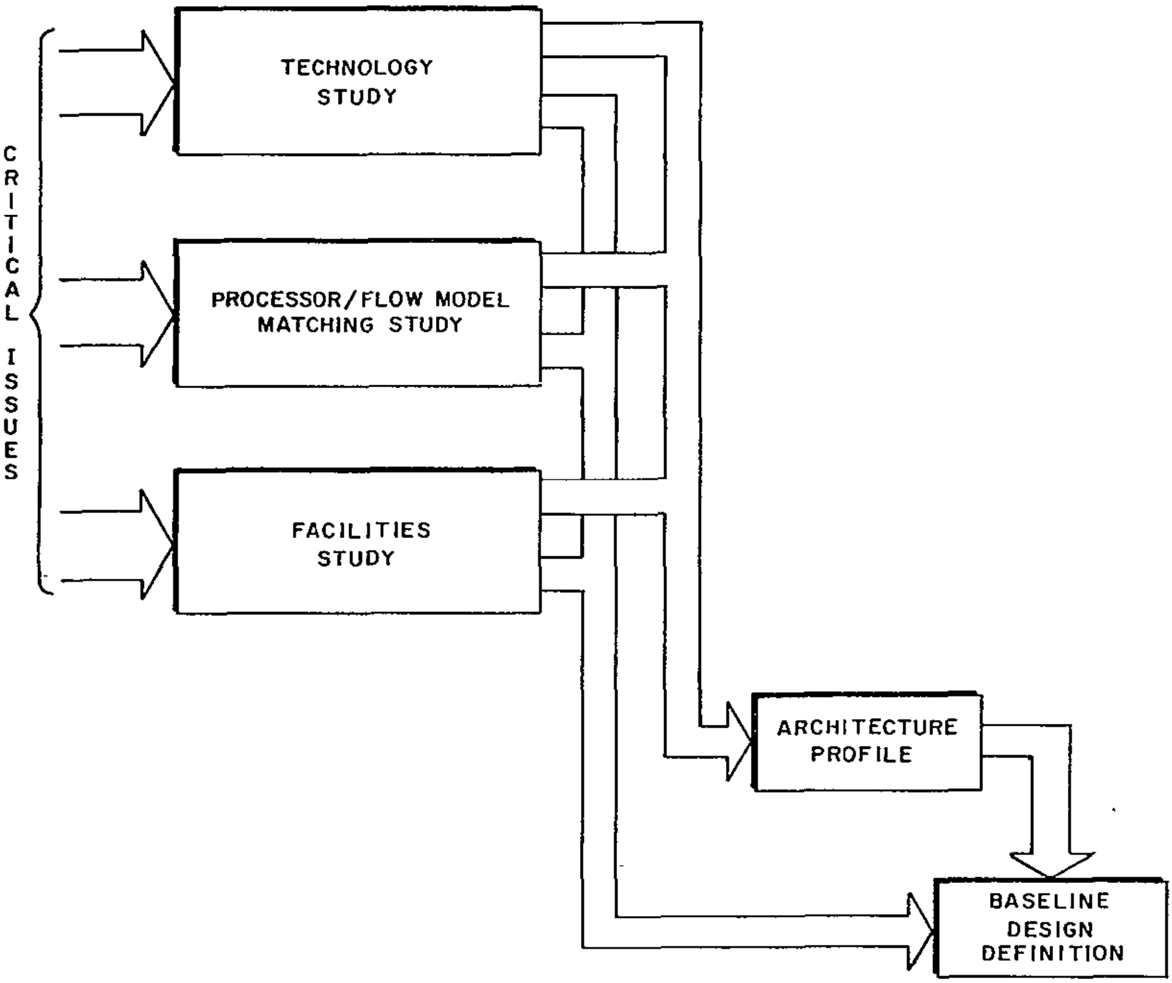


Figure 3 NASF Study Approach

That is, first a processor architecture was evolved as a result of the sub-studies, then a second iteration of the studies supported a more detailed design to the functional design level referred to as the Baseline Design. The result is an NASF definition that directly addresses the salient issues of the problem itself. This NASF definition meets or exceeds all requirements and can be built with a high degree of confidence - an assertion of great significance for such an ambitious task.

TECHNOLOGY STUDY OVERVIEW

The objective of this phase of the study was to establish a technology forecast for the NASF time-frame and assess which logic and memory technologies are most appropriate for the design of such a facility.

The approach taken consisted of the following four tasks:

- Data Gathering
- Establish Critical Issues
- Examine Technologies & Trends
- Extrapolate 1980-85 Forecast.

Data gathering consisted of a three phase effort: a comprehensive literature search, trade conferences and workshops, and interviews with vendors and suppliers such as Motorola, Fairchild, National Semiconductor, Intel, Signetics, and Texas Instruments.

The critical issues which were established were of two types - those affecting performance and those affecting development.

PERFORMANCE

- speed
- density
- reliability
- power

DEVELOPMENT

- cost
- maturity
- extensiveness
- availability

Metrics for judgement of these issues and clarifications of their importance were then developed and used as criteria in the architecture/design process.

Under performance issues, speed of a logic family may be judged by propagation delay times, while with memory the key figures are read/write times. Density refers to the average number of gates or memory cells per chip. Reliability is largely a function of density since failures frequently occur at the substrate to pin connection, and as the number of pin connections decreases per given function, the reliability increases. Power consumption is a measure of the energy costs and reliability associated with a device. A smaller speed-power product indicates better system performance per kilowatt.

As to developmental issues, cost should be considered in the light of performance per dollar, as well as absolute cost. Maturity is determined by field verification of manufacturer's specification. Another consideration in selecting a technology, is the availability of the devices. In addition, multiple sources for all componentry are essential. These factors are important considerations in the selection of a technology family.

The technology survey provided inputs to the study not only in the obvious area of surveying the implementation of digital logic, but also in some areas of packaging, random access and serial memories, and archives.

From the many technologies used to implement digital logic, three are of sufficient interest to report here:

- ECL has been the technology of choice in implementing high-speed digital computers for over ten years. The speed-power product, and hence the amount of processing that can be done per watt of power, has been continually improved, and in the last year some LSI has been available in ECL. ECL is a mature but still developing technology, exemplified by Fairchild's "100K" ECL family. This family could be used as a starting point for a baseline design.

- I^2L has much better speed-power product than ECL, allowing far more functions per watt. It is currently too slow for the NASF requirement but both speed and availability of standard parts are improving each year. I^2L would consume considerably less power than ECL and is currently utilized internally in LSI chips where the speed is tolerable.

- MESFETs promise another improvement, by an order of magnitude, in the speed-power product as compared to I^2L. They are also very fast; however, they are still in early development. Years of development will be required before the MESFET's technology becomes mature.

From this study we conclude that ECL is the most feasible current technology for implementation of an NASF design, and the base line design will begin with ECL as a starting point.

Memory technology represents an area of low risk for the (NSS). 16K-bit dynamic RAM's (Random Access Memory) are currently available. 16K-bit static RAMS and 64K-bit dynamic RAMS are on the drawing board.

CCD shift register memory is currently available in pilot quantities in the 64K-bits size. Another factor of four in storage size (256K-bits) is expected by 1980.

Manufacturers reported the occurrence of spontaneous errors in CCD memories. This leads to a requirement for continuously monitoring the contents of a CCD memory and rewriting it correctly when bit errors occur.

Present bubble memories put severe complexities into the controlling and driving circuitry, making them very difficult to use.

Sufficient information about the magnetic storages available for the archive was obtained to indicate that there are several commercially available contenders for the archive storage. No effort was made to determine which of today's contenders were likely to be withdrawn from the market in the next two years, nor to uncover the new contenders which are undoubtedly under development.

PROCESSOR - FLOW MODEL MATCHING STUDY OVERVIEW

The key sub-study in this effort was the Matching Study. Certainly, it had the most profound effect on the evolution of SAM as the chosen processor architecture as well as some design details. This sub-study was broken into several tasks prior to the actual matching or evolving process itself.

- Cataloging and examination of pertinent generic architectures for consideration to be used as a starting point.
- Establishment and discussion with NASA-Ames of critical issues and basic requirements and capabilities imposed on the architecture by the problem definition.
- Research and discussion of the fundamental characteristics of the flow models which affect the processor architecture.

Following these tasks, the results were merged with those of the other two sub-studies the total implications of which determined the final architecture.

Generic Architectures considered as starting points were:

- Hybrid system composed of analog computation devices with digital control and storage
- Parallel array architectures with replicated arithmetic units executing the same program on different data achieving performance as a multiple of the number of arithmetic units.
  - Type 1 - Lock-Step synchronous arrays with clock-by-clock tight coupling of arithmetic units
  - Type 2 - Non Lock-Step arrays with coupling at predetermined synchronization points rather than every clock
- Pipeline architectures where operations are streamed through different stages with performance as a multiple of the number of states.

A complete discussion of these generic architectures is found in Appendix L of the final report.

Critical issues, basic requirements and capabilities were jointly developed between the study team and NASA Ames personnel. Topics examined were:

- Navier-Stokes Solver Capabilities
- Programming
- NSS - I/O

NSS CAPABILITIES

The ability to solve the three-dimensional Reynolds averaged Navier-Stokes equations, using both explicit/implicit and totally implicit, dimensionally-split, finite-difference methods.

The ability to compute, at high efficiency, problems containing a variety of boundary conditions which include the independent variables, their derivatives, and other auxiliary variables, a variety of internal and external geometries and a variety of turbulence models ranging from algebraic to seven differential equation descriptions.

The ability to compute solutions for up to one million grid points. This implies a data base range to 14 million words for:

5 conservation variables at 2 time levels

1 turbulence variable

to 40 million words for:

5 conservation variables at 2 time levels.
7 turbulence variables at 2 time levels.
3 grid coordinates
12 metrics (including time)
1 Jacobian

The ability to obtain steady state solutions for one million grid points in 10 minutes of CPU time for 3-D problems using algebraic turbulence models. At present this must be measured using 2-D explicit/implicit and implicit codes as performance metrics.

Two examples of typical programs and their computational requirements are given below:
Explicit code (MacCormack) status: A 2-D airfoil steady-state solution was obtained in 7 minutes on CDC 7600 for 2100 grid points. The steady-state was reached after 13 chord lengths of travel by computing inviscid solution for 7 chords and viscous solution for remaining 6 chords. Effective computing speed on 7600 is about 2 MFLOPS. Assuming twice the computational effort at each grid point for the 3-D case, this implies that to compute 13 chords in 10 minutes for one million grid points requires an effective computing speed of 1.4 gigaflops. Greater efficiencies by 1980 can be expected.

Implicit (Lomax, Steger) code status: A 2-D airfoil steady-state (12 chords traveled) was obtained in 10 minutes on CDC 7600 for 2300 grid points - all calculations were viscous. The effective computing speed on 7600 is about 2 megaflops. This code implies that an effective computing speed of 2 gigaflops will be needed for a 3-D calculation over one million grid points. However, researchers working on the implicit code are confident that improvements in the treatment of boundary conditions and other strategies can improve the speed of the method by a factor of 2 which implies that at least one-gigaflop effective rate will be needed.

It is concluded that the minimum effective computing rate needed for the Navier-Stokes problem is one gigaflop.

A precision of 10 decimal digits is required.

PROGRAMMING
A high level programming language consistent with ease of mapping the solution methods onto the machine, optimum machine performance and the available language development time is necessary.

Desirable programmability features of the Navier-Stokes machine are as follows:
A FORTRAN-like high level language with extensions necessary for efficient problem mapping. As well as the following features.

- a stable optimizing compiler
- good compiler diagnostics
- warning from the compiler of possible run-time inefficiencies
- ability to give good run-time diagnostics and statistics
- vector length independence
- freedom from the need to do explicit-mode vector manipulation
- ease in specifying data allocation.

NSS I/O

The primary I/O activities of the machine are the input of initial problem parameters, restart from stored data, and the output of snapshots and restart dumps. Another important activity is the output of debug dumps. Two basic types of Navier-Stokes solutions are desired—steady and unsteady (or more correctly quasi-steady). Steady cases are characterized by the appearance of a solution that does not vary with time after some large number of time steps or large number of characteristic body lengths travelled. Unsteady cases are characterized by the appearance of a solution that is periodic in time after some large number of time steps. In order to analyze the unsteady or periodic nature of these solutions more time steps (on the order of six times that of steady cases) are required. Additional data output is also required in these cases. It is estimated that 75% of the time will be used to solve the steady flow case and the remaining 25% the unsteady.

The following output capabilities for these cases are desired.

- Snap Shots
  a. Integrated quantities such as drag, lift and moments approximately every 15-30 seconds.
  b. Surface quantities such as pressure and skin friction. If the grid moves with time, the grid coordinates must also be output. A given quantity such as pressure, plus the coordinates could total up to approximately 60,000 words of output every 15-30 seconds.
  c. Flow quantities in the field such as pressure or Mach number. For a grid of 1,000,000 points an entire field of, say, Mach numbers plus coordinates would be 4,000,000 words. However, it is anticipated that only selected grid points need to be output and this would be about a hundredth of the above or 40,000 words every 30 seconds. These snapshots require the heaviest output and for 60 minute runs would accumulate up to 5,000,000 words for the unsteady cases.
- Restart Dumps
- Debug Dumps
- Formatted I/O

FLOW MODEL CODE CHARACTERIZATION AND ANALYSIS

Codes, supplied by NASA Ames were analyzed statically and dynamically to determine what the specific characteristics of the Flow Model problems are and how do they impact computer architecture. The codes studied were written for two specific computers. Features in each code that were specific to its target machine were stripped away to find the basic issues. The areas that were examined group themselves naturally into those issues which address processor requirements, memory requirements, or communications requirements, and are outlined below.

Memory Requirements

- Data Base Size - (The actual input/output variables)
- Program Size
- Workspace Size (Those variables never outputted in normal production code - the temporaries)
- Access Patterns (dimensionality of problems, subarray structure, indexing patterns)

Communications between Processors & Memories

- Number of Computations per Data Base Access
- Interaction of Problem Variables
- Data Dependency
- Control Structures
- Access Patterns (planes, rows, columns, etc.)

Processor Requirements

- Word Size and Format
- Relative frequency of operations
- Index computations
- Number of input operands per output operands
- Scalar operations
- Frequency of intrinsics
- Program structure

Each of these issues were examined in detail and the results are listed in Chapter 8 of the final report with a full discussion of the methodology.

The study of the memory requirements showed that the canonical problem variables and number of grid points produce a data base memory of 14-40 million words (NASA-Ames requirement). The workspace size was found to be approximately 40 temporaries per database variable. This of course is programmer and architecture dependent and hence is only an indication of the relationship between work space and data base. It was found that the problem arrays are generally 4 dimensional with 3 geometric and 1 variable coordinator. They are accessed in a fairly regular manner in the sense that the indexing is a function of the loop variables plus or minus a small integer. There is almost no indexing that occurs as a function of loop variable and another integer variable set outside of the loop. The structure of the loops indicate that entire arrays are processed in a given piece of the computation rather than small subarrays. Program size is relatively small at under 4000 card images.

Requirements on communication between processor and memory structure were determined by a number of flow-model...program parameters. The data dependency studies of variables in loops showed that there existed complex first order linear recurrences which were functions of each of the three geometric variables. These recurrences occurred in over 60% of the executing Implicit program. The study of the control or branching structures within the programs showed them to be relatively simple and generally linked to loop variables. Some were data dependent but when they occurred the variables were functions of inner loop parameters.

Further studies of the relationship between the data base memory requirements, the work space requirements and the number of floating point operations showed that a fetch or store to data base memory occurred infrequently in comparison to the number of floating point operations. Typically the Implicit (Steger) program has an average incidence of 15 floating point operations per fetch.

Additionally, by investigation of the indexing patterns within loop structures one found that there is relatively low interaction among problem variables on different grid points. For example, variables are fetched from several adjacent points, computations are performed and then a result is stored relative to the grid point. There is no continual switching back and forth of index patterns. The access patterns appear to be simple rows, columns and planes with a skip distance of 1.

Processer requirement studies showed that multiply, add, and multiply-add instructions are extremely important floating point operations. For example, in the Implicit Code it was found that 53% of all operations were multiplies, 44% were adds and 2.5% were divides. About 60% of all operations occurred as multiply-add pairs. Division and intrinsics as SQRT and EXP occur rarely and double precision is never required. Since most of the array references are to 3- and 4- dimensional arrays integer arithmetic calculations are a strong requirement. The combination of work space requirements and the average number of input operands to output operands (3.5) places certain requirements on the processor. NASA-Ames has additionally specified 10 digit accuracy requirement.

The data collected from the studies were used to define and delimit the characteristics of the requisite architecture. The output from the matching study together with the technology study and facilities study data were then used to develop definitions of an architecture discussed after the results of the facility study.

FACILITY STUDY

The primary objectives of this sub-study were threefold:

- Identify housing and support requirements of the facility
- To provide cost and schedule engineering estimates for effective planning
- Assessment of NSS implementation issues as they would impact architecture and design choices.

These objectives were pursued by determining the facility requirements of those units or sub-systems already identified and placing reasonable bounds on facility requirements for those elements which have yet to be specified. After a preliminary definition of the NSS, an implementation schedule and an engineering cost estimate were assembled, and analyzed. As the NSS definition proceeded, additional iterations on the schedule and cost were performed.

Finally, the critical issues relevant to implementing the NSS were defined and guidelines developed to insure that the design would indeed be realizable. This effort raised some interesting considerations which impacted the architecture choice and some design details as well.

Critical issues affecting the implementation or realization of the NSS in particular are:

CRITICAL PATH ANALYSIS was examined to eliminate short waterfalls in the schedule by locating their source and minimizing their occurrence.

PROCUREMENT problems can be avoided if there is an early identification of long-lead items, if custom componentry is minimized, if multiple sources are employed wherever possible, and if adequate protective documentation is obtained from each vendor. This issue can be the largest single risk factor in any program's schedule, cost, and possibly performance.

PRODUCTION considerations include maximizing the number of replicated units to minimize production learning curves and take advantage of economies of scale. Standardization of componentry, connectors, cables, etc., minimizes inventory problems and smoothes the production process.

MODULE OR SUBSYSTEM INTERFACE MANAGEMENT demands the reduction of complexity of interconnections between all functional elements.

DEBUGGING AND MAINTENANCE: As in the production considerations, if the number of complex elements, which field engineers must work with, are kept to a minimum, then debugging and maintenance are simplified -- furthermore, this minimizes the inventory of spares.

PACKAGING of any design must have the highest density consistent with heat removal. It must be such that the LRU (lowest replaceable unit) is easy to isolate, test and replace. Additionally, usage of common board types should be maximized.

LOGIC DESIGN RULES AND NOISE BUDGETS. A technology choice for the design must be mature enough to develop credible noise budgets, and provide adequate operational margins.

POWER. Finally, power considerations suggest that we avoid complex power distribution schemes, and concurrently maximize the distribution of heat dissipation. These considerations will lead to some interesting features explained in the next section.

ARCHITECTURE EVOLUTION

The selection of the Synchronizable Array Machine for the NSS is presented as an evolution of concepts that grew out of the findings of the three sub-studies.

The first step in this evolution was the selection of a parallel architecture after examination of three generic types: hybrid, pipeline, and parallel. The hybrid was rejected for three reasons.

DIFFICULTY OF PROGRAMMING. Many difficulties make it impossible to translate the current Navier-Stokes algorithms to a hybrid machine. Years have already been spent in algorithm research in digital form. Even more investigation would be needed to recast the equations into suitable form for analog computation.

INACCURACIES, AND UNPREDICTABILITY OF THE INACCURACY. Such limited accuracy as exists in analog computation is often data dependent, and changes with age. In digital computation, any desired degree of accuracy can be specified.

COMPONENT FAILURES. Unlike a digital computation, where tests can continuously ensure that correct results are being produced, an analog computer has no error control. A faulty component or off-scale input produces an output voltage which is not distinguishable in kind from the output voltage of a properly functioning component.

Although analog processors have a very high computation rate, these limitations are totally unacceptable for the objectives of an NASF project.

Pipeline architectures as we know them today appear to suffer from inefficiencies, namely:

- Long start up times between vector operations,
- Difficulty in dealing with transpositions, and
- The need for massive amounts of work space memory to accommodate propagation of temporary variables.

Certainly these problems can be dealt with and solutions developed to make a pipeline a suitable architecture (as we have done for the parallel architecture) but a reexamination of the Facilities Study highlighted other issues which made the selection of a parallel array more sensible for Burroughs.

Assuming both architectures could be evolved to produce a design of equal performance, Burroughs is more confident that the parallel machine can be manufactured with less risk. The claim is based on observations:

- The large number of replicated units in a parallel array minimizes production and debugging and field engineering learning curves. Certain economics of scale could be realized in development as well.

- Burroughs experience, in three generations of parallel high performance systems (namely ILLIAC, PEPE, and the Burroughs Scientific Processor (BSP)), provides an invaluable data base of knowledge in the detailed design and manufacture of such a system.

The beginning of the architectural development, therefore, was based on the generic parallel configuration shown in Figure 4.

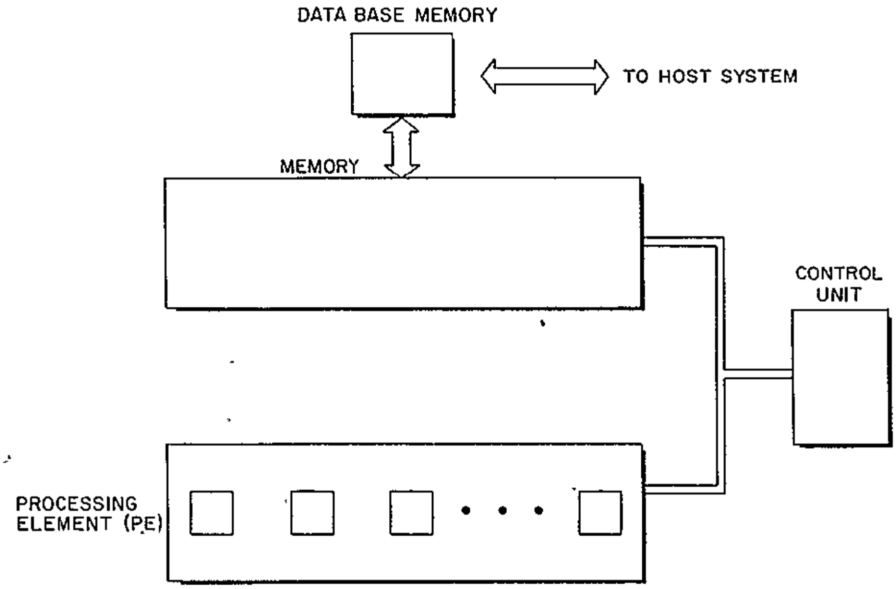


Figure 4 Parallel Configuration

From this point, the definition of SAM can be well understood as a series of refinements based on results of the sub-studies.

The ADI method of solution of the aerodynamic equations, with split operators, demands that many data arrays be transposed during access. The access patterns of this method require that 2-dimensional planes of the 3-dimensional grid be accessed in parallel. Planes are required from any 2 of 3 dimensions in the same grid. This implies the need for an efficient transposition mechanism.

Several different designs were considered. The selected Transposition Network (TN) is a unique innovation offering:

- low parts count
- minimal data access delay
- simple control requirements
- simple but flexible data allocation.

This design demands that memory be partitioned into a prime number of banks larger than the number of processors.

The Transposition Network (TN) is shown in Figure 5 as the first refinement of the generic parallel configuration.

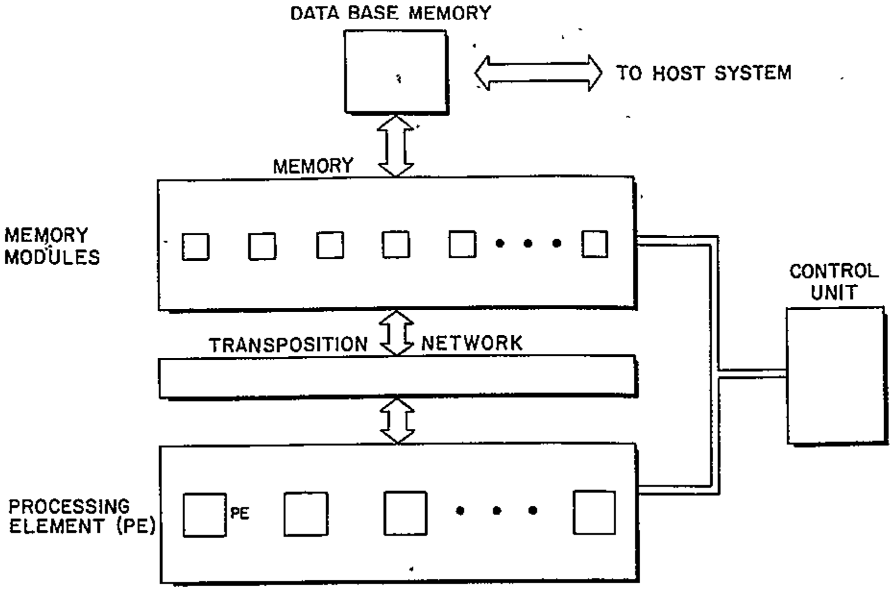


Figure 5 Parallel Configuration - Refinement 1

The occurrence of a significant number of floating point operations to fetches (especially in the implicit code) implies a large workspace requirement. In fact up to 40 temporary variables per data base variable may be generated. Propagation of such a large number of temporaries throughout the machine would cause severe timing penalties. To mitigate this problem, local memories for each processor are required. In addition, the bandwidth of the TN can then be reduced without performance degradation. This makes the Transposition Network simpler and less costly. The absence of data dependencies among points in the same plane allows this refinement (Figure 6) to occur. The increased cost of many data memories in the processor is offset by the decreased requirement for storage capacity in Extended Memory for temporary variables. The nomenclature for the main memory can now be appreciated as Extended Memory (EM).

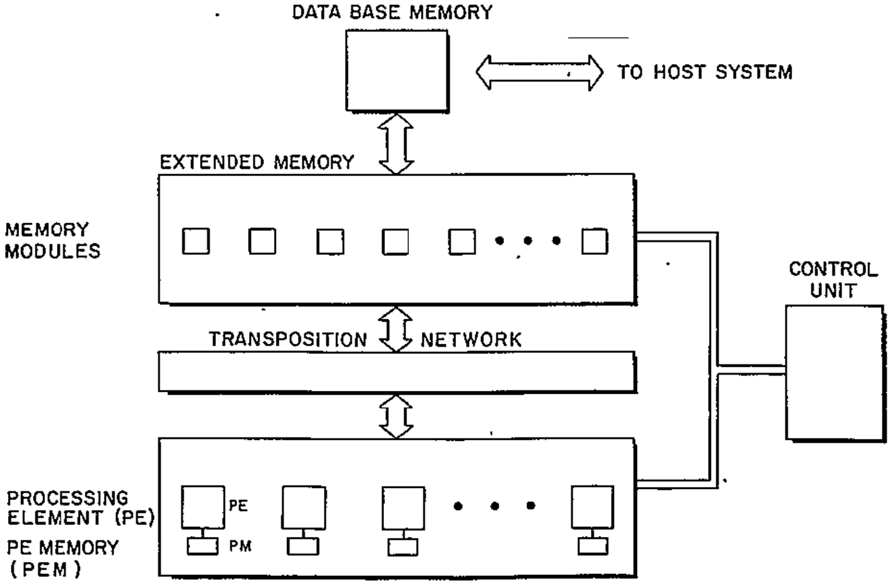


Figure 6 Parallel Configuration - Refinement 2

The result of this refinement allows one to think about parallelism as a series of vertical slices. That is:

Given a series of statements of the following form:

```
DOPARALLEL (one or more indices, say I, J, K, between limits)
   STATEMENT  1 -- involving variables indexed on the parallel indices
   STATEMENT  2 -- involving variables indexed on the parallel indices
        .
        .
        .
   STATEMENT  n -- involving variables indexed on the parallel indices
ENDDO
```

there are two ways of thinking of the parallelism.

In the first method, statement 1 is executed on the vectors implied by the parallel indices. Then statement 2 is executed as a vector statement, and so on up to the nth statement. Having each statement executed separately as a vector statement is called "horizontal slicing" of the parallelism.

The second method is to assign a processor to a particular instance of the set of indices. Processor 17, for example, may handle all computation associated with J=1 and K=19, while processor no. 222 handles J=3 and K=22. Each processor now executes, essentially independently, a piece of code involving the I index. This kind of parallelism has been called "vertical slicing." Vertical slicing is appropriate when, as in the Navier-Stokes equations, there is little interaction between the variables at one grid point and the variables at another.

Three or more generations of parallel processors have shown that instruction interpretation of parallel constructs by the CU creates a bottleneck. The CU must be extremely fast to keep up with the array. Its complexity is severe enough without this responsibility. The program size has been observed to be small enough to consider placing program memories in each processor as shown in Figure 7. This now results in a stand alone processor with manageable interface (very few lines) to the control unit, elimination of massive cabling and a simpler CU. These savings and their attendant design and schedule issues will offset the cost of multiple copies of the program memory, as well as improve performance.

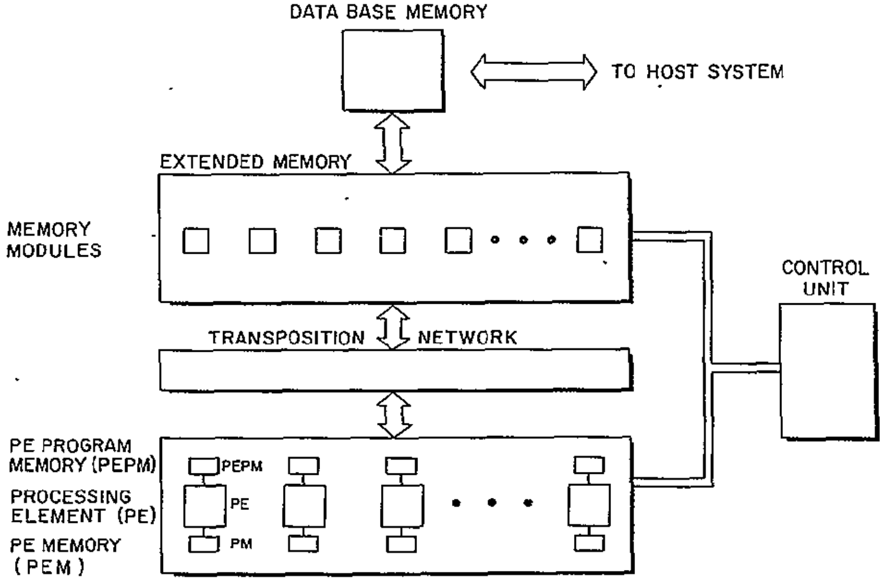


Figure 7   Parallel Configuration - Refinement 3

In a parallel array with a single program memory, the distribution of instructions by the CU serves to synchronize the operation of the PE's. Distribution of the program to local program memories results in a requirement for a synchronization mechanism between CU and PE's. To provide maximum flexibility, we elected to invoke the synch mechanism explicitly in the code stream (Figure 8). This allows synchronization to occur only when necessary, (i.e., just prior to parallel fetches and stores). Processors can run concurrently without waiting for each other, which permits data dependent instruction options (e.g. round after normalize if overflow) to be executed only when needed. The independence allows idle processors to execute confidence checks on themselves. Different code sequences for different areas of the airspace may be executed in different processors.

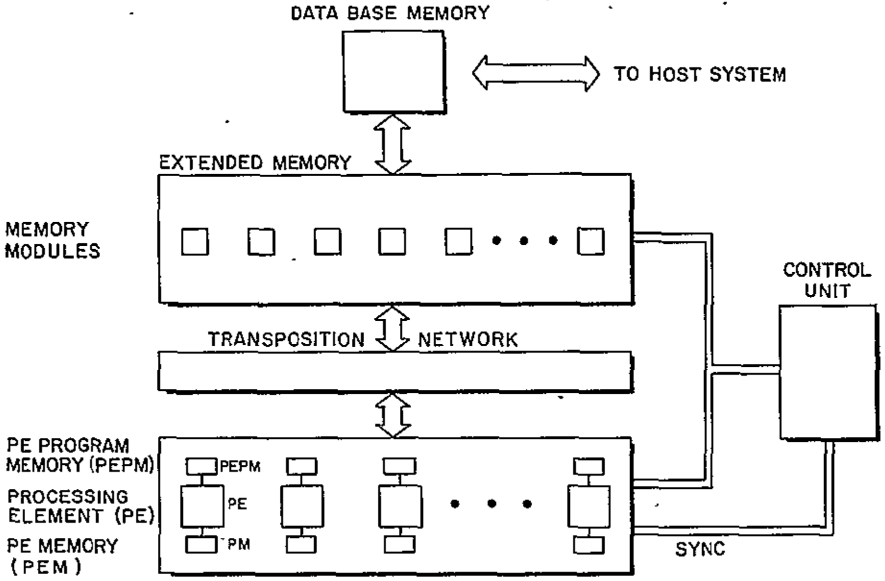


Figure 8 Parallel Configuration - Refinement 4

The choice of 512 as the number of processors is based primarily on the highest expected speed of efficient memory chips. 16k-bit static RAM chips are expected to be available at about 100 ns cycle time, by 1980, and are appropriate to processor and control unit memories. 64k-bit dynamic RAM chips are expected to be available at about the same time, at speeds nearly matching the present 200 ns or so speed of current 16k dynamic RAMs. These are the memory chips in the baseline system.

Consider, for example, the effect on the design of a choice of 256 processors. The twice-as-fast processor memories would require 50 ns chips, which would be available only in a 4k-bit size. Thus, the total number of memory chips would double, from the 37,888 memory chips of the baseline system to a total of 75,776 chips. The twice as fast EM would require 16k-bit chips to maintain the same speed, and its size would quadruple from 29,176 memory chips to 116,704 chips. Parts count in the twice-as-fast processor is estimated to double, making no net savings, but increasing the required design effort.

The size of data base for codes expected to execute for 10 minutes indicates as much as 10^{17} bits of data are operated upon. To expect no failures in that time is ambitious indeed, therefore it was necessary to impose a strict philosophy of fault detection and correction in the design of the hardware and software, including:

- Hardware Error Detection
- Hardware Error Correction
- Arithmetic Checking

Figure 9 is a block diagram of SAM, the Baseline Design for the NSS. Its evolution, as well as subsequent design decisions and guidelines results in a design which features the items described below.

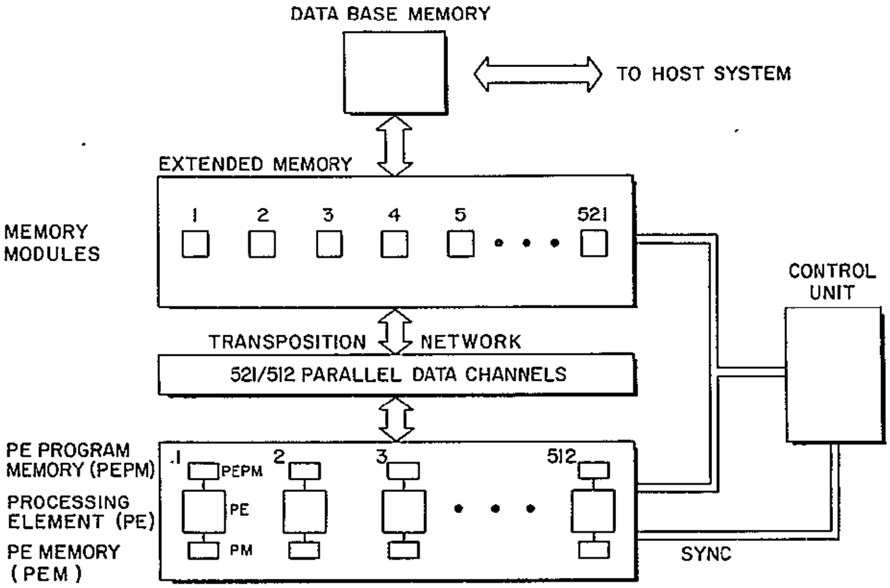


Figure 9 Parallel Configuration - Refinement 5

HIGH THROUGHPUT

The throughput potential of the NSS is 1.7 billion Floating Point Operations per second. This is derived from the selective ratios of the instruction mix combined with the expected execution time of the operations. This yields 294 nsec per 512 floating point operations which is equivalent to 1.7 billion FLOPS. Additional study of the baseline for the specific codes indicates that the required effective rate of 1 billion FLOPS is achievable.

EASE OF USE

High level language requirements, the guidelines of matching machine code to the user language and indeed the use of a High Level Language to write the compiler were inportant decisions made early in the study. The Vertical Slice Concept allows all classical serial optimization techniques to be utilized on the SAM. Recognizing that this architecture has unprecedented flexibility, it is incumbent upon the compiler to have debug aids to protect the user.

The protected environment in which SAM operates – the high speed computational envelope isolated from the rest of the system – requires that it have only a very small operating system of its own. I/O to and from that envelope will not encumber the user or SAM as well. A typical work flow is illustrated in Figure 10.

This architecture is a tradeoff optimized for the aerodynamic problem, yielding lesser performance for:

- Problems with intimate arithmetic data dependency from one grid point variable to another,
- Interactive environments, and
- Multi-programming environments.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

We feel this represents a unique solution to the problem of numerical aerodynamic simulation, and Burroughs presents this design with full confidence in its feasibility. We believe that this system is the best approach to meeting the NASF goals in a timely and cost-effective manner, maintaining NASA's position in the forefront of scientific endeavor.

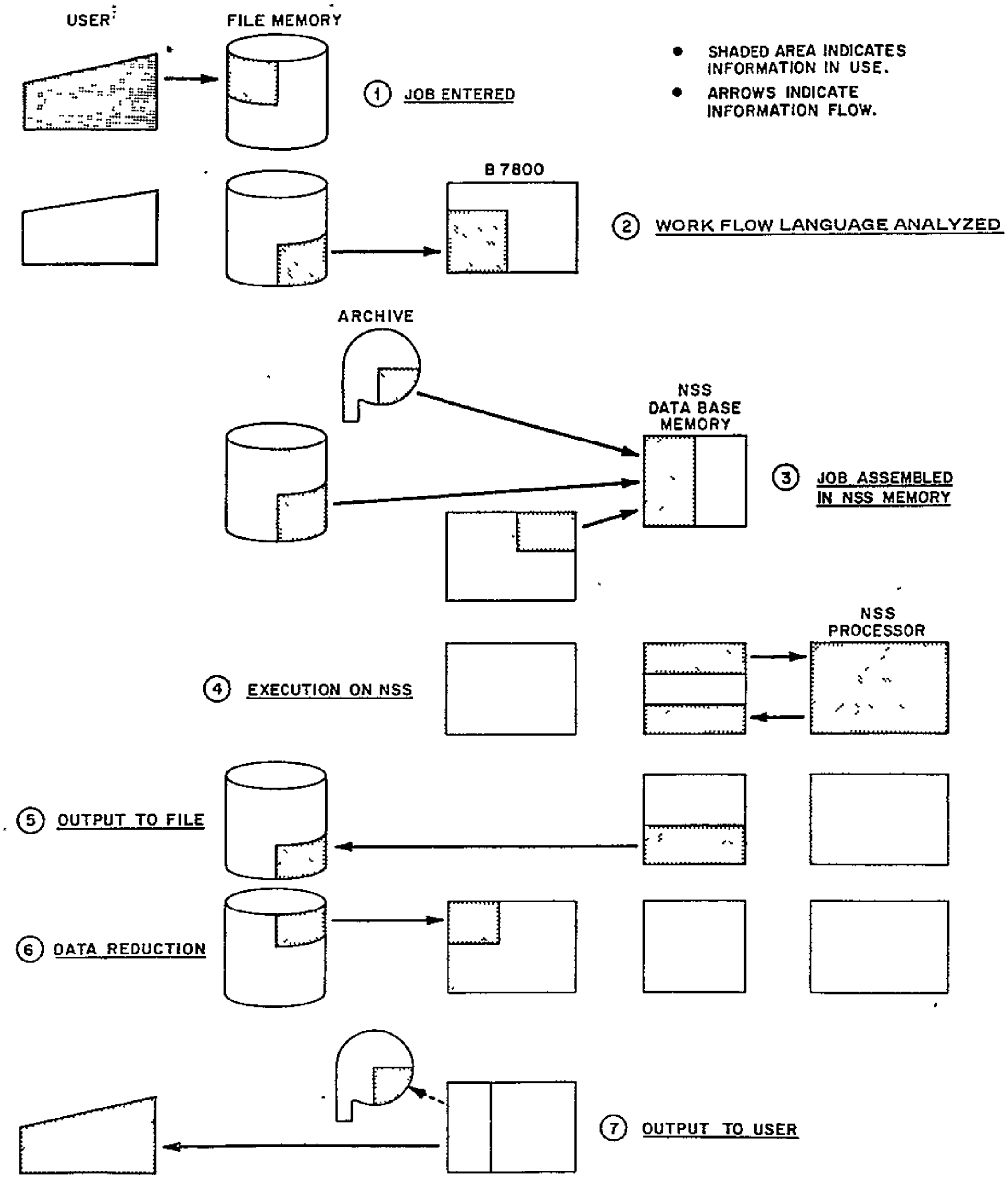


Figure 10 Typical Work - Flow Schematic

D5

N78-19783

PRELIMINARY STUDY
FOR A
NUMERICAL AERODYNAMIC SIMULATION FACILITY

SUMMARY REPORT

BY: N. R. Lincoln

OCTOBER, 1977

Distribution of this report is provided in the interest of information exchange. Responsibility for the contents resides in the authors or organization that prepared it.

Prepared under Contract No. NAS2-9457 by:

CONTROL DATA CORPORATION
2800 Old Shakopee Road
Minneapolis, Minnesota 55440

for

AMES RESEARCH CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NUMERICAL AERODYNAMIC SIMULATION FACILITY
SUMMARY REPORT

For the past 6 months the Research and Advanced Design Laboratory of Control Data Corporation has been conducting a joint study in cooperation with Ames Research Center of the National Aeronautics and Space Administration (NASA). The objective of this study was to determine the methodology and feasibility of construction of a Numerical Aerodynamic Simulation Facility (NASF). This facility would be utilized by NASA as an integral component of a complete service to the aerodynamic design and evaluation community represented by industry and government engineering organizations alike. These services would include the open availability of the NASF, physical wind tunnels of all sizes, and the vast expertise possessed by NASA engineers, physicists, and mathematicians.

The study began with several assumptions. First, no existing computational ensemble could provide the necessary solutions to three-dimensional Navier-Stokes equation systems representing aerodynamic shapes in all speeds of airflow. The second assumption was that such a facility would find its most critical needs arising about 1982. This date was itself a compromise between the desire for a high performance computational capability to meet immediate needs and the known state of the computer art in 1977 which is not capable of meeting even the most modest objectives set for the NASF. The third assumption was that no more than two computational approaches would be viable for the NASF, and that work at Ames in development of the program was sufficiently mature to permit actual codes to be used in the study.

The Control Data approach to the study was then to make a quick, early assessment of the probability of achieving computational performances in excess of 100 times the CDC 7600 speeds being realized by the existing Ames installation. At the outset it was felt that with technologies already in hand and architectural principles already demonstrated, achieving the performance goals by 1982 was a certainty. At the direction of Ames personnel, however, Control Data proceeded to examine the state-of-the-art of relevant technologies, the state-of-the-art of systems and processor architectures, and the measurable computational requirements of the two Navier-Stokes solution programs then in existence. The purpose of this phase of the study was to provide NASA with sufficient information so that its staff members could make an independent evaluation of the best approach for construction of the facility. At the same time Control Data would attempt to develop a system design to meet the objectives.

The general technical approach to the system design was to use, wherever possible in the design, standard parts and components to reduce development costs and risks for those components. This resulted in the identification of two main components in the NASF, the front-end or support processing system, composed of commercially available equipment and software, and the back-end or Navier-Stokes Solver (NSS), which must utilize special design, special technology, and special software to meet the speed requirements of the facility. Initially, it was felt that a derivative of the STAR-100 architecture and design could be used for the NSS. This would further reduce the development costs and project risks, as well as manufacturing costs due to volume ordering of common components. Since a member of the STAR family, the 100C, appeared to possess a basic computational speed on which to build a specialized processor, the concept of commonality appeared quite appealing.

About two thirds of the way through this study effort, however, it was found that some radical departures from STAR architecture and design had to be taken to meet the goals of the NASF. It did appear, however, that certain of the technological achievements in LSI technology and system organization of subcomponents could be borrowed from the STAR-100C project to reduce design time and risk of completion of the NASF.

SIGNIFICANT RESULTS OF STUDY

Given this initial orientation, the study yielded significant results that are summarized below.

TECHNOLOGY

- The basic memory unit for an NSS is still best constructed of bipolar memory parts of the emitter coupled logic (ECL) family or a family with similar speeds. For a system of this generation, memory access speeds in the 30-to 40-nanosecond range for up to 8 million words of data are attainable.

  A lower range of memory speeds is available with current technology, with attendant cost and power reductions over the high performance ECL memory. To meet the needs of the three-dimensional Navier-Stokes codes, there are a known number of occasions when memory must be accessed in an unstructured, random manner. To reduce the delays accompanying sequences of random accesses, the memory can be built into a multitude of banks such that the probability of two successive references can be almost eliminated. There are times, however, when all computation must pause while a required operand is retrieved from the memory system. In such cases, the access time delay for a single operand becomes important. Thus, to ensure that no facet of the Navier-Stokes solution becomes a bottleneck, the memory must exhibit the combination of properties of high bandwidth, fast access, and multiple banking. It is felt that the fastest, reliable technology available today is the correct choice for memory technology.

- The basic logic element for a processor of this type will be based on high-speed, large scale integration (LSI) devices with switching speeds in the 500-picosecond range. Exotic elements such as Gallium Arsenide and Josephson devices have not progressed sufficiently in initial research to be used in a manufacturing environment in 1980 to 1982.

  Studies of various technology families and architectural alternatives have revealed that it is more cost-effective and more reliable to build a superprocessor from a minimum number of parallel units implemented with the fastest technology available than to attempt to meet the same level of performance with a large number of parallel, but individually slower speed processors. The NSS should therefore be constructed of the best technology available in the 1977 to 1982 time frame. Of course, the performance, manufacturing, and cost advantages of LSI dictate the use of the highest integration possible. For ECL speeds, the number of gates possible today per LSI component is between 150 and 200. Expectations for an LSI component with 400 to 500 gates to be available for construction of the NSS are reasonable, though not without some risk.

Lower speed components of the MECL variety will necessarily be employed where circuit speeds are not as important as power dissipation, cooling, and cost. For example, the I/O system, trunks, and some peripheral subsystems will be constructed from existing technologies, both MOS and lower-powered ECL. If at all possible, all components should be built with industry standard parts, even the LSI portions. Membership in a larger family ensures some long-term longevity for spare parts and support from a variety of semiconductor vendors.

- Slower speed memories will be fabricated with charge coupled devices (CCD) for NSS applications, since the state of development of electron-beam memories (EBAM) and magnetic bubble memories cannot yield components of the desired bandwidth or reliability.

Million-word (64 bits) systems of CCD memories are being built to practical specifications today with 65K circuits. There is a realistic chance that operational CCD parts containing 265 kbits will be available for prototype system implementations in 1978. If the analysis of the NSS memory requirements is sustained by later studies, a 256-million-word system will be needed by 1982. Within the limitations of packaging, cooling, and reliability, it therefore appears quite practical to anticipate a 256-million-word system to be available for an operational NASF in 1981 to 1982. The programmatic study of the specimen flow model codes shows that a brute-force swapping technique can be employed between the main memory and the auxiliary storage medium. If this technique greatly simplifies hardware and software control, it must also possess data bandwidths of at least 1.6 billion bits per second (each way) to achieve the sustained processing rates desired for the NSS.

Although million-bit bubble memories are now available for prototype experimentation, the bandwidths of such chips are limited to the 400 to 500 kilohertz range. In addition, the access time for data blocks is quite a bit higher than for the corresponding CCD technology. The ability of bubble memories to retain data in the event of power failures is desirable, but if the total run time for which data must be retained is less than 20 minutes, the loss of bandwidth and access time is not worth the cost. For example, existing million-bit chips would have to be arranged in parallel, with 4000 chips simultaneously transferring data, to achieve the 1.6-gigahertz data rate. Bubble memories of smaller size will most likely be found in some of the peripheral subsystems as replacements for small disks and fast-access drums that now hold directories and store-and-forward message buffers.

- Rotating magnetic media will remain the primary form of mass storage and archival storage for a system built in the early 1980's.

Extensions of existing knowledge and technologies involved in rotating magnetic memory are readily projected for the next 4 years. There remain only the solutions to several nagging engineering questions before another improvement in density and transfer rates can be seen. The most probable direction will be in the form of sealed (almost hermetic) units containing disks, positioners, and head groups. These units will employ plated disk (rather than oxide-coated disks) to reduce film thickness and thus improve resolution. Factors of 4 to 16 times the existing storage densities will be achieved in the NSS timeframe. As an example, an 819-size unit (one single disk unit) will be able to house from 40 to 50 billion bits.

Laser and photostorage devices are not yet in the same ballpark with rotating mass storage for reliability and system availability. In the case of the NASF, the predicted on-line storage requirements can be met with the next foreseeable generation of disk storage devices.

Archival storage is an area still undergoing great upheaval and experimentation. Although the IBM and CDC mass storage subsystems represent today an imperfect engineering approach to archiving, offshoots of them will probably still engage magnetic tape technology and random selection systems being pioneered by them. For this reason, site requirements were based on existing units such as the 38500 mass archival storage.

PROBLEM ANALYSIS

A large portion of time was spent in the analysis of two-dimensional specimen codes provided by NASA/Ames personnel. These were the explicit code being evolved by Bob MacCormack, and the implicit code under development by Steger, Pulliam and Lomax. Both codes were first run in their original FORTRAN form on the STAR-100 where the STAR instrumentation could be used to sample the key elements of the code operation. Both codes were then vectorized for the STAR-100 as a first step in the process of developing parallel algorithms to match the NSS, and as guidance for the creation of a unique NSS processor.

Finally, as the NSS structure took shape, the implicit code was restructured to match the new architecture and a set of rough estimates made as to the behavior of that code on the proposed NSS.

A summary of some of the results of this phase follows:

1. The explicit code required 7 minutes of 7600 time to compute a particular solution for the Garabedian-Korn airfoil to 256 time steps. The original scalar version of this code with no vectorization or optimization required 16 minutes of STAR-100 time reflecting the state of the compiler development, as well as the 80-nanosecond scalar issue rate of the STAR-100. A partially vectorized version of this code (one of the split operators) was run at 4.5 minutes. A fully vectorized version was not completed due to the diversion of attention to the implicit code. The explicit code was operating at an average rate of two megaflops for the total run on the 7600.

2. The implicit code, processing basically the same problem as the explicit code, was timed at about 12 minutes on the 7600 and 35 minutes in scalar FORTRAN on the STAR-100, while a first attempt at vectorization for the STAR-100 yielded a five-minute running time. The implicit code does not rely on special casing of computational regions and thus performs many more floating-point computations than does the explicit form. The implicit code operated at an average of around two megaflops on the 7600 also. The code developers are convinced that the three-dimensional form of this implicit program can be refined to reduce the computational requirements. This programming ploy is essential to the NASF meeting its system goals.

3. The implicit code was then singled out for restructuring for a hypothetical NSS. A method of processing slices of the data, similar to the scheme used by Lomax on the ILLIAC IV, was devised to permit a reduction in the size of the costly, high-performance main memory. A system of small, high-performance buffers, backed up by 8 million words of main memory, and that backed up by 256 million words of block transfer memory, can be effectively utilized by the slice mechanism. Depending on slice lengths the restructured implicit code was estimated to perform on the NSS between 660 and 940 megaflops in 64-bit mode and from 950 to 1910 megaflops in 32-bit mode.

4. A three-dimensional form of the implicit code can be sliced more efficiently and, by using 32-bit computation mode for a majority of calculations where accuracy permits, it is estimated that the NSS should run at an average rate in excess of 3000 megaflops, assuming a main computer clock of 10 nanoseconds.

SYSTEM

Figure S-1 gives a block diagram overview of the NASF system envisioned.

It can be seen from this figure that the NSS processor represents only a small portion of the equipment volume, as well as only about one-third the total system cost. The mass storage equipment and graphics subsystem needed to support the NASF are shown in rough outline form only, but represent the projected needs of an installation that will be operational in the 1982-1983 timeframe.

Some salient features of the displayed system are:

- A dual processor front-end configuration composed of computing equipment available in 1977 would provide sufficient power and reliability to meet the demands of a front-end system for the NASF. Computing equipment currently under development for standard sales in the 1980's promises even higher performance and reliability along with reduced cost, thus ensuring that the computational facility will have substantial power in the supporting subsystems.

  Experience with the STAR-100 system has shown that the development of even a minimal operating system to meet today's normal needs for system access and features is a monumental undertaking. From a manufacturer's point of view, when P and L statements become persuasive inhibitions to grandiose plans, some means of reducing cost and schedules for putting a new computer architecture into production are absolutely essential. The computational facility concept was thus defined, wherein the STAR processor performed primarily calculations, and CDC CYBER processors performed all the data management functions, file and user security, access functions, and communications management functions necessary for a production system. This substantially reduced the resource and time requirements for STAR software. Further, it meant that a more stable operating system was available earlier in the production cycle.

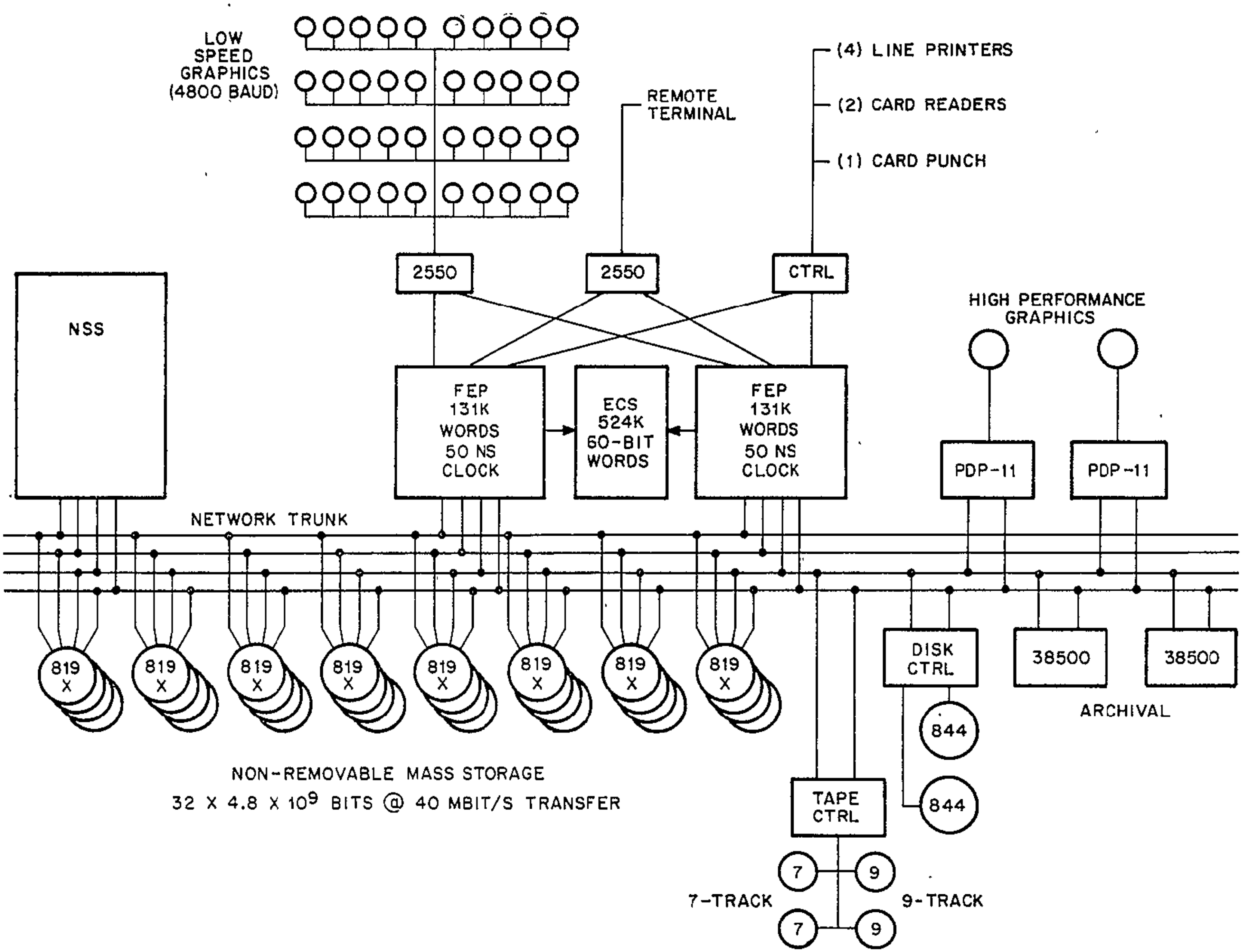


Figure S-1. NASF System Interconnection

By choosing a mature computer system for the front-end function, fully supported with the entire range of software available, NASA can be assured that the continuation of high levels of effort on performance, feature and stability aspects will yield a better system in 1982 than one designed specifically for Ames.

- A network trunk scheme of system interconnect would provide a more flexible means of harnessing all the equipment needed in the NASF. The distances which can be achieved, the number of connections to one trunk, and the sustainable bandwidths make this system quite appealing to meet the system requirements of the NASF.

  Network trunks with 50 million bits/second transmission capability and cable lengths of approximately 600 meters (2000 feet) are now operational. In addition to allowing peripheral devices and peripheral subsystems to be more remote from the attached computer, the trunk scheme is specifically designed to mate with alien equipment. This becomes a plus for users, such as NASA, permitting them to make the best choice of equipments to be attached (with the appropriate, moderate-cost adapter) to the trunk without concern for matching electronic channel and software protocol requirements.

  Such a network system allows the user to determine whether data can be transferred from one disk storage system to any attached processor without having to pass through a front-end machine. This can reduce bottlenecks due to demands for processor attention, as well as ensuring that the fastest I/O channels can be matched with available trunk bandwidth.

- Graphics hardware and software which are generally available and not customized for a particular site still leave much to be desired when matched against NASF requirements. Most notable, terminal costs and reliability, as well as response times, for complex 3-D displays need substantial improvement.

  However, graphics systems are receiving considerable industry attention and are being increasingly recognized as effective design tools. Also, developers of graphics systems seem to be placing growing emphasis on reducing, or eliminating, application dependence and equipment dependence. While these factors are favorable for expectations of adequate graphics capability, technology advances (such as the advent of the microprocessor) are providing cost improvements and increased reliability.

- As recommended by Ames study team personnel at the outset, compiling and scheduling of the NSS back-end is best performed on the front-end computing system. This makes possible early development and checkout of those very complex software elements on existing processors, well in advance of the availability of the NSS. Although experience has shown that a compiler operating on the target machine is better able to optimize code for the target machine, the time scale for this project dictates an early start on the compiler that could best be supported by existing equipment.

It would appear at this time that the best approach for the language processor is to identify the front-end processor as soon as possible. Then an examination of the existing compiling system on the front-end processor could determine the feasibility of using the basic front-end compiler with vector extension modifications to compile for the NSS. As much as possible, new compiler design, programming, and documentation must be reduced to accommodate the schedules.

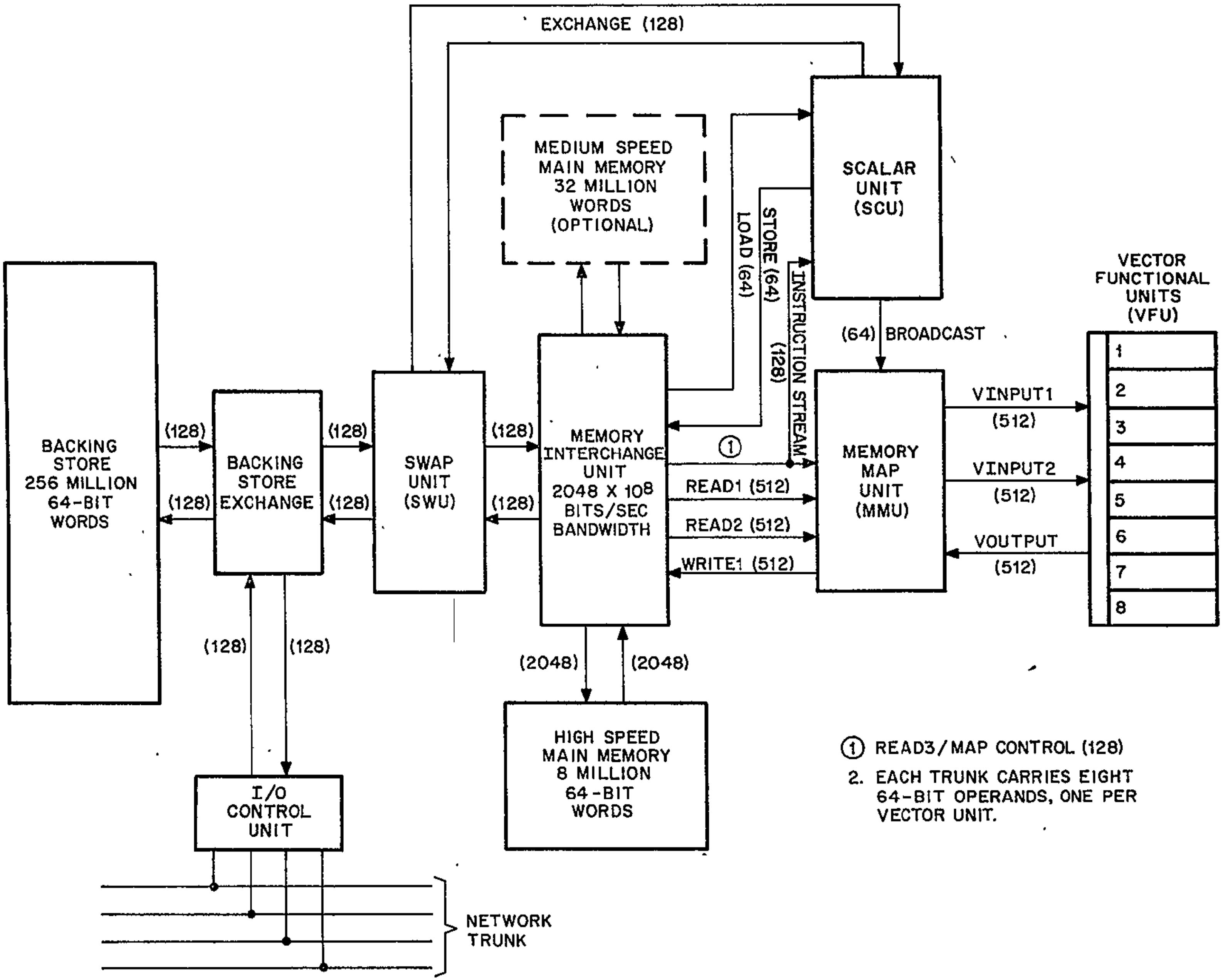
NSS PROCESSOR

Figure S-2 gives a broad overview of the proposed NSS processor. Each of the major blocks represents a separately designed, and somewhat modular, functional entity. The vector units, map unit, scalar unit and swap unit can operate concurrently with each other, and in many cases, independently of each other. The major architectural feature shown here, in addition to the massive memory and memory bandwidth, is the utilization of 'functional' parallelism. The process of extracting data from memory for processing, and putting it back again, is called mapping. Thus, the map unit can perform memory access operations for restructuring data, while the vector units are performing computations on a separate piece of data that is held in buffer registers within the vector units.

Correspondingly, the management of the memory hierarchy (the main memory and the backing storage unit) requires the addressing and transfer of large blocks of data. This operation can proceed at the same time as vector arithmetic and mapping. Finally, many setup and housekeeping chores are necessary in nature and can be performed concurrently with the swapping, mapping, and arithmetic.

The choice of 8 vector units was based on tradeoffs between the search for a higher performance logic family than exists today, the amount of trunking and data alignment required, and the maximum amount of hardware that appears feasible to assemble, from power, cooling, physical geometry, and reliability standpoints.

Some additional points to be considered in the design and utilization of the NSS are:



① READ3/MAP CONTROL (128)

2. EACH TRUNK CARRIES EIGHT 64-BIT OPERANDS, ONE PER VECTOR UNIT.

Figure S-2. Major NSS Components and Data Paths

- No existing computer system can perform the computations needed for 3-D Navier-Stokes solutions for flow field simulation. The NASF objective of complete solutions of these simulations in 7 to 15 minutes requires that such a processor achieve a sustained rate of computation between 1 to 2 gigaflops (1 to 2 billion floating-point operations per second). The fastest known machines today can attain peak rates for 64-bit computation slightly more than 100 million floating-point operations per second (100 megaflops), with sustained rates closer to 20 megaflops. This is a factor of 50 times slower than required.

- Given the projected technologies for the 1980's, no known existing computer architecture will yield the desired machine performance.

- With sufficient parallelism, such a machine is possible to design and build for operational employment in 1982.

- Key factors in achieving these goals are: the construction of sufficient memory to contain the entire problem on-line, without recourse to accessing slow speed mass storage devices; the ability to build a reliable collection of highly parallel hardware; and the programming and control of all the parallel hardware.

- Most of the data base (95 percent) can be maintained in 32-bit format, which reduces storage cost. Most of the computations (85 percent) can be performed in 32-bit form, with extended precision of at least 40 bits of coefficient required for a limited set of calculations. This makes possible the doubling of throughput of functional units when run in 32-bit mode instead of 64-bit mode.

- A processor containing a fast access memory of 8 million words of working storage and 256 million words of secondary storage can hold all projected problems. More importantly, such a memory can be made with known technologies and be made highly reliable through the use of error detection and correction techniques that are becoming commonplace in commercially available equipment.

- A processor with an 8-to 12-nanosecond clock and only eight separate functional units, each containing some localized parallelism, could achieve the 1-gigaflop threshold.

- The major problem to be solved in such an ensemble is that of sustaining the computing rate regardless of the manner in which memory is being accessed, linearly or randomly.

- Programmability and control of the necessary parallelism can be accomplished by melding together concepts taken from the STAR-100, the Texas Instruments ASC, and the ILLIAC IV.

- The most direct means for achieving programmability, reliability, and buildability is to begin with a single instruction stream, multiple data stream (SIMD) architecture.

- The NSS should be time-shared only in the most brute force manner, full rollout of the job in progress and the rollin of a new job, and then only in extraordinary circumstances. Otherwise, jobs should be permitted to go to completion.

RISKS

- Hardware risks anticipated for the proposed NASF range from negligible or minimal for front-end systems and network trunks, to moderate for graphics subsystems, to considerable for the NSS mainframe. The processors and peripheral devices with sufficient capability for front-end systems exist today and network trunks need little maturing to be sufficient. Graphics subsystems require some additional development of hardware and software as well as stabilization of approaches and techniques.

- To achieve the cost, performance, and reliability objectives established for this project, the NSS should be built with a second-generation, high-speed LSI. This technology is not yet available, and only expert opinion is available to ensure that this new generation will be available in time for the NASF. Alternative approaches can be taken yielding various degrees of reduced performance but decreasing the risk. Rough estimates of some of these approaches are:

    - Use of the planned STAR-100C for a run time of 30 minutes at essentially no risk.

    - An eight-pipe NSS using existing technology should yield a 15-minute run time with a risk factor of 0.1.

  - An eight-pipe NSS using double-density chips should yield a 10-minute run time with a risk factor of 0.35.

  - An eight-pipe NSS with double-density chips, 400-ps gate, should yield a 5-minute run time with a risk factor of 0.6.

- Software development absorbs an incredible amount of resources for even simple, uniprocessor systems. With much of the software expected to be used off-the-shelf, this risk can be ameliorated somewhat; however, considerable elapsed time will be required to stabilize the NSS compiler to the point where it can be put into general use. Three years is generally a minimum for such activity, even with a well-known language such as FORTRAN.

- The evolution of better algorithms for solving a system of partial differential equations such as the Navier-Stokes system could yield programs that would diverge radically from the form of the performance metrics. Thus, a specially tuned NSS could perhaps not be optimally tuned for the new algorithms.

- Although it is felt that costs and performance objectives can be tightly controlled to meet NSS requirements, scheduling remains very significant. The time frame is short, the technology is not yet in hand, and the design and simulation labor is extensive to produce the hardware complex. The biggest schedule risk, however, comes from the software development. Some steps which can be taken to help minimize the risk due to scheduling are:

  - Earliest possible initiation of each program phase, and earliest possible definition and stabilization of requirements.

  - Early selection of the front-end processor and leasing of time from the vendor of the target processor for software checkout.

  - Early release of software without all features for broader use and exercise of the software.

RECOMMENDED FUTURE WORK

The next logical step in the development of the NASF is to refine:

- The structure and architecture of the NSS computational engine
- The analysis of the various forms of 2-D Navier-Stokes solutions
- The 3-D versions of the Navier-Stokes codes
- The definition of the resulting 3-D version of the Navier-Stokes program as the performance metrics
- The preliminary Navier-Stokes programming for the proposed NSS
- The definition of the programming language
- The system structure, applying workload data for peak and average operating periods to demonstrate that the supporting system will be adequate
- The schedules for all remaining aspects of the program

The final work product of this effort should be a series of detailed specifications for every component, whether it be programs, hardware, or buildings to be used to direct the design and construction of the NASF as well as to measure progress throughout the project.

AFTERWORD

The phase I study of the NASF has been a worthwhile experience for Control Data Corporation, and in particular, the RADL study team. It should be apparent that the design of the system, and most specifically that of the NSS, has undergone revision and evolution. This came about through a process of give-and-take with the staff at Ames. With the openness and candor permitted by the cooperative nature of this study phase, it was possible, RADL believes, to arrive at a better solution for the NSS architecture than could have been arrived at solely by the best resources within Control Data or Ames.

The probability for success of this project will rely heavily on the continuation of this excellent contractual relationship between NASA and vendor study teams. Only by merging the strengths of hardware production experts and mathematical and programming specialists can the most optimum system be obtained with the least cost and risk to all.

COMPUTATIONAL AERODYNAMICS REQUIREMENTS - THE FUTURE ROLE OF THE COMPUTER AND THE NEEDS OF THE AEROSPACE INDUSTRY

Paul E. Rubbert
The Boeing Company
Seattle, Washington

ABSTRACT

This paper presents the commercial airplane builder's viewpoint on the important issues facing us in the development of improved computational aerodynamics tools. The success of these tools depends upon both good computer equipment and good software. The development of more powerful computers optimized for our fluid flow problems is an important first step. However, the success of this new effort in aerodynamics will also depend on how we plan to use it. We can develop a special purpose computer and reserve its use for its designers, or it can be run using the same philosophy that we use in operating our wind tunnels. Both of these approaches will tend to keep the one person who needs the machine the most, the engineering designer, from using this new tool to its fullest potential. This new computational aerodynamics tool should be more than a computerized wind tunnel. It has the potential for placing new and more powerful methods for analysis and design at the finger tips of the engineering designer. The usefulness of this new equipment and software depends upon how well we solve this user interface problem. The development of a new computer by NASA should serve as a "pathfinder" experience for equipment to be eventually purchased by industry. It should, therefore, be developed not as a single purpose tool, but should be viewed within the total framework of future use within the industrial environment.

INTRODUCTION

The role of computational aerodynamics methods today is to provide a limited ability to design and analyze aerospace vehicles. An increasing reliance is being placed upon these methods to initialize and optimize designs as the methods become substantiated, extended into various flow regions, made more accurate, and are provided in more useable fashion to the engineer. The future undoubtedly will demonstrate greater dependence upon these computational methods for vehicle design. The reason for this is that these new methods can systematically optimize designs in a timely manner and with less manpower than would be possible solely with wind tunnel testing. Moreover, due to the myriad configurations that can be evaluated, production risk is reduced. As we convince ourselves that our numerical modeling incorporates more of the physical aerodynamics, more responsibility will be given to the role of computational aerodynamics in the design process.

To meet this expanding role the aerodynamic computational needs hinge, to a certain extent, upon who will be the primary users of these methods; aerodynamics researchers or design engineers? These two groups have different needs and responsibilities. In addition, there exist two different categories of user communities; the builders of commercial aircraft, and those organizations participating in the design and construction of military vehicles.

This paper will identify the different needs of the people that use computational aerodynamics; their working environment in terms of time, cost, and risk constraints; and how these factors impact on our attempts to define the needs and future role of computational aerodynamics in the aerospace community. Since the military aircraft design community is adequately represented by the other speakers at this workshop, this paper will stress the impact of future software and computers as viewed by the commercial airplane builder.

THE USER OF COMPUTATIONAL AERODYNAMICS

In exploring the ideas introduced above, it is important to understand the viewpoint of the various people involved. The aerodynamics researcher is in the forefront of technology. He devises new computational methods, demonstrates their feasibility, and attracts attention to the potential gains in design technology that further implementation into an engineering tool would unleash. The researcher is primarily concerned with the development and proof of his mathematical algorithm. The primary user of computational methods in a commercial aircraft company is the design engineer who is concerned with solving practical engineering problems. The aerodynamics researcher may develop a new method and furnish a much needed insight into the modeled flow, but for the aircraft company, the payoff comes from the designer.

The end results of the researcher's labors are better tools for the engineering designer. Several important facets of the intermediate development process between feasibility demonstration and delivery of a useful tool have come to light during the past few years. In general, the engineering user has had to work with computer code written by the researcher, and apparently, for the researcher. Some researchers may disagree with this statement, but all one has to do is look at most of the computer code now available and review the program documentation to understand the point being made here. Most of the programs are very poorly documented internally and there is very little external documentation to tie the computer code to the theoretical development. All too often, very little thought has been given to the user and to the programmer who must work with, modify, and adapt the researcher's code. The end result has been tremendous hidden costs and greatly compromised effectiveness of the tool. These adaptation and maintenance costs can often be higher than the cost of the development of the original code.

From the engineering user's viewpoint the development of program interfaces and pre- and post-processing capability for new computational methods is just as important as the algorithms and machine architecture used to grind out the numbers. As we develop new methods that compute more and more details of the entire flow field, the visibility of output data becomes a major problem. The problem is then doubled when we add a design capability to a method. It is very important that the user be able to see, understand, and interpret the results calculated by our new and more powerful methods. Today, the engineering manhour costs expended because of the need to work with programs having only primitive user interfaces are enormous, and the research/development community is moving aggressively to improve the situation.

If as researchers we design algorithms that the engineering user cannot understand, and program them in some new language foreign to the user (or fail to use comment statements), then we will also have to provide all the needed user interfaces including the pre- and post-processing of the data. However, our recent experiences have indicated that the development of good software requires a team effort consisting of the researcher, the computer scientist/programmer, and the engineering user, the final customer. Fortunately, new software development techniques are available. The concepts of structured programming, top-down design, and related techniques are being applied to our new engineering programs for the first time. So far, the results have been highly encouraging. It is imperative that all future software which is intended as an end-item product be designed for maintainability, efficiency, growability,and independence from the development team, as is every other common fabricated product.

In the commercial airplane business few engineering design tasks are solved with a single computer program. The extremely high cost or flow time of running some of our new methods means that they cannot be used on every design cycle from the beginning to the end. Instead, we use our faster (and also more approximate) methods to zero in on a design that comes close to meeting our requirements. For example, we will use incompressible flow methods to arrive at a preliminary shape that, from past experience, we know will give certain desirable transonic characteristics. We then check our design with the more exact (and more expensive) analytical tools almost in the same manner that we have in the past used the wind tunnel. This process is repeated several times until we have a shape worth releasing for a wind tunnel test. The results from the wind tunnel test help us tune our computational methods and we go back through the analysis/design cycle again. Figure 1 illustrates the computational part of this design process.

The problem of computer run costs has a direct influence as to how we in the commercial airplane business use the computer. The magnitude of this problem is hard to impress on people who have not actually worked in our highly cost-constrained environment, and who are used to computer costs that are about one-fifth of those that we face. This cost problem is one of the driving forces behind our interest in this workshop. The computational specialists are coming up with tools that we need but cannot afford to use. A new machine, such as that envisioned for the proposed national computational facility,is an important step in solving this problem.

The design process mentioned above probably will be employed with the advent of new computational aerodynamic methods and machines. These new tools will not be able to solve all of our problems. For example, greater design capabilities will be needed to complement our improved analysis methods. Some of the boxes in Figure 1 will be replaced,but a variety of programs will still be required to get the job done. The design engineer will still be closely involved in the process. Interactive computing and interactive graphics can provide this important interface.

Remote access by the user is a key requirement for an aerodynamics computer. This is important both for the researcher-programmer-engineer team that is developing new programs, and for the eventual design engineer. We need to be able to access the new National Facility machine in its early stages so that we can learn how to use it, so we can properly evaluate it, and so we can be ready to justify the purchase of similar machines for our own facilities. From the previous discussion we can also see that a new machine must be versatile enough to handle a variety of engineering problems and algorithms in addition to those that it is specifically designed for.

A new computer will need a careful front end design so that we can have the maximum interface capability with remote interactive and batch terminals, mini-computers, and mini-computer/graphics facilities. We also need to be able to transmit data efficiently between our existing main-frame computers and the new generation machine. This will include both input geometry data from our remote data bases, and output data that will be needed by further analysis programs and for plotting.

The success or failure of this endeavor will depend, to a great extent, upon the degree of development of the computer operating system. We are particularly concerned about this because past experience has shown that specialized machines tend to only communicate with special problems and special people. The development of the operating system frequently lags far behind the hardware development. This means that our estimates of the facility costs should include a liberal allowance for development of the operating system.

In short, we are faced with a very difficult problem of designing a special purpose computer to match our new computational algorithms, yet still have a useable machine for our more general engineering problems, and be able to interface efficiently with the user. Are we asking for the impossible?

THE COMPUTATIONAL FACILITY CONCEPT

The NASA Computational Aerodynamics Design Facility project has established several key goals: the development of a national computational facility, a new computer optimized for the solution of fluid flow equations which would achieve at least a two order-of-magnitude performance gain over the general purpose CDC 7600 computer, and to provide a new tool for simulating 3-D viscous flows about aerospace vehicles. There have been some suggestions that this facility might also be viewed as being a National Aerodynamic Design Facility.

The proposed National Computational Facility if properly conceived and structured can provide a much needed service for the aerospace community. Historically, the government has pushed the development of advanced computers for defense purposes. The CDC 6600, 7600 and STAR computers were all initially contracted by the Lawrence Radiation Laboratories. This initial support has made it easier for computer manufacturers to service the limited scientific market with today's scientific computers. Without the government's needs, it is doubtful that scientific computer processors in the United States would have assumed the position of world leadership and excellence that they presently possess, and which are becoming increasingly essential for the maintenance of this country's leadership role in aerospace.

Advances in scientific computing capability clearly will be of benefit to our country, both commercially and militarily. The proposed new facility represents a strong forward thrust that will produce such advances. And, in the opinion of the Boeing Company, it is a proper role for NASA to promote the advancement of computational technology in this manner.

However, NASA should broaden its outlook beyond the needs of computational fluid dynamics. It should identify all aerospace technology areas that would benefit greatly by increased computer power and evaluate the computer configuration from this broader viewpoint. This must be done in a timely manner so as not to significantly delay progress.

The major risk involved in this project would be the evolution of an overly specialized processor; one that would be overly dedicated towards the specific, narrow set of computational algorithms we know today. One should seek a configuration that retains a high probability of compatibility or adaptability towards algorithms that are yet to be invented. A computer's particular operating characteristics channel the course of algorithm research. A computer which is too narrow in its performance characteristics would serve to constrain future computational research and development.

The proposed new computational facility has also been suggested as being a national computational design facility. We do not find this very appealing. As commercial airplane designers we face considerable cost risks in new airplane design. We want assurances that our design analyses are sufficiently comprehensive to minimize these risks. Basing a design on a new and unproven method could have a catastrophic effect on our company if we could not meet our guaranteed performance. Additionally, during the design process we need to be able to control usage priority of the facility. We could not afford the time to compete with other companies for machine access, i.e., job processing turn around.

With so much at stake, we in our business cannot afford to rely on someone else to "help" us with our designs. Of course, NASA sponsors much research that pushes the technology in all the technical areas. But we must weigh the potential advantages and risks involved as we incorporate new technology in an airplane that we build. It is very difficult to design and build a transport to the close performance guaranties that we have to work with. In evaluating the potential advantages of new technology and as we attempt to minimize the risk involved in each new design, we must make use of engineering tools that we trust and that have become a standard part of the design cycle. Our designers will use only the tools that they know well, that they depend on, and that they control themselves.

Because of these factors, Boeing would expect to use the proposed National Computational Facility only in circumstances where its unique capabilities showed an advantage over our own, and produced needed results that were otherwise unavailable. To avoid a dependence on facilities that we do not control, we pursue a continuing program of computer hardware acquisition to ensure that we have the necessary services. Implicit in this program is the philosophy of acquisition of a larger special purpose computer at a point in time where such a facility has been demonstrated to provide a unique and necessary capability.

In summary, the Boeing Company supports the concept of an advanced computer and believes that it is in the national interest to encourage its development. We believe that NASA is organized and chartered to do this effectively. However, it should not be configured and chartered as a National Design Facility. It should be viewed, rather, as a pathfinder, a prototype to stimulate advanced computer development and to serve as a vehicle for computational research and demonstration of advanced computational capabilities. It should be "national" in the sense that the facility should be made available for both government and non-government research and capability demonstration.

Our forecast for the future is that when computational capabilities are demonstrated (via this advanced host computer) which clearly have the potential of improving the airplane design process and/or reducing design risk to a degree compatible with the investment, private industry will acquire advanced computers which are tailored to their overall requirements. These will be improved machines that benefit from the perspective hindsight provided by the pathfinder experience, both in terms of processor configuration and capabilities, and in terms of providing a data base of total procurement and software costs.

TRANSONIC WING DESIGN

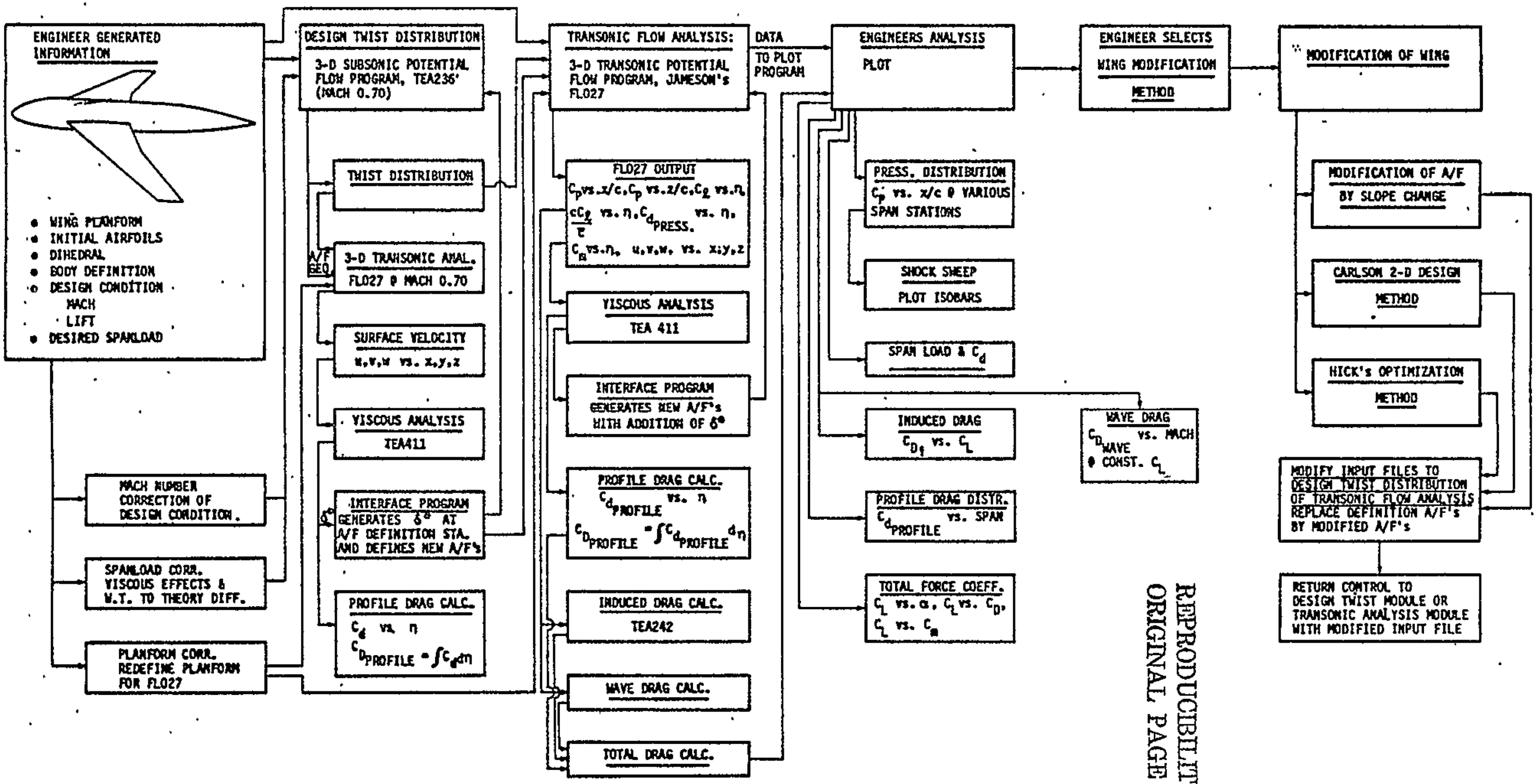

ENGINEER GENERATED INFORMATION
• WING PLANFORM
• INITIAL AIRFOILS
• DIHEDRAL
• BODY DEFINITION
• DESIGN CONDITION
MACH
LIFT
• DESIRED SPANLOAD
MACH NUMBER CORRECTION OF DESIGN CONDITION.
SPANLOAD CORR. VISCOUS EFFECTS & W.T. TO THEORY DIFF.
PLANFORM CORR. REDEFINE PLANFORM FOR FL027
DESIGN TWIST DISTRIBUTION
3-D SUBSONIC POTENTIAL FLOW PROGRAM, TEA236' (MACH 0.70)
TWIST DISTRIBUTION
A/F GEO.
3-D TRANSONIC ANAL.
FL027 @ MACH 0.70
SURFACE VELOCITY
u,v,w vs. x,y,z
VISCOUS ANALYSIS
TEA411
δ*
INTERFACE PROGRAM GENERATES δ* AT A/F DEFINITION STA. AND DEFINES NEW A/F's
PROFILE DRAG CALC.
C_d vs. η
$C_{D_{PROFILE}} = \int C_d d\eta$
TRANSONIC FLOW ANALYSIS:
3-D TRANSONIC POTENTIAL FLOW PROGRAM, JAMESON's FL027
FL027 OUTPUT
C_p vs. x/c, C_p vs. z/c, C_ℓ vs. η
$\frac{cC_\ell}{\bar{c}}$ vs. η, $C_{d_{PRESS.}}$ vs. η,
C_m vs. η, u,v,w, vs. x,y,z
VISCOUS ANALYSIS
TEA 411
INTERFACE PROGRAM GENERATES NEW A/F's WITH ADDITION OF δ*
PROFILE DRAG CALC.
$C_{d_{PROFILE}}$ vs. η
$C_{D_{PROFILE}} = \int C_{d_{PROFILE}} d\eta$
INDUCED DRAG CALC.
TEA242
WAVE DRAG CALC.
TOTAL DRAG CALC.
DATA TO PLOT PROGRAM
ENGINEERS ANALYSIS
PLOT
PRESS. DISTRIBUTION
C_p vs. x/c @ VARIOUS SPAN STATIONS
SHOCK SWEEP
PLOT ISOBARS
SPAN LOAD & C_d
INDUCED DRAG
C_{D_i} vs. C_L
PROFILE DRAG DISTR.
$C_{d_{PROFILE}}$ vs. SPAN
TOTAL FORCE COEFF.
C_L vs. α, C_L vs. C_D,
C_L vs. C_m
WAVE DRAG
$C_{D_{WAVE}}$ vs. MACH @ CONST. C_L
ENGINEER SELECTS WING MODIFICATION METHOD
MODIFICATION OF WING
MODIFICATION OF A/F BY SLOPE CHANGE
CARLSON 2-D DESIGN METHOD
HICK's OPTIMIZATION METHOD
MODIFY INPUT FILES TO DESIGN TWIST DISTRIBUTION OF TRANSONIC FLOW ANALYSIS REPLACE DEFINITION A/F's BY MODIFIED A/F's
RETURN CONTROL TO DESIGN TWIST MODULE OR TRANSONIC ANALYSIS MODULE WITH MODIFIED INPUT FILE

Figure 1

SESSION 4

Panel on COMPUTATIONAL AERODYNAMICS REQUIREMENTS

Paul E. Rubbert, Chairman

11

Remarks On
Future Computational Aerodynamics Requirements

R. G. Bradley & I. C. Bhateley
General Dynamics/Fort Worth Division
Fort Worth, Texas

Abstract

General Dynamics is aware of the need to continually upgrade and expand computational aerodynamics methods for the design and analysis of aircraft configurations. We concur with NASA's plan to expand the Nation's computational capability and desire to participate in that activity. However, we do feel that several considerations are important in such a program. The development should be performed by both government and industry to ensure that the objectives for aircraft design are satisfied from both the industrial competitive design standpoint and from the government standpoint. We further feel that it is imperative that any programs developed be heavily user-oriented and that they provide maximum visibility and creditability to management. We at General Dynamics agree that the government should proceed with developing advanced processors specifically for the purpose of solving aerodynamics problems, and we support the NASA plan for the development of a national computational facility. However, we do feel that early consideration should be given to the adequate management of such a facility when it becomes available.

INTRODUCTION

From an aerospace industry point of view, the goal of computational aerodynamics can be simply stated as follows:

- o To develop a digital flow-simulation capability that can be used with confidence as an aircraft design and analysis tool.

The objective, then, is to speed up the design process by cost-effective aerodynamics calculations that can significantly reduce the amount of wind tunnel testing required in the development of new aircraft systems.

Although measurable progress has been made over the past years in reaching this goal, from a practical standpoint, a vast amount of development work remains to be done. This development work involves both the adequate numerical simulation of the physical flow processes that occur and the advanced processor equipment needed to provide solutions that are both accurate and cost effective. It is, of course, the stated objective of this workshop to address both of these development needs.

In an attempt to evaluate General Dynamics' computational aerodynamics program development and usage at the Fort Worth Division, we have compiled some statistics over the time period 1971-1977. Figure 1 shows the trends in the number of new computer codes generated per engineer and the total computational time per engineer. The trend shows a steady decrease in the number of new computer codes initiated in the aerodynamics area. This seems to be a bit surprising since fairly significant gains in computational aerodynamic methodology have occurred over this time period. Possible explanations for this trend are that programs being developed are more complex, the government is taking a more active role in methodology development, less money is allocated for the development of computational methods, computational potential is not realized or appreciated by management, and correlation of existing computational codes with experimental evidence is lacking, which in turn may lead to management's disenchantment in the computational field. Also, much more of the simpler tasks are being done on programmable calculators, thus reducing the number of large computer programs that are generated.

Fort Worth Division Statistical Data

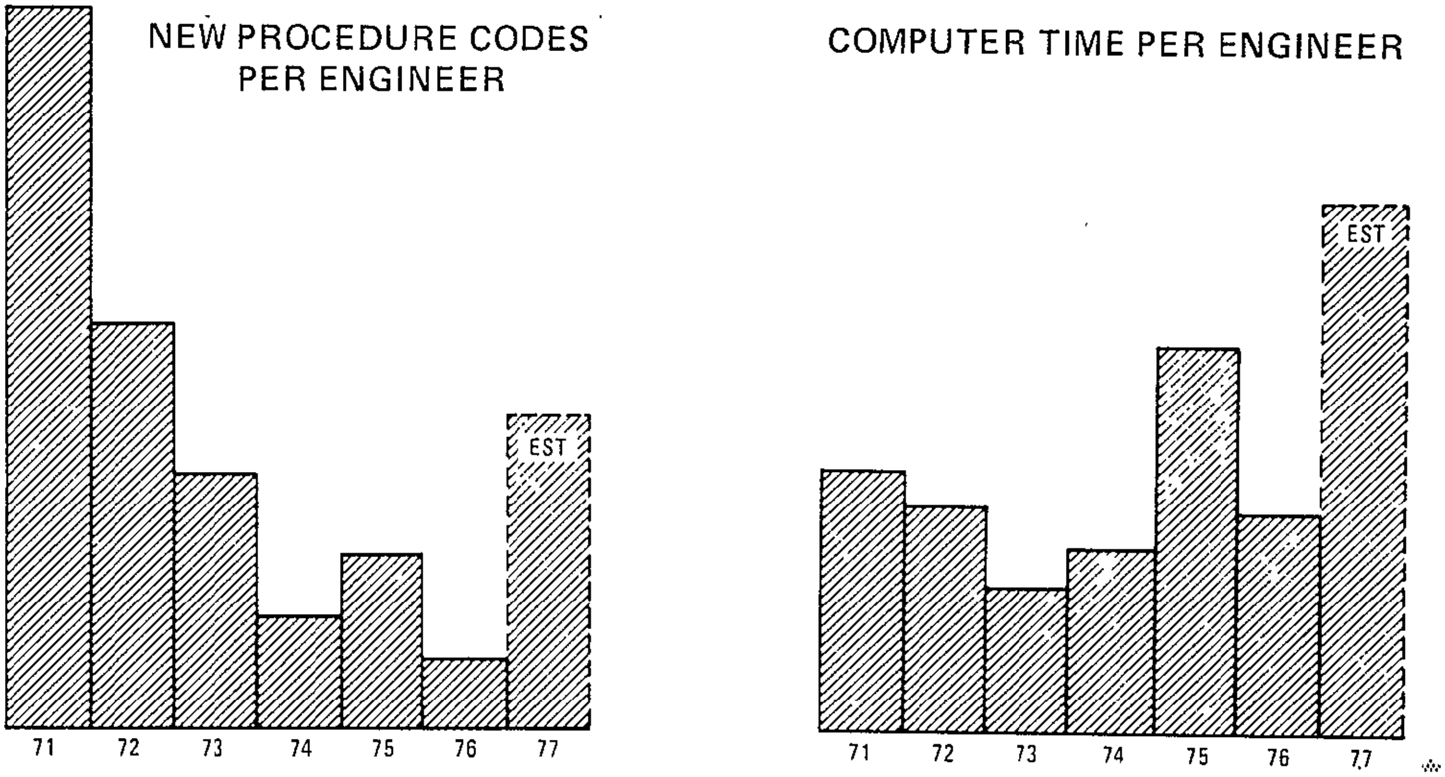


Figure 1 Recent Trends in Aerodynamics Computer Usage

On the other hand, it can be seen in Figure 1 that computer time has been generally increasing. This could be as a result of more complexity in the computer codes, longer running codes, or just more application of computer technology in the design process. The basic difference in the trend between program development and program usage seems to indicate that the tendency at Fort Worth, at least, has been toward fewer program developers and more program users.

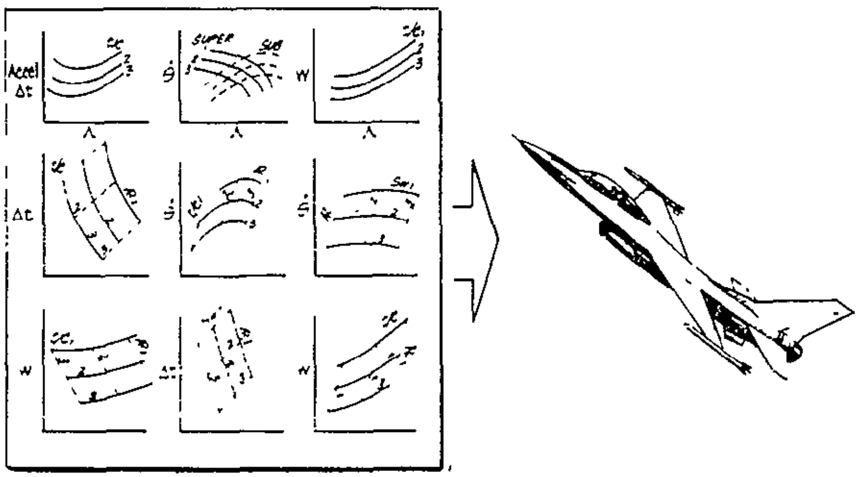
While these trends in themselves may seem to have no particular significance to this meeting, since they represent the experience of one aerospace company, they do lead to a point that is felt to be important. Basically, one may divide computational aerodynamicists into two basic categories: (1) the program developer or program generator and (2) the program user (the airplane designer). In the following discussion, we will examine the computational aerodynamics requirements of the future from the standpoint of the program user, that is, the aircraft designer. We take this approach since it seems imperative that the future success of advanced computational techniques and, indeed, the development of a national computational facility as proposed by NASA will depend very much on the endorsement of the program user rather than on the enthusiasm of the program developer.

DESIGN REQUIREMENTS

The aircraft design process may be divided crudely into two basic activities: preliminary design, which is defined to include conceptual design, and detailed design. Figure 2 presents a listing of the basic difference in these two activities and the computational requirements that are peculiar to each.

In the preliminary design area, computational requirements are for speedy programs that give rapid calculations showing trends for early design decisions during configuration development. Normally, these methods are available as short computer programs; currently, in more cases, they are programs that are designed for advanced programmable calculators. Examples of methods that may be used for this class of design are empirical or semi-empirical handbook-type aerodynamic lift and drag methods, wave-drag methods, performance calculation techniques, and design synthesis programs which are multi-disciplinary programs to give design trends and sizing for fixed missions. Also included in this category of computational methods are methods that are based on data banks for correlation and validation of the quickie predesign methods. Limited configuration optimization procedures may also be placed in this general class of problems. These basic

PRELIMINARY DESIGN



COMPUTATIONAL PROBLEMS:

- Preliminary Sizing to Mission Requirement
- Configuration Trade Studies
- Gross Optimization Studies for a Large Number of Configurations

SOFTWARE REQUIREMENTS:

- Rapid, Efficient Computational Methods
- Multi-Discipline Synthesis Techniques
- Handbook-Type Aero-Methods
- Interactive Graphics

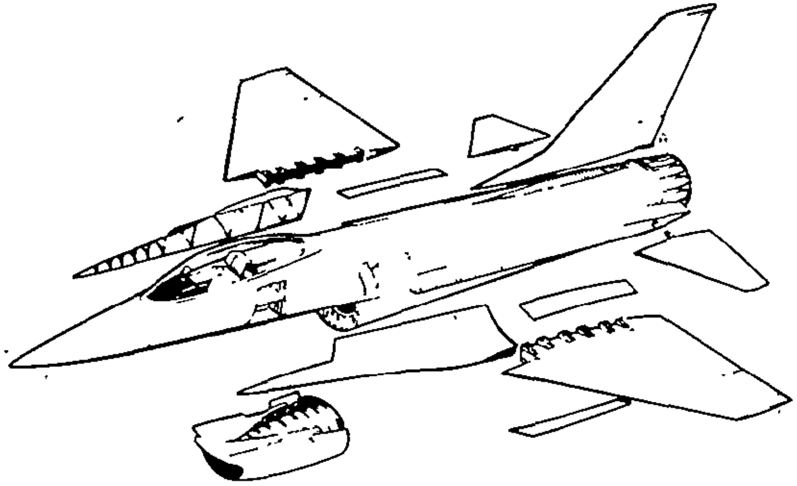
HARDWARE REQUIREMENTS:

- Existing Computer and Mini & Micro Processors
- Programmable Calculators

DEVELOPMENT RESPONSIBILITY:

- Continuing Industry In-House Development
- Government/Industry Interaction

DETAILED DESIGN



COMPUTATIONAL PROBLEMS:

- Configuration Details
- Air Loads
- Prop/Airframe Int
- Stability Derv
- Aero Characteristics
- Weapons Carr & Sep
- Power-Induced Effects
- Wing Design

SOFTWARE REQUIREMENTS:

- Detailed Flow-Field Definition
- Arbitrary Geometry Capability
- Real-Viscous-Flow Analysis
- Interactive Graphics

HARDWARE REQUIREMENTS:

- Existing High-Speed Computers
- New Special-Purpose Processors

DEVELOPMENT RESPONSIBILITY:

- Joint Govt/Aircraft Ind Software Development
- Joint Govt/Comp Ind Hardware Development

Figure 2 Aircraft Design Requirements

computer programs are generally developed by industry and are continually in a state of update and reformulation. In this area, perhaps it is appropriate that industry should be responsible for the development of the programs it uses, with some interaction between the government agencies who will be evaluating configurations.

On the other hand, the detailed design task is characterized by more complex computer codes and more detailed flow-field-oriented calculation methods. These computational methods are used for the detailed development of configurations, analysis of design changes, and effects of interacting flow fields and components. Appropriately, such detailed methods should be developed jointly by government agencies and by industry. These programs should require the maximum computational power that is available, and it is for this purpose that new processors are appropriate. Some examples of specific problems that need to be addressed in the detailed design phase are noted in the figure. The overall objective of the detailed design computational methods should be to minimize wind tunnel test requirements and to make design decisions concerning configuration features. Also included in this classification is the efficient analysis of test data for incorporation into the design process.

In the remainder of this discussion, we will concentrate on the detailed design methods. It is the further development of these methods that requires large expenditures in funds to develop satisfactory flow models and to advance the state of the art in computational capability in the form of new processors and new computational hardware. Advances in the design and analysis capability in this area are dependent very strongly on cooperative government and industry program development.

NEAR-TERM COMPUTATIONAL REQUIREMENTS

Near-term computational requirements are defined to be those computational methods that may be derived within the framework of currently available or state-of-the-art computers. Figure 3 presents a listing of the solution capabilities that are considered necessary in the near future as well as other considerations that are important in developing these computer procedures. Whereas, a great deal of work has been accomplished in the solution of general 3-dimensional inviscid flow fields in the subsonic and supersonic regime, a need still exists for improved accuracy in the transonic calculation area. This is particularly true for aircraft configurations, geometries including wing, body, tail surface or canard surface, and their inter-

STATE-OF-THE-ART COMPUTERS

SOLUTION CAPABILITY

- Arbitrary Body 3-D Inviscid Analysis in Sub/Trans/Supersonic Flow
- Semi-Empirical Viscous Analysis of 3-D Bodies in Sub/Trans/Supersonic Flow

EXPERIMENTAL DATA COUPLING

- Consolidation and Cataloging of Test Data Base for Rapid Random Access
- Combined Theory/Test Evaluation and Extrapolation

USER ORIENTED STRUCTURE

- Simplified Input/Output
- Interactive Graphics
- Standardized Subprogram Structure

MANAGEMENT ORIENTED PROGRAMS

- Cost-Effective Programming
- Acceptable Accuracy Validation
- Management Visibility of Payoffs

JOINT IND/GOVT DEVELOPMENT AND UTILIZATION

Figure 3 Near Term Computational Requirements

actions. Further, within the framework of the currently available computers, a workable 3-dimensional viscous-solution capability is needed. Such a capability must consider semi-empirical correlations of turbulence and separated flow to be viable in the near-term.

A further requirement in the computational area deals with the efficient use of the computer in evaluating and applying experimental data to the design process. This requirement covers massive test data handling, including display of the data and automated methods of analysis of the data. Oftentimes, industry collects a vast data bank of information in the development of a configuration. Unfortunately, these data are stored away and used only sparsely in other programs. Computer methods to store these data in the form of a data bank by combining it with theory to use in the design process could be developed.

Of prime significance in the development of computational methods is a program structure that is oriented toward the user - the designer. The obvious implications of this requirement are simplified input and output and simplified descriptions of complex geometries that can be put into the machine and used in a wide variety of computational methods. Interactive graphics is necessary to efficiently use these programs in design. Further, programs should be developed with a standardized subprogram structure so that combinations of programs may be easily and efficiently generated to meet particular design objectives. It is often the experience in industry that complex computer codes can only be used by the program generator, thus making the practical application of these methods in a variety of programs impractical if not impossible. The key issue to be addressed is the issue of ease of usage.

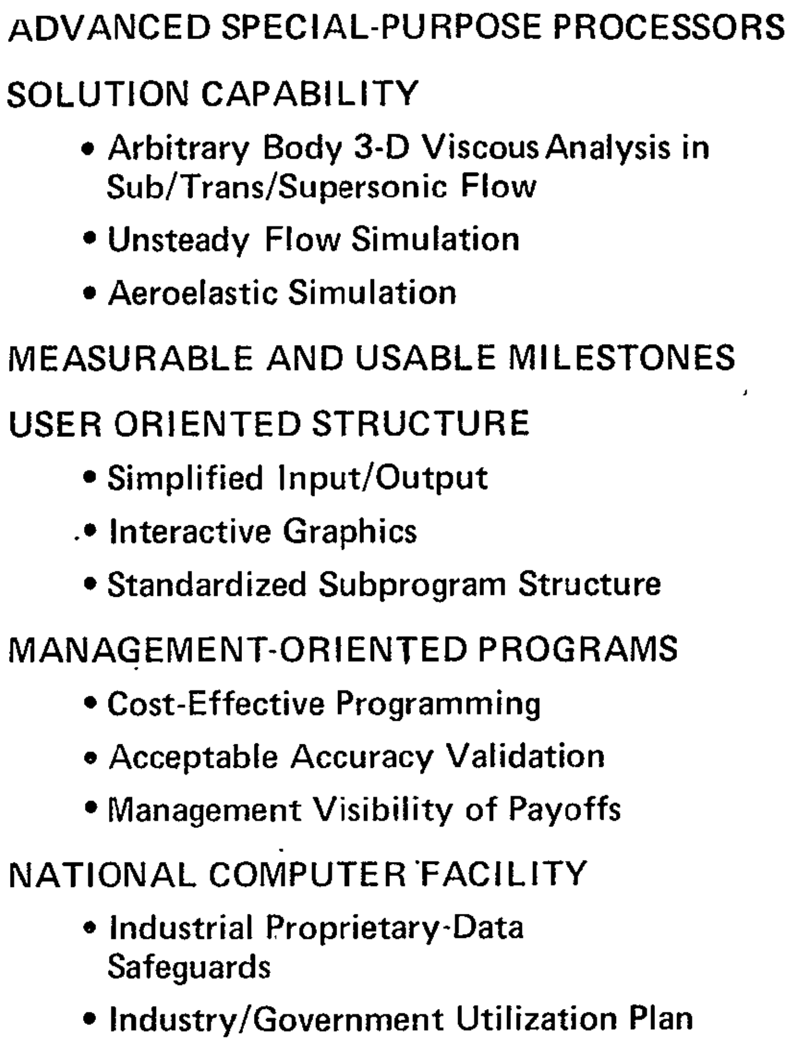
Another important issue in the development of computational capability is the need to have programs and output oriented toward management decisions. This means simply that the programs must be cost effective and not use large amounts of the computational budget for a development program, they must have a high level of accuracy validated by comparisons with experiment, and they must provide the necessary visibility for an understanding of the significance of the computed results. Management is reluctant to act on results from computational methods when these simple criteria have not been satisfied. To sell a computational method and, more importantly, to sell the development of specialized advanced processors, management considerations in the design process must be considered and weighed heavily.

In the near-term, it is felt that the continued cooperative development of these computational procedures with both government and industry working together is important. From an industry standpoint, development of such methods is costly; government interaction is important to ensure that technology steps are taken to develop the program. Further, from the government's standpoint it is desirable that computer codes used in the design process are thoroughly understood by government and can be accepted with confidence in evaluation of proposed configurations.

FUTURE COMPUTATIONAL REQUIREMENTS

Future computational requirements are defined as those that can be initiated with state-of-the-art processor equipment but, to be used efficiently in the design process, require development of advanced processors, perhaps tailored for an aerodynamic simulation. A list of future requirements in the area of computational aerodynamics is presented in Figure 4. The obvious requirement is a completely generalized 3-dimensional viscous-flow solution capable of modeling the 3-dimensional average Reynolds stresses, including simulation of unsteady flows, large eddy structure where appropriate, and aeroelastic interactions between the configuration geometry and the flow field. Some of the detailed planning on how such a solution capability can be devised is, of course, the subject of future panels in this workshop. The task is a formidable one and will require intensive cooperation between government agencies and industry to make its development practical. The ultimate goal is achievable, but not without the expenditures of large amounts of both money and engineering talent.

If this challenging goal is to be realized, three considerations are felt to be of paramount importance: First, the program must develop in a building-block manner so that milestones along the way are measurable and are usable to both industry and government. At each step along the way, extensive correlation and validation with experimental data are essential. Further, the sub-programs or the steps that are taken in the development, must be usable in a sense that actual design applications may be made with confidence. Second, the capability must be developed with careful attention to user requirements, as was discussed in the previous section. Third, and perhaps most importantly, management must have the visibility and the confidence in computational methods that is necessary to assure continued commitment of the resources necessary to develop the methods.



ADVANCED SPECIAL-PURPOSE PROCESSORS

SOLUTION CAPABILITY

- Arbitrary Body 3-D Viscous Analysis in Sub/Trans/Supersonic Flow
- Unsteady Flow Simulation
- Aeroelastic Simulation

MEASURABLE AND USABLE MILESTONES

USER ORIENTED STRUCTURE

- Simplified Input/Output
- Interactive Graphics
- Standardized Subprogram Structure

MANAGEMENT-ORIENTED PROGRAMS

- Cost-Effective Programming
- Acceptable Accuracy Validation
- Management Visibility of Payoffs

NATIONAL COMPUTER FACILITY

- Industrial Proprietary-Data Safeguards
- Industry/Government Utilization Plan

Figure 4 Future Computational Requirements

NASA/Ames has proposed the development of a national computational facility. The plan is a valid one if the general 3-dimensional viscous problem is to be solved. It is the feeling of General Dynamics that early management planning is essential to the development of such a national facility. Problems that must be resolved include the protection of proprietary data and a utilization plan that will ensure industry access to the facility in a timely manner. As an example, consider a major aircraft proposal in which over a period of from 30 to 60 days as many as eight companies may require extensive computational support. How the facility's time may be managed in such a case needs to be addressed and clearly set forth early in the development program.

CONCLUDING REMARKS

General Dynamics is aware of the need to continually upgrade and expand computational aerodynamics methods for the design and analysis of aircraft configurations. We concur with NASA's plan to expand the Nation's computational capability and desire to participate in that activity. However, we do feel that several considerations are important in such a program. The development should be performed by both government and industry to ensure that the objectives for aircraft design are satisfied from both the industrial competitive design standpoint and from the government standpoint. We further feel that it is imperative that any programs developed be heavily user-oriented and that they provide maximum visibility and creditability to management. We at General Dynamics agree that the government should proceed with developing advanced processors specifically for the purpose of solving aerodynamics problems, and we support the NASA plan for the development of a national computational facility. However, we do feel that early consideration should be given to the adequate management of such a facility when it becomes available.

FUTURE REQUIREMENTS AND ROLES OF COMPUTERS IN AERODYNAMICS

by

Thomas J. Gregory*

Ames Research Center, NASA
Moffett Field, California

This conference is addressing two very exciting topics: the solution of the Navier-Stokes equations, and the development of very fast digital computers. While faster computers will be needed to make solution of the Navier-Stokes equations practical and useful, most all of the other aerodynamic solution techniques can benefit from faster computers. Since there is a wide variety of computational and measurement techniques, the prospect of more powerful computers permits extension and an enhancement across all aerodynamic methods, including wind-tunnel measurement. It is expected that, as in the past, a blend of methods will be used to predict aircraft aerodynamics in the future. These will include methods based on solution of the Navier-Stokes equations and the potential flow equations as well as those based on empirical and measured results.

The topics (figure 1) of this paper are to first identify the primary flows of interest in aircraft aerodynamics, then to comment on some of the predictive methods currently in use and/or under development, and then finally to analyze two of these methods in terms of the computational resources needed to improve their usefulness and practicality.

The left-hand side of figure 2 shows a partial list of the primary flows of interest in aircraft aerodynamics. Each is a complicated topic in itself and solution can be attempted with a variety of methods. The right-hand side of the figure shows the primary predictive techniques in terms of their mathematical basis. There is no attempt to order or rank these methods or the flows in terms of importance or present utility. However, for predicting aerodynamics on today's aircraft, certainly the most extensively used methods are the theoretical/empirical approaches which are backed up by both wind tunnel measurements and detailed calculations using specific solutions of the fluid flow equations. At the present time the theoretical/empirical approaches are used primarily in conceptual/preliminary design and wind tunnel measurement approaches are used in final design. The other methods are important, are widely used, and are expected to be more useful in the future.

For complicated flows, i.e. interactions of the aerodynamic flows listed on the left side of the figure, it is anticipated that a blend of methods will be used in the foreseeable future. For this to occur, each of the methods must be developed sufficiently to be very useful to the aerodynamicist, that is, to be sufficiently reliable, accurate, and cost effective. Depending upon the particular problem to be analyzed and the resources available, the choice is usually easy.

*Chief, Aircraft Aerodynamics Branch.

Faster computers should be beneficial in using all these methods. As an example, consider the theoretical/empirical techniques as used in conceptual and preliminary design (figure 3). Computer programs currently exist in most organizations that will predict aircraft coefficients for conceptual aircraft designs. The U. S. Air Force DATCOM system which is currently being computerized is a typical example. These programs usually have between 5,000 and 15,000 executable statements without significant looping or matrix manipulations.

In a typical conceptual design study the aerodynamics of 20 configurations are needed to explore tradeoffs and select a preferred approach. In general, the aerodynamics at approximately 20 flow conditions are required and a typical aircraft synthesis in the future is expected to have 100 design variables and/or vehicle constraints. Design variables include items such as wing sweep, thickness, inlet size, and the coefficients of shape functions used to describe the surface of the vehicle. Typical constraints include take-off and landing field length, maximum speed, turn rate, approach speed, etc. Current vehicle synthesis requires approximately 20 vehicle evaluations per optimization where optimization means the best combination of the design variables that produce the minimum weight (or cost) design while still meeting all the specified constraints. If the typical numbers in figure 3 are multiplied out, the conceptual preliminary design task requires approximately 2.4 x 10^{10} floating point operations just to predict the aerodynamics in this type of aircraft design process.

The CDC 7600 computer is a representative high-speed scientific computer with roughly a speed of 5M floating point operations per second. Hence, the CPU time to calculate the aerodynamics for the above described design study would be approximately 1.3 hours. The other aeronautical disciplines such as propulsion, structures, weights, etc. would require similar computation times. The fastest uniprocessor that might be foreseen may be approximately five times faster than the 7600, i.e. 25M floating point operations per second, hence we could expect the central processing unit (CPU) time to be approximately 20 minutes on such a machine. If the Navier-Stokes solver (NSS) which has been the subject of this workshop, could use parallel and pipe-line concepts to achieve one gigaflop, that is, 10^9 floating point operations per second, then the resources needed in the aerodynamic prediction for conceptual/preliminary design would be approximately 20 seconds to 120 seconds depending upon the degree that the parallelism could be accommodated in this problem. It is expected that today's aerodynamicists and computer scientists would be sufficiently clever to make high use of the parallel and pipe-line features of the NSS. This presumes that the architecture of the NSS would permit programming the sequence of floating point operations and that these floating point operations would include some means for computing transcendental functions.

The anticipated increase in speed for conducting the conceptual preliminary design activity would permit quantitative evaluation of a wide variety of airplane concepts as contrasted to the present circumstances where many selections are made on a qualitative basis due either to inadequate time or resources. Hence, a much better design selection would be expected at the conclusion of the preliminary design process. Of course the answer is very dependent upon the accuracy and generality of the theoretical/empirical techniques inherent in these approaches. Continual enhancement of these techniques is taking place and needs to be continued. One of the best ways of enhancing the theoretical/empirical techniques, is by incorporating the results from more complicated theoretical methods into the empirical data base that the simpler methods are based upon.

Another place that faster computers are required is in aerodynamic paneling, that is, the use of linear potential flow techniques that depend upon large matrix manipulations to predict the pressures at every point on an aircraft surface. Figure 4 describes the aerodynamic prediction resources needed in a typical preliminary design activity in which paneling is used to calculate the pressures, forces and moments on a complete aircraft. Approximately 3,000 panels are needed to describe the surface detail on a realistic aircraft. (At this time it is uncertain whether a finite difference Navier-Stokes solution within a 100^3 grid would give sufficient resolution of a complicated flow field about a complete aircraft.)

For a single calculation of the flow, approximately 5×10^{10} floating point operations (FLOP) are required. This is comprised of approximately 3.6×10^{10} floating point operations for the generation of the aerodynamic influence coefficient matrix. The solution of this matrix would require another 1.3×10^{10} FLOP. Hence, to calculate one flow on the 7600, approximately 2.8 hours would be required. The fastest uniprocessor would require approximately 30 minutes. The Navier-Stokes solver running at one gigaflop would require approximately one minute. Again, there is a major assumption that aerodynamicists and computer scientists would be clever enough to make use of the parallelism and pipe-line features of the NSS.

Practical calculation of all the linear aerodynamic range for a configuration may lead to a reduction in the amount of wind tunnel testing currently used in this flow regime, but a reduction would ultimately depend upon whether these solutions could be generated quickly, accurately, and cost effectively.

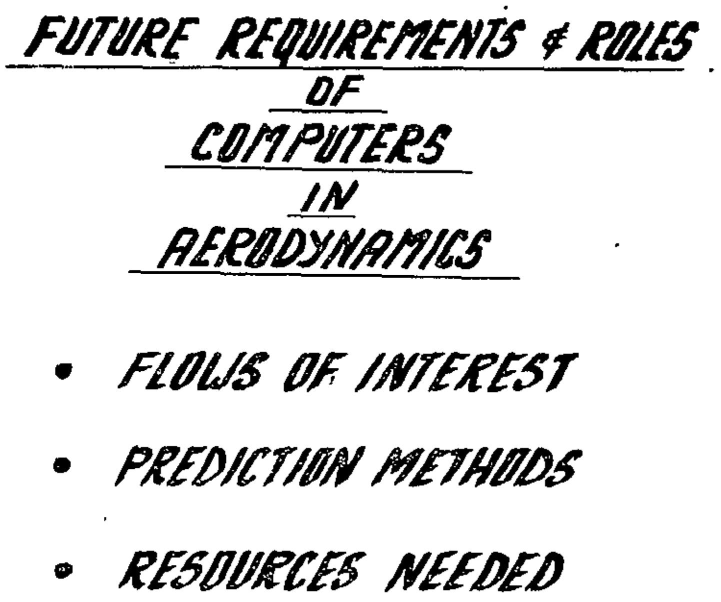
There are four points in summary, as shown in figure 5. First, as in any engineering activity, the selection of the proper method will be that which fits the problem best. For example, we will use linear potential flow methods in subsonic and supersonic attached flow cases. We would not use more complex methods. However, if a less expensive or quicker method is required, then possibly the theoretical/empirical methods would be used instead.

Finite difference methods have a unique place in aerodynamic prediction, particularly in transonic aerodynamics or in highly turbulent separated flow. But even with the advanced computer, finite difference solutions of the Navier-Stokes equations will probably describe only the complicated flows about components and not complete aircraft.

All aerodynamic methods will benefit from faster computers, especially if they are reliable and less costly than other computers. Future computational speed is expected to come through basic technology, but primarily from parallel and pipe-line architecture. A major challenge in aerodynamics will be to capitalize on these characteristics.

The most complicated flows in aerodynamics involve interactions of turbulence, potential flows, shock-waves, mixing, or vortices. The solution of these combined flows will probably require hybrid methods. The major elements in these hybrid techniques are likely to be measurements made on wind tunnel models. However, continued development of all aerodynamic predictive methods is the prudent direction to follow, since all will benefit from advancements in computers.

Acknowledgment. The author expresses appreciation to Ralph Carmichael, Larry Erickson, and Richard Medan for assistance in estimating the computer operations and times required for aerodynamic paneling.



FUTURE REQUIREMENTS & ROLES
OF
COMPUTERS
IN
AERODYNAMICS

- FLOWS OF INTEREST
- PREDICTION METHODS
- RESOURCES NEEDED

Figure 1 Topics To Be Covered

AERODYNAMIC FLOWS

- POTENTIAL
- BOUNDARY LAYERS
- TURBULENT SEPARATION
- TRANSONIC
- MIXING
- VORTICAL
- INTERACTIONS OF THE ABOVE

AERODYNAMIC PREDICTIVE METHODS

- THEORY + EMPIRICAL
- SPECIFIC SOLUTIONS
- LINEAR ALGEBRA
- FINITE DIFFERENCE
- FINITE ELEMENT
- MEASUREMENTS
- HYBRIDS

Figure 2 Predictive Methods for Aerodynamic Flows

AERODYNAMIC PREDICTION RESOURCES NEEDED

CONCEPTUAL/PRELIMINARY DESIGN

- THEO. + EMPIRICAL FORMULAS

~ 6000 EXECUTABLE STATEMENTS
~ 5 FLOATING PT. OPS./STMT.
~ 20 CONFIGURATIONS · 20 FLOWS
~ 100 DESIGN VARIABLES/CONSTRAINTS
~ 20 VEHICLE EVALUATIONS/OPTIMIZATION

≈ $2.4 \cdot 10^{10}$ FLOATING PT. OPS.

	7600 5 MF	FASTEST UNIPROCESSOR 25 MF	NSS 1 GF
CPU TIME	1.3 HR.	~ 20 MIN.	~ 20-120 SEC.

Figure 3 Resources Needed for Conceptual and Preliminary Design

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

AERODYNAMIC PREDICTION RESOURCES NEEDED

PRELIMINARY DESIGN

PANELING

3000 PANELS (N)

1 FLOW = 5×10^{10} FLOP

MATRIX GENERATION

$4KN^2 = 3.6 \cdot 10^{10}$ FLOP

MATRIX SOLUTION

$1/2\ N^3 = 1.3 \cdot 10^{10}$ FLOP

	7600 5 MF	F. UNI. 25 MF	NSS 16 F
CPU TIME/FLOW	~2.8 HR.	~30 MIN.	~1 MIN.

Figure 4 Resources Needed for Preliminary Design

SUMMARY

1. PROBLEM DEFINES METHOD
2. FINITE DIFFERENCE METHODS TO BE INCLUDED
3. ADVANCED COMPUTERS USEFUL TO ALL METHODS
4. HYBRID METHODS NEEDED

Figure 5 Summary Points

D9

PROJECTED ROLE OF ADVANCED COMPUTATIONAL AERODYNAMIC METHODS AT THE LOCKHEED-GEORGIA COMPANY

Manuel E. Lores
Lockheed-Georgia Company
Marietta, Georgia

SUMMARY

Advanced computational methods are being used at the Lockheed-Georgia Company to aid in the evaluation and design of new and modified aircraft. Experience with these methods and their attendant computer requirements indicates that large and specialized computers like the proposed Numerical Aerodynamic Simulation Facility will be needed to make advanced three-dimensional viscous aerodynamic computations practical. The cost of such a computer and the fact that it will provide computer capabilities greatly exceeding the day-to-day needs of any one aerospace company make the proposed facility a viable approach. Therefore, plans to develop such a facility and continued emphasis on improved computational aerodynamics are supported by Lockheed-Georgia.

The ultimate purpose for the Numerical Aerodynamic Simulation Facility should be to provide a tool for designing better aerospace vehicles while at the same time reducing development costs. Its primary function should be to perform computations using Navier-Stokes equations solution algorithms since complex viscous flows which can only be solved using those equations play a dominant role in vehicle aerodynamics. However, the computing facility should have a secondary function of permitting less sophisticated but nevertheless complex calculations to be made efficiently because such results can often be useful in a design study.

Experience with current large computer programs indicates that the proposed computer will have to have about two orders of magnitude greater core size and perform arithmetic operations and access core about two orders of magnitude faster than CDC 7600 class computers. The mainframe should support remote batch and possibly time-sharing terminals. Rapid data transfers and good communications between remote terminals and the mainframe is imperative.

INTRODUCTION

An objective at Lockheed in the development of better aerodynamic design and analysis procedures is to achieve a balance between experimental and theoretical work so that for a new aircraft the development costs, time, and design risk are minimized, while aircraft performance is maximized. Until recently, most configuration aerodynamic development both at Lockheed and elsewhere was done in wind tunnels. However, theoretical methods are now available which permit both a reduction in the number of wind tunnel test hours, and a better understanding and use of wind tunnel data. Test hours and costs are reduced by eliminating many candidate configurations through the use of theoretical methods prior to testing, by permitting better test planning, and by using theoretical methods to interpolate test data and thus reducing the size of test matrices. The required test data can be better understood by using theoretical methods to gain insight into flow details that manifest themselves in gross aerodynamic parameters, and to take into account wind tunnel related phenomena such as wall and scale effects. On the other hand, carefully planned and conducted wind tunnel tests can be used to both validate new theoretical methods, and to provide data for the formulation of needed empirical models.

The availability of new theoretical methods is a result of dramatic improvements in computational aerodynamic procedures together with the advent of relatively large and fast digital computers. Although these new methods permit the treatment of many problems that were previously intractable, significant improvements in both computational techniques and computer capabilities are needed before theoretical methods can be used efficiently as engineering tools in complete configuration design. Rapid advances in the area of improved solution algorithms are now being made. These new methods cannot be fully developed nor can they be used in engineering codes until bigger and faster computers are available.

A large, centralized computer complex, referred to as the Numerical Aerodynamic Simulation Facility (NASF), has been proposed by the Ames Research Center, NASA, to provide the needed computer capabilities. This paper will define the Lockheed-Georgia Company's position on the NASF and the companion computational methods, the best way to provide the capabilities, and the procedure for making them useful to industry. The position will be arrived at by reviewing the current status and applications of computational aerodynamic methods at Lockheed-Georgia, and using these experiences to identify our projected needs. Since subsonic and transonic aircraft are designed at Lockheed-Georgia, the discussion will be restricted to methods applicable to these flight regimes. However, many numerical techniques are valid at any Mach number; therefore, much of the discussions will be of general applicability. Also, since finite difference and/or finite volume techniques applicable to non-linear equations offer the most versatility and place the greatest demands on computer capabilities, the discussions will focus on these types of methods.

CURRENT PROCEDURE

STATUS OF METHODS

The use of new computational methods has increased dramatically at Lockheed in recent years. This increased utilization is demonstrated in Table 1 where the theoretical methods now in use at Lockheed are compared with past procedures. Also shown here is the anticipated need for solutions of the three-dimensional time-averaged Navier-Stokes equations in an advanced aircraft design process. The methods are listed in order of increasing complexity, and the attendant computer programs are categorized as production, development, or research codes. A production code is one which solves a proven formulation, using proven techniques, and operates efficiently on current computers; the code is an engineering design or analysis tool. A development code treats a proven formulation, but techniques are being developed to improve its efficiency and/or accuracy, and it taxes the capabilities of present day computers. A research code is a program which is used to develop either formulations or solution techniques, and which is unproven. In general, research codes cannot be used on a day-to-day basis on existing computers.

TABLE 1 TRANSONIC TRANSPORT DESIGN METHODS CHRONOLOGY

METHODOLOGY	C-141 1960	C-5 1965	ATA #1 1975	ATA #2 1977	CXX 1985
PRODUCTION					
Airfoil Section Data Banks	x	x			
Lifting Surface Theories	x	x	x	x	x
2-D Viscous Airfoil Methods		x	x	x	x
3-D Subsonic Panel Methods			x	x	x
2-D Transonic Weak Interaction				x	x
DEVELOPMENT					
3-D Boundary Layer Methods				x	x
3-D Inviscid Transonics				x	x
3-D Viscous Weak Interactions				x	x
2-D & 3-D Aerodynamic Optimization				x	x
RESEARCH					
2-D Navier-Stokes					x
3-D Navier-Stokes					x

The philosophy of the current design approach is to use the simplest method to do a specific task, and to rely on proven, correlated methods. At present, production codes are in general easy and economical to use, and they yield accurate predictions of vehicle aerodynamics. For example, two-dimensional viscous transonic flow calculations can be done in less than one minute computation time on a CDC 7600 class computer using only a small fraction of available core.

Some production codes can, however, be expensive to use. For example, a symmetric airplane analysis using 3-D inviscid subsonic panel methods at one Mach number requires around 2,000 panels and uses about one hour computation time and all available core on a CDC 7600 computer. The cost of such a run makes the avoidance of input errors imperative. Also considerable engineering manhours can be required to prepare program input even when automated lofting techniques are used.

Three-dimensional inviscid transonic methods, 3-D viscous subsonic techniques, and optimization procedures are examples of development codes. A 3-D transonic code might use about 100,000 grid points, and a single point (Mach number, angle of attack) solution would require about 10 minutes computation time and use about 200,000 (decimal) 60-bit words of core. Despite these demands, 3-D transonic methods are being used often at Lockheed and they are proving to be useful analysis methods.

Numerical optimization is a technique, classified here as development coding, which permits the use of any aerodynamic analysis method to design configurations that are in some sense optimized for specific flight conditions. In this approach, the chosen aerodynamic method is used repeatedly to generate solutions for computing gradients and search information. Consequently, the feasibility of numerical optimization is dependent on the efficiency of the aerodynamic module. Lockheed is doing both two- and three-dimensional transonic optimization. Much of this work is being done in cooperation with NASA-Ames. Results to date are encouraging. In one airfoil design study engineering hours were reduced by a factor of four from previously used design procedures. Also, wind tunnel verification studies show the optimized airfoil to out-perform the conventionally designed section. On the other hand, 3-D optimization, although yielding encouraging results, is hampered by large computing costs and poor job turn-around time resulting from the large core requirements and run time. Nevertheless, because of its versatility, numerical optimization will undoubtedly be an important design tool in the future.

Methods for solving the time-averaged Navier-Stokes equations are now being developed. The computer programs are considered to be research codes because both the formulations and solution techniques are constantly changing. In general, they cannot be run on a day-to-day basis on available computers. Lockheed is, however, involved in research concerned with applications of such codes. The approach we have chosen is to work in cooperative programs with government agencies. In these programs, the basic solution algorithms are developed primarily within NASA. Lockheed works with NASA in the application of the algorithms to specific problems, and in correlating the theoretical solutions. This approach has proven to be successful in the evolution of 3-D transonic methods and we believe it has been mutually beneficial to Lockheed and NASA. We also think that methods for the NASF will be developed through similar combined government, industry, and university programs.

CURRENT COMPUTERS

A company-owned Univac 1100/11 computer is used at Lockheed-Georgia for the majority of scientific computations. This computer capability is augmented by remote batch terminals providing access to CDC 6600 and 7600 computers on the CDC CYBER Network. The terminals also provide a means for accessing NASA and other government computers. The dual computer system has proven to be a viable approach for providing needed computer capability at Lockheed-Georgia. Most production codes listed in Table 1 are used efficiently on the company-owned U-1100/11. Some production work and most development programming is done on the CDC 7600. With this approach, the U-1100/11 is heavily used, and the outside CDC computers are employed only when needed. On the other hand, current Lockheed-Georgia computer usage does not warrant the acquisition of a CDC 7600 with its attendant higher operating costs.

The use of off-site computers has not proven to be a problem. Operation of the remote batch terminals is straight-forward. They are capable of reading and transmitting information at the rate of about 400 cards per minute, and the printer operates at up to 500 lines per minute. Although the system is I/O bound, it provides sufficient job turn-around for the applications here. The very good communications between terminal and mainframe has proven to be indispensable in operating the remote batch terminals.

PROJECTED REQUIREMENTS

A large, centralized computer, referred to as the Numerical Aerodynamic Simulation Facility (NASF), has been proposed by the Ames Research Center, NASA to fulfill the need for increased computational capability required to develop and use advanced aerodynamic methods. The ultimate purpose of the NASF should be to provide a tool for designing better aerospace vehicles while at the same time reducing development costs. To be useful in industry applications, the facility must:

1. Provide solutions sufficiently accurate for engineering applications.
2. Be easy to use.
3. Be relatively inexpensive.
4. Be easily accessed and provide reasonable job turn-around.
5. Provide appropriate data security.

The methodologies and computer capabilities needed to attain this purpose are forecast in this section.

METHODS

Experience has shown that the simplest method which yields sufficiently accurate engineering predictions of vehicle aerodynamics will be favored over a more exact theory which is more difficult and costly to use. Therefore, a variety of aerodynamic methods will probably be used in 1980-1990 design processes. The types of methods which we think will be needed and which impose computational requirements exceeding current capabilities are listed below:

1. 3-D Nonlinear Potential Flow with Weak B.L. Interactions
2. 3-D Nonlinear Potential Flow with Patched N/S Solutions
3. Numerical Optimization with 1 and 2
4. Complete N/S Analysis

Because of the interest in active control systems and because of the inherent unsteadiness of transonic flows, formulations of these methods applicable to both steady and unsteady flows will be needed.

Three-dimensional subsonic panel methods with weak interactions have been excluded from the list despite the fact that they are useful methods which tax current computers. They have been excluded because their formulation requires the solution of dense diagonally dominant matrices while either tri-diagonal or block tri-diagonal matrices must be solved in finite-difference or finite volume techniques. Hopefully, increases in computational speed may result from developing a specialized processor for tri-diagonal matrix solution. This possible improvement in computational speed, together with anticipated refinements in finite difference equation solution algorithms, may result in computation times for finite difference solutions which are about the same as that required for panel methods. If this proves to be the case, then although panel methods may be used in special cases on the NASF, most complete configuration aerodynamic design and analysis will be done using finite difference or volume methods.

Another reason for emphasizing 3-D nonlinear potential flow methods instead of panel techniques is that they are of course applicable at any Mach number for which the flow is nearly isentropic; specifically, they can be used to compute transonic flow fields. Accurate transonic aerodynamic predictions will be needed in the 1980-1990 time period because efficient transonic performance will continue to be an important design requirement for both transport and fighter-type aircraft. Both extended small disturbance (ESD) and full potential equation (FPE) transonic methods will probably be used in the transonic design process. The advantage of ESD methods are that they are exceedingly easy to use and they require less computation time than FPE methods. ESD methods are also more easily combined with boundary layer methods. However, FPE methods, of course, yield more accurate solutions, especially near the wing leading edge.

Weak interaction solutions will be used in the envisioned design process because they can be expected to yield sufficiently accurate cruise aerodynamics for well-designed configurations, and they should be easier and more economical to use than complete N/S methods. That the weak interaction solutions should be more economical is a consequence of the need to use fewer grid points, to store fewer dependent variables, and the use of simpler equations in viscous regions. Viscous flows which cannot be modeled using Prandtl's boundary layer hypothesis can be treated using patched N/S methods. Examples of such flows are wing-body-fillet flows and shock wave/boundary layer interactions. Although the viscous flow equations are more complex than the boundary layer equations, the patched solution still offers a means to reduce the size of computational grids and thus using less core and computation time than would be needed for a comparable complete N/S solution.

Numerical optimization is a powerful new technique (at least in detailed aerodynamic design) which will be used often in the future. Basically, the technique is the theoretical equivalent of a parametric wind tunnel test in which various configurations are investigated. However, the design space from which the configuration can be investigated is much larger in computational optimization. Numerical optimization with reliable, proven theoretical methods should be much faster than wind tunnel configuration development. The cost of numerical optimization is dependent upon the efficiency of the aerodynamic analysis module. Since the usual design objective is to develop configurations with well-behaved, attached flows, weak interaction solutions and not N/S methods will probably be used.

The prediction of complete configuration aerodynamics including the presence of embedded separated turbulent flows requires the solution of the time-averaged Navier-Stokes equations. If efficient solution methods become available, and if problems such as turbulence modeling are overcome, then much aerodynamic research and configuration development that up until now was conducted in wind tunnels can be performed on computers.

To date, the most successful method for solving the N/S equations has been the time-splitting finite-difference technique developed at NASA-Ames for the time-dependent form of those equations. Dramatic reductions in run times have recently been reported. Although most research is now concerned with two-dimensional flows, continued improvement in the time-dependent approach, including extensions to three-dimensions can be anticipated. However, recalling that early two-dimensional inviscid transonic methods were ultimately replaced for the most part by relaxation solutions of steady state equations, an examination of relaxation methods applicable to the N/S equations would seem to be likely in the future. If this proves to be the case, then recent progress in using alternating direction implicit techniques to solve nonlinear problems indicates that such techniques will probably be useful in solving the N/S equations.

The finite volume (or element) method is also receiving considerable attention nowadays, although it has not yet been applied to the N/S equations. Difficulties have been encountered in applying the finite volume method to discontinuous flows. However, the flexibility that the method offers in applying boundary conditions still makes it an attractive solution technique for 3-D flows about arbitrary configurations. Consequently, the continued development of the finite volume method is warranted, and a successful application of the method to discontinuous flows is likely by the early 1980's.

The availability of a finite volume method applicable to arbitrary three-dimensional flows would greatly simplify the problem of grid generation since the method permits the use of more or less arbitrarily arranged control volumes. In fact, configuration lofting methods currently used to define complex configurations for linear subsonic and supersonic surface singularity solution techniques could probably be modified to provide surface boundary conditions for the finite volume method, and to serve as a basis for erecting the control volumes throughout the flow field.

In the projected design approach, shown schematically in Figure 1, simpler methods such as 3-D potential flow techniques with weak boundary layer interactions together with numerical optimization will be used to eliminate many possible configurations. The most promising configurations would then be analyzed in detail using the N/S solvers in much the same way a final configuration development is now done in wind tunnels. The design approach will achieve the goal of providing accurate engineering solutions. With appropriate automatic grid generation and configuration lofting routines, the methods should be easy to use.

COMPUTER

A new computer system is needed to achieve the stated goals of providing a relatively inexpensive, easily used, and easily accessed tool which can yield solutions to the already defined methods with a reasonable job turn-around. Researchers actively engaged in developing fundamental solution algorithms can better define the detailed machine architecture requirements. However, our

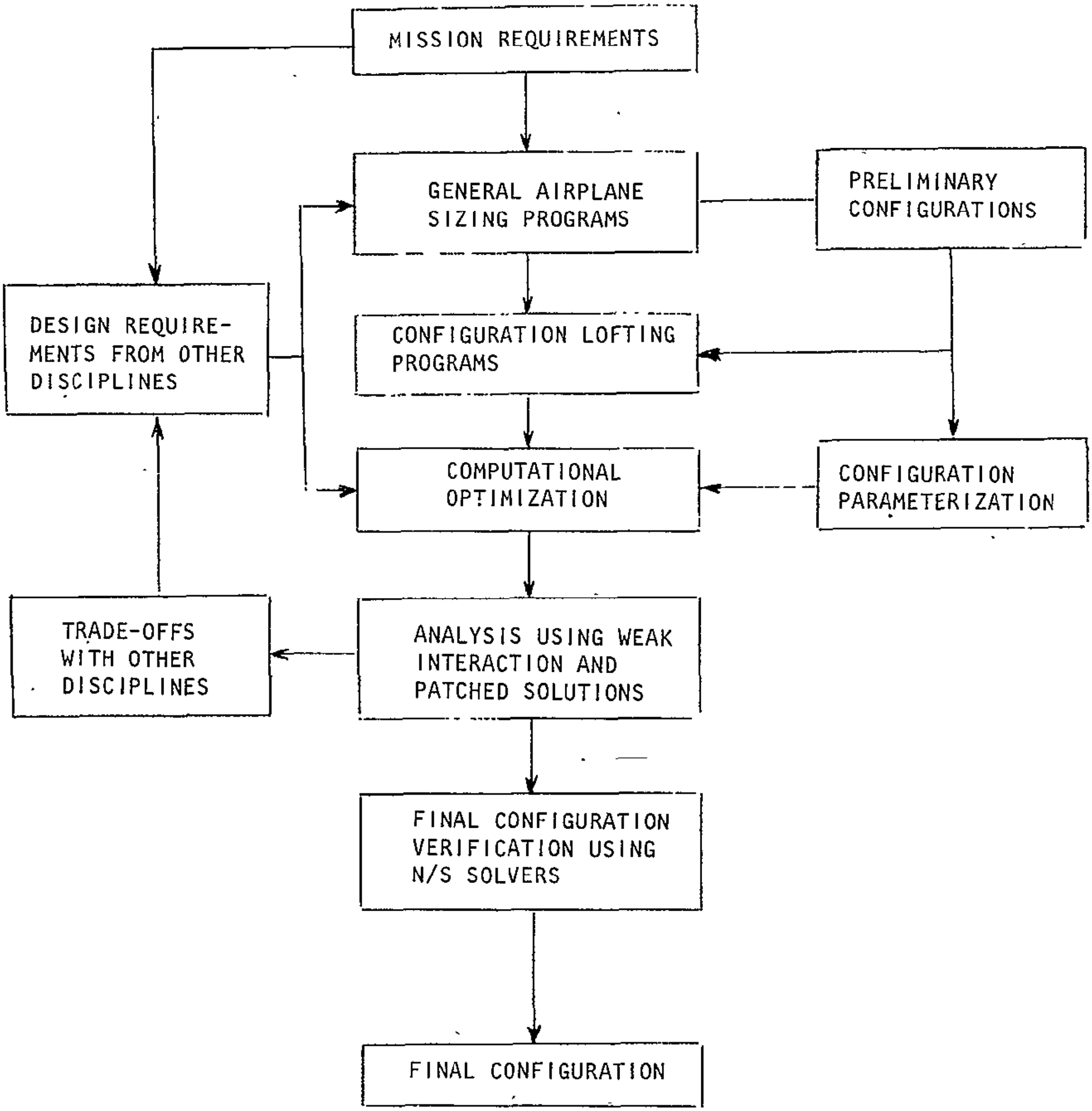

MISSION REQUIREMENTS
GENERAL AIRPLANE SIZING PROGRAMS
PRELIMINARY CONFIGURATIONS
DESIGN REQUIREMENTS FROM OTHER DISCIPLINES
CONFIGURATION LOFTING PROGRAMS
COMPUTATIONAL OPTIMIZATION
CONFIGURATION PARAMETERIZATION
TRADE-OFFS WITH OTHER DISCIPLINES
ANALYSIS USING WEAK INTERACTION AND PATCHED SOLUTIONS
FINAL CONFIGURATION VERIFICATION USING N/S SOLVERS
FINAL CONFIGURATION

FIGURE 1. PROJECTED AIRCRAFT DESIGN PROCESS

experience at Lockheed-Georgia using new theoretical methods on both on-site and remotely accessed computer systems enables us to provide some general suggestions.

Since the solution of the N/S equations places the most stringent computational demands on the proposed computer, the computer should be sized to treat these problems. Also, since these computations will be the most costly, the computer should be designed to solve N/S equations algorithms efficiently, at the cost of less efficient treatment of other problems.

An estimate of the capabilities required of an advanced computer can be made by extrapolating the computer requirements of numerical methods that are now practical. Three-dimensional inviscid transonic flows can be solved more or less routinely on the CDC 7600. Approximately 100,000 grid points are used in 3-D transonic programs to compute the flow field for a configuration in symmetric flight. Significantly more grid points will be needed to solve the 3-D N/S equations and to analyze yawed aircraft. This need arises because many grid points must be concentrated in the viscous layer adjacent to the body. For example, on the order of 20 to 40 grid points normal to the surface are used in current finite difference boundary layer programs, compared to about 50 total vertical grid points in a 3-D transonic code. If so-called large eddies which are typically smaller than 10% of the boundary layer thickness are to be computed numerically as an integral part of the solution, then many more grid points will be needed through the viscous layer. This very fine grid resolution normal to the surface dictates the use of fine grids in the other coordinate directions to keep the grid aspect ratio within acceptable limits. Consequently, at least an order of magnitude more grid points will probably be needed to solve the N/S equations than are currently used to solve 3-D inviscid transonic flows. Since at least five flow variables (instead of a single potential function) must be stored at each grid point, a need for at least a two order-of-magnitude increase in core storage is easily envisioned. A corresponding increase in computation rates and memory transfer rate will also be required to handle the increased number of computations. The following table compares the CDC 7600 with the projected computer:

	7600	Projected
Memory Words	577K	60,000K
Memory Access, Words/Sec.	36×10^6	$4,000 \times 10^6$
Floating Point Operations/Sec.	4×10^6	400×10^6

The achievement of the projected goals seems to dictate the development of a specialized computer, since it is difficult to foresee an evolution of an existing computer design providing the needed capabilities.

Although NASA, as in the past, can be expected to do most of the research associated with the development of solution algorithms, past experience indicates that some parallel research will be done by industry and in universities. Also, industry aerodynamicists are perhaps in the best position to provide the coding to make NASA-developed algorithms into production programs. In particular, configuration definition procedures and data output formats can probably best be defined in cooperation with industry. Therefore, the computer should handle many remote terminals efficiently. The capability of transferring data to and from other computers needs to be provided. Because of the significant amount of input and output associated wtih 3-D viscous flow calculations and because of the exceedingly fast computation speed envisioned for the computer, special attention should be paid to providing rapid, diversified, and efficient input and output. For example, I/O rates should be at least as large as a CDC-734 remote batch terminal operating at full capacity on a 9600 BAUD line.

New computational methods tend to increase rather than decrease the need for understanding the physics of fluid flows. Consequently, competent fluid dynamicists will be involved in developing and applying methods. To facilitate their task of communicating with programming personnel, the computer coding language should be as simple as possible and preferably similar to the familiar FORTRAN symbolic language.

Much of the configuration definition and aerodynamic data used on the NASF will be classified either for national security or industry proprietary reasons. A fail-safe data file acquisition approval system must therefore be devised. The problem of transferring classified data to and from remote terminals also needs to be resolved.

CONCLUDING COMMENTS

A commitment to improve computational capabilities and to use new theoretical methods to design more efficient aircraft has been made at the Lockheed-

Georgia Company. The availability of new methods together with a computer system like the proposed Numerical Aerodynamic Simulation Facility would permit the attainment of a balanced experimental and theoretical design process. Plans to develop such a centralized facility are supported by Lockheed because it will provide industry with access to needed computer capabilities without burdening the Company with an on-site computer which is too large and too expensive for our day-to-day needs.

Continued discourse between government, university, and industry personnel involved in developing and applying new theoretical aerodynamic methods is needed to ensure the development of a facility that is useful to all parties. The Lockheed-Georgia Company is interested in participating in future discussions.

COMPUTATIONAL AERODYNAMICS REQUIREMENTS IN CONJUNCTION WITH EXPERIMENTAL FACILITIES*

J. Leith Potter and John C. Adams
ARO, Inc., AEDC Division
A Sverdrup Corporation Company
von Karman Gas Dynamics Facility
Arnold Air Force Station, Tennessee

ABSTRACT

Examples are presented to show how computational aerodynamics has an important role in improving quality and efficiency in production of information at a wind tunnel test center. Some principal applications of the calculations are (1) to extend or clarify the understanding of experimental data, particularly when wind tunnel or scaling limitations prevent attainment of all conditions of interest and (2) for furnishing on-line or near-on-line math-model results or other comparative data needed for test direction. Significant computational abilities are needed for these purposes.

*The research reported herein was conducted by Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC), U.S. Air Force. Research results were obtained by personnel of ARO, Inc., contract operator of AEDC. Further reproduction is authorized to satisfy needs of the U.S. Government.

INTRODUCTION

Computational aerodynamics, viewed in terms of manpower engaged, is at present only a small fraction of the experimental activity at the Arnold Engineering Development Center. However, there is frequent use of computed, theoretical results in conjunction with experimental research, development, testing, and evaluation (RDT&E). Typically, the purpose of the computations is one of the following:

a. To confirm or support an otherwise uncertain conclusion, based on experiment, e.g., the attainment of fully developed turbulent boundary-layer conditions in a test or the trend of center of pressure location with changing Mach or Reynolds number.

b. To extend experimental results to the full-scale flight conditions so that wind tunnel and flight data may be compared. For example, to evaluate imperfect scaling, e.g., wall temperature ratio, Mach or Reynolds numbers, or ablation effects.

c. To evaluate the influence of tunnel free-stream conditions when, e.g., wall interference, flow nonuniformity, or real gas effects are suspected.

d. To produce math-model results on line during a test, e.g., to yield comparative data or to give trajectory points of a "dropped" store.

Owing to present limitations on computational capabilities, such supplemental calculations normally are made for simplified configurations which nevertheless serve to establish useful baselines for the engineer. Many other computations are made in the process of reducing and manipulating data, but that subject is not dealt with here.

The Arnold Engineering Development Center (AEDC) mission, as quoted from an official document, is:

> "...to support the development of aerospace systems by testing hardware in aerodynamic, propulsion, and space environmental ground test facilities that simulate flight conditions; and develop advanced techniques, instrumentation, and facilities through performing research and supplying new techniques."

Experience shows that fulfillment of this mission in a cost-effective and responsible manner often is aided by recourse to computed theoretical results when circumstances and abilities coincide. As computational abilities grow, circumstances wherein computed data are both useful and obtainable will, no doubt, arise more often. Providing needed aerodynamics information to users in the most efficient manner is expected to require both experimental and computational facilities. Some general examples are given to indicate how the two approaches are complementary and necessary.

EXAMPLES OF THE NEED FOR COMPUTATIONAL SUPPORT TO WIND TUNNEL TESTING

An exhaustive listing of examples is not appropriate to this brief presentation. Rather, it is intended to furnish a reminder of the fact that wind tunnels sometimes do not directly provide all the information needed by designers, and that advanced computational capability can represent a worthwhile addition to the resources of a test-oriented center. It is also apparent that computational aerodynamics will not soon overcome all obstacles to acquisition of all the information one might wish to obtain in laboratory facilities.

Deficiencies in wind tunnel flow conditions may take the form of, e.g., (1) inadequate Reynolds number; (2) "wet" flow; (3) nonuniform flow; (4) support interference; (5) wall effects; (6) inability to create high-enthalpy, real-gas flows; (7) excessive turbulence and noise; or (8) unrealistic fluid temperature. The use of computed results to estimate or extrapolate measured data to Reynolds numbers beyond the range available in the wind tunnel is commonplace. The same may be said for the evaluation of items (3), (5), (6), and (8) in the above list. Calculations are less conclusive but nonetheless of some use in analyzing the remaining items listed. Some examples will be given in the next section.

Pitfalls in modeling may result from the impracticality of duplicating surface roughness, waviness, gaps in structure, ablation, propulsion unit inlet and exhaust flows, airframe flexibility, or wall temperature ratio. All the above represent possible causes for error when model data are extrapolated to flight. Computational aerodynamics offers less aid in these cases, but all of the factors named can be evaluated to varying degrees of approximation by computations.

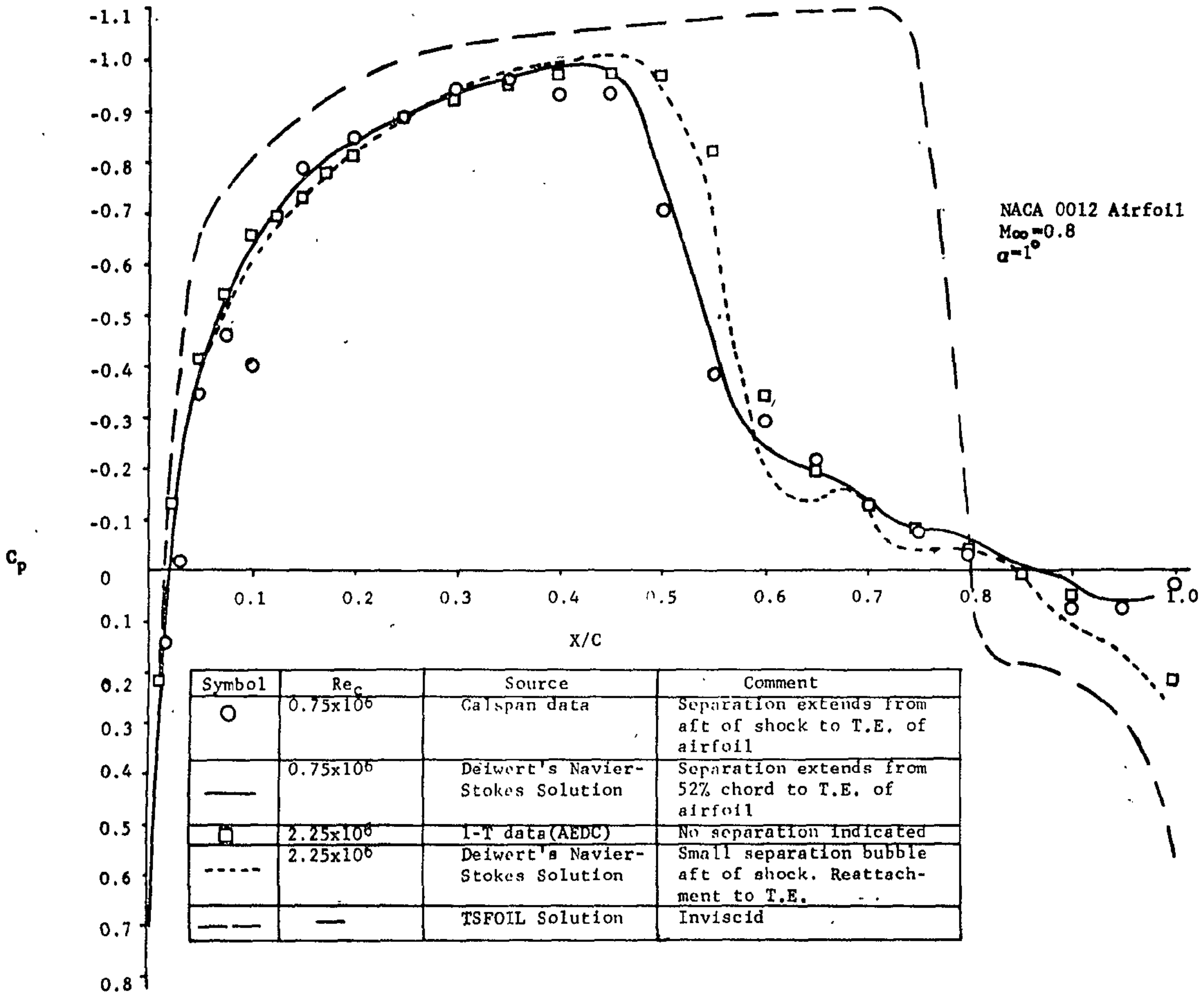
A particularly troublesome error in extrapolation to flight conditions may come about when uncertainty exists as to boundary-layer transition locations in either or both model and full-scale cases. For example, it is easily understood that differences in transition location will cause differences in local and total skin friction drag and heat transfer rate. Not only may skin friction drag account for a significant fraction of total drag, but the state of the boundary layer (i.e., laminar, transitional, or turbulent) often dictates the location or nature of flow separation and where local shock waves occur. The latter is especially important under transonic drag-rise conditions. It may also influence base pressure and, therefore, base drag. Effects of transition on the flow around a body can be very complex and a major cause of discrepancies between wind tunnel and flight results. The reasons for failure to reproduce flight or full-scale boundary-layer transition under wind tunnel conditions are at least partially understood, but the remedy is often out of reach because the flight case cannot be predicted at the time of the wind tunnel testing. If it were possible, then transition could be fixed at the same scaled position on the wind tunnel model. Even that situation would not be wholly satisfactory because tripping the boundary layer can result in a thickness that is not properly scaled due to a false increase of thickness compared to natural transition.

It is important to remember that transition usually is influenced by a combination of factors and that the roles sometimes are interrelated. Furthermore, the influences are not always qualitatively the same, e.g., reducing wall temperature ratio (T_w/T_{aw}) seems to delay transition when $T_w/T_{aw} \gtrsim 1/2$ and encourage it when $T_w/T_{aw} \lesssim 1/2$. Because of the number of factors and their complex interaction, it is generally the case that one cannot predict where transition will occur in the tunnel or in flight on arbitrary bodies. Periodically, a method for predicting transition is proposed, but none have proved adequate under general conditions yet. Therefore, computations cannot be relied on for the actual prediction of transition location on an airframe; they can only be used for parametric "what-if" studies. Progress in this long-standing wind tunnel problem probably will require both experimentation and analysis of high order. Thus far, computational approaches have entailed assumed flow models which were designed to yield transition-like results which matched some set of experimental data.

EXAMPLES OF CURRENT UTILIZATION OF ADVANCED COMPUTATIONAL CAPABILITY

To demonstrate more clearly the advantages of computational support to wind tunnel testing, we will show two representative examples of recent work at the AEDC.

The adaptive wall concept relative to interference-free transonic wind tunnel testing is an area of great current interest, both at AEDC and other testing centers. Recent experimental measurements of the upper surface pressure distribution were made on an NACA 0012 airfoil at a freestream Mach number (M_∞) of 0.80 and 1.0-deg angle of attack in the AEDC/PWT 1-ft transonic wind tunnel using an adaptive wall. Results did not agree with supposedly wall-interference-free data taken in the Calspan 8-ft transonic wind tunnel with respect to either shock location or trailing-edge pressure, as can be seen in Fig. 1. Note that the Calspan data correspond to a lower chord Reynolds number (Re_c) than the AEDC/PWT data by a factor of three. In order to better understand the aerodynamic impact of this mismatch in Re_c, numerical calculations for turbulent transonic flow based on the time-dependent Navier-Stokes equations in conjunction with an eddy viscosity model of turbulence were performed using uniform freestream boundary conditions for each of the two different Re_c conditions. The solution corresponding to the Calspan data indicated that the flow was entirely separated from the 52-percent chord location to the trailing edge of the airfoil, whereas there was less flow separation shown by the higher Re_c calculation corresponding to the AEDC/PWT data. This separation region for the Calspan flow condition displaced the shock forward relative to the higher Re_c AEDC/PWT condition, and also produced a trailing edge pressure plateau not indicated by the AEDC/PWT data or calculation. It is also important to note that the inviscid transonic small disturbance theory calculation shown on Fig. 1 is in substantial disagreement with the viscous Navier-Stokes calculations and



Symbol	Re_c	Source	Comment
○	0.75×10^6	Calspan data	Separation extends from aft of shock to T.E. of airfoil
——	0.75×10^6	Deiwert's Navier-Stokes Solution	Separation extends from 52% chord to T.E. of airfoil
□	2.25×10^6	I-T data(AEDC)	No separation indicated
- - - -	2.25×10^6	Deiwert's Navier-Stokes Solution	Small separation bubble aft of shock. Reattachment to T.E.
— —	—	TSFOIL Solution	Inviscid

Fig. 1 Upper Surface Pressure Distribution vs Non-Dimensional Chord

the experimental data. This served to strongly emphasize the often dominant role of viscous effects on transonic airfoil flows. The ability to examine these experimental data in the light of theoretical calculations obviously was of much value.

One of the most frequent AEDC/VKF applications of analytical techniques is in verification and understanding of turbulent boundary-layer flows produced in hypersonic wind tunnel tests where the boundary layer has been "tripped" in some manner. It is generally required to use relatively large trips to achieve transition in hypersonic wind tunnels, and that raises questions about unwanted flow disturbances.

Presented in Fig. 2 are typical results for centerline heat transfer distributions (in terms of the Stanton number, St_∞) on the Phase B McDonnell-Douglas Delta Wing Orbiter at 50.0-deg angle of attack with a "tripped" turbulent boundary layer. The effects of change in the freestream unit Reynolds number (Re_∞/ft) at an essentially constant freestream Mach number (M_∞) and wall temperature ratio ($T_w/T_{o,\infty}$) can be seen from the two AEDC/VKF Hypervelocity Wind Tunnel F results for different Re_∞ on Fig. 2. Wall temperature effects on turbulent boundary-layer heat transfer as reflected in the Stanton number may be seen by comparison of the AEDC/VKF Tunnel B results with the Tunnel F results at a time of 135 msec. Note that Re_∞/ft is about the same for these two flows, with a slight mismatch in M_∞; wall temperature ratio is the primary difference ($T_w/T_{o,\infty}$ = 0.64 in Tunnel B and 0.20 in Tunnel F). The agreement shown in Fig. 2 between three-dimensional turbulent boundary-layer theory and experiment indicates that upstream "tripping" of the boundary layer (in this case with carborundum grit) was indeed effective. Furthermore, the use of computed results served to confirm the existence of fully-developed turbulent boundary

MDAC Orbiter at α = 50 deg

Data Symbol	AEDC VKF Tunnel	Time, msec	M_∞	Re_∞/ft	$T_w/T_{0,\infty}$	$St_{\infty, ref}$
▲	B	-	8.0	3.73×10^6	0.64	2.65×10^{-2}
■	F Run 3659	61	10.70	12.65×10^6	0.24	1.72×10^{-2}
⬢	F Run 3659	135	10.53	4.16×10^6	0.20	2.92×10^{-2}

——— Three-Dimensional Turbulent Boundary-Layer Theory

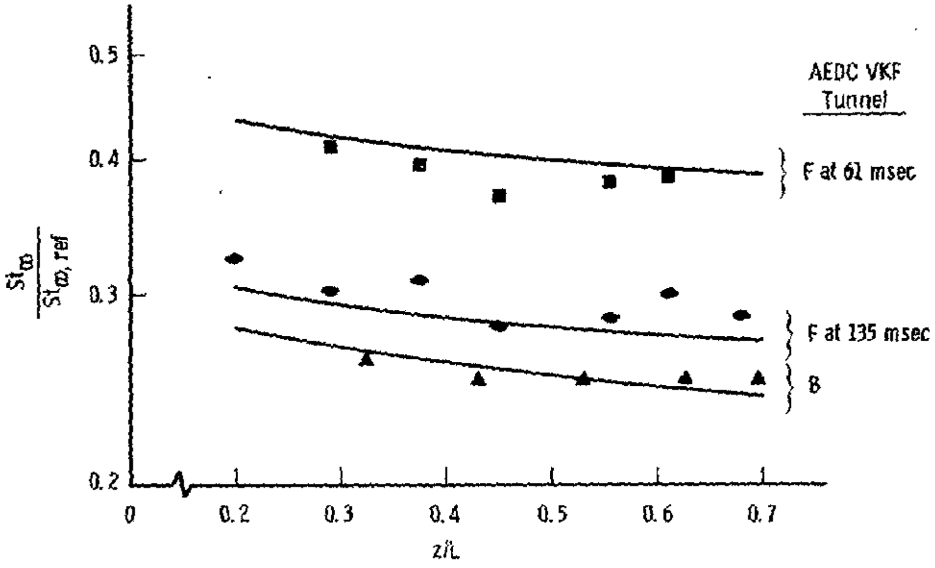


Fig. 2 Effects of Mach Number, Reynolds Number, and Wall Temperature Ratio on MDAC Orbiter Windward Centerline Turbulent Heat Transfer under High Angle-of-Attack Conditions

layer flow at all Reynolds numbers and to clarify the cause of the difference in Stanton numbers obtained from Tunnels B and F.

CONCLUDING REMARKS

Other presentations in this session have addressed future computational aerodynamics requirements for the subsonic/transonic flow regimes. Presented in Table 1 are the authors' views on some requirements for reentry vehicles and lifting bodies in the supersonic/hypersonic flow regimes. The most pressing computational need today, in our opinion, is for three-dimensional codes allowing analysis of general geometry (ablated) nose tips at incidence under both inviscid and viscous flow conditions. As an extension of this, a three-dimensional viscous shock layer code written for general body geometry and including turbulence modeling is also needed. This type of analysis has been shown to be very useful for application at high Mach number. Good general body geometry packages are currently available for both reentry vehicles and lifting bodies.

To be of value, the computational results must take into account the users' needs and merit their confidence. The wind tunnel operators have devoted years of study to tunnel-related problems in the areas of simulation and scaling and are in a good position to supplement experimental data with computations which will enhance the information acquired in the laboratory. The computational facilities needed for this service must be capable of furnishing speedy solutions of large codes so that maximum efficiency in test direction can be realized, i.e., so that decisions can be made during the course of testing instead of well afterwards.

TABLE 1

CURRENT STATUS AND FUTURE REQUIREMENTS FOR COMPUTATIONAL AERODYNAMICS APPLIED TO REENTRY VEHICLES AND LIFTING BODIES

INVISCID FLOWS

- Adequate Generalized 2-D and 3-D Codes Available for Supersonic Conditions.
- Embedded Subsonic Regions Need More Work.
- General 2-D and 3-D Blunt Nose Code Needed.

VISCOUS FLOWS

- Adequate Generalized 2-D and 3-D Boundary-Layer Codes Available.
- General 3-D Viscous Shock Layer Code Needed.
- General 2-D and 3-D Blunt Nose Navier-Stokes Code Needed.

GEOMETRY

- Adequate Generalized Codes Available.

  QUICK - Grumman
  PREQUICK - AEDC/VKF
  KWIKNØSE - AEDC/VKF

COMPUTATIONAL FLUID DYNAMICS (CFD) -- FUTURE ROLE AND REQUIREMENTS AS VIEWED BY AN APPLIED AERODYNAMICIST

H. Yoshihara
Boeing Company
Seattle, Washington

ABSTRACT

The problem of designing the wing-fuselage configuration of an advanced transonic commercial airliner and the optimization of a supercruiser fighter are sketched, pointing out the essential fluid mechanical phenomena that play an important role. Such problems suggest that for a numerical method to be useful, it must be able to treat highly three dimensional turbulent separations, flows with jet engine exhausts, and complex vehicle configurations. Weaknesses of the two principal tools of the aerodynamicist, the wind tunnel and the computer, suggest a complementing combined use of these tools, which is illustrated by the case of the transonic wing-fuselage design. The anticipated difficulties in developing an adequate turbulent transport model suggest that such an approach may have to suffice for an extended period. On a longer term, experimentation of turbulent transport in meaningful cases must be intensified to provide a data base for both modeling and theory validation purposes. Development of more powerful computers must proceed simultaneously.

The role and requirements for CFD in the near future will be sketched from the point of view of the user aerodynamicist who has the task of incorporating advanced concepts into the design of new aircraft. This will be accomplished by first describing two problems of current interest, identifying the key fluid mechanical phenomena that must be modeled. The primary weaknesses of the two principal tools of the aerodynamicist, the wind tunnel and computer, are next reviewed, thereby setting the stage for defining a meaningful role of the computer in the near future.

Consider first the near-term optimization of the next generation transonic commercial transport, several versions of which are shown in Figure 1. Here one important subtask is the determination of the wing-fuselage configuration which has the highest drag divergence Mach Number (where the drag abruptly increases) for a prescribed lift, no drag creep, and an acceptable buffet margin. Significant computational progress on this problem has been made on an inviscid framework by Jameson, but the formidable remaining obstacle is our inability to model the crucial three dimensional (3D) viscous interactions at the shock.

Another problem is the design of a new combat aircraft, the so-called supercruiser, which is required to have increased supersonic radius (for decreased vulnerability) and still be able to maneuver with agility in the transonic speed regime. The dilemma here is the incompatibility of the configurations demanded by the two requirements. Thus high supersonic radius mandates low zero-lift drag that then necessitates wings of low aspect ratio and large leading edge sweeps as shown in Figure 2. In the subsonic and transonic regimes with such a configuration it is not only difficult to generate significant loadings on the planform, but whatever loading generated is diminished by pressure leakages over the near-proximate edges of the wing. Since the supersonic performance is not to be compromised, the primary task is thus to find means to enhance the transonic high lift performance of the supercruiser configuration. One possibility is the use of leading edge separation vortices to induce increased suctions on the wing upper surface as shown in the lower part of Figure 2.

Another potential means is thrust vectoring whereby the engine exhaust is deflected downwards by means of a 2D nozzle. This generates lift, not only by the jet reaction, but also by the aft cambering effect produced by the jet plume. These devices are shown in Figure 3.

When the above aft devices are employed, a difficult problem is to balance out the resulting nose-down moment to trim the aircraft. One possibility is the use of a canard as shown in Figure 3 to provide a lift forward of the vehicle center of gravity. Such a canard is positioned to interact favorably with the wing such that the canard leading edge separation vortices pass over the wing upper surface without bursting to generate additional suction over the wing. Vortex bursting is somewhat akin to boundary layer separation wherein the tight spiraling motion degenerates into a highly disorganized turbulent motion by a still unknown mechanism. When such bursting occurs upstream of the wing as shown in Figure 3, the lift of the wing is greatly diminished.

The above two problems are not atypical of those confronted by applied aerodynamicists. Such problems involve strong viscous interactions with complex 3D separations, presence of regions of increased stagnation enthalpy as in the jet engine exhaust plume, and the need to consider complex vehicle configurations. Any contemplated prediction tool must be able to handle these complications.

Two tools available to the aerodynamicist are the wind tunnel and the computer. Although wind tunnels are reasonably reliable in the supersonic regime, they are inadequate in the transonic regime, just the regime of importance in the above two problems. A prudent engineer uses a transonic wind tunnel mainly to obtain incremental effects in a configuration study. There are numerous causes that distort transonic wind tunnel data, but the two that are difficult to assess or to eliminate are due to wall interference and the inability to model the full scale viscous interactions.

In the case of CFD the primary limitation is the inability to model the turbulent transport to the generality required to cover situations described above. Extrapolating the past and present progress of turbulent transport modeling, one cannot be optimistic of developing an adequate model to cover the extreme situations described above. One formidable obstacle is the generation of useable empirical data base on which to construct the model.

In this environment what should then be the role of CFD in the immediate future, perhaps within the next decade? At least for the immediate future, in the transonic regime, one viable procedure will be the complementary use of the computer and wind tunnel whereby the strength of one is used to supplement the weakness of the other. Here we probably must be still content not in the prediction of the performance in an absolute fashion, but in determining incremental performance differences among candidate configurations. In particular the determination of the drag to the required accuracy may still be well out of reach. The precise details of the joint use of the wind tunnel and the computer must be ad hoc, tailored to the specific problem on hand. One possibility for the simpler case of the transonic wing-fuselage design of the commercial transport will be outlined for illustrative purposes.

Consider the specific example of minimizing the drag of a wing-fuselage configuration at a a given transonic Mach number having a prescribed lift. When the flow over a prescribed configuration cannot be calculated with sufficient ease, it is difficult to carry out a formal optimization process for example as a variational procedure. A commonly used and meaningful alternative is to design the wing to achieve uniform isobars on the wing upper surface reasonably aligned with the local wing sweep. In this manner severe premature deterioration of the shock-induced losses along the span is avoided. Thus of the hierarchy of sophistication to model the viscous interaction, one of the crudest will suffice for the present application--namely, the modeling of the displacement effect of the boundary layer. This then will permit the determination of the pressure distributions and hence the isobar pattern.

The detailed steps in this approach are shown in Figure 4. Here one presupposes the availability of an exact potential code as that developed by Jameson but with a generalized mesh generation subroutine. Additionally the computer program must have the option of prescribing surface pressures in specified regions of the configuration in lieu of the shape.

In Step 1 of Figure 4 an initial configuration is designed using for example the above inviscid code possibly supplemented by a viscous displacement model generated by a previous example. The resulting configuration is then tested in the wind tunnel (Step 2) at a Reynolds number of the order of 2-4 x 10^6 per mean chord where extrapolation to the full scale Reynolds number will not produce qualitative surprises. The measurements must include pressure distributions at a sufficient number of span stations to enable a determination of the isobar pattern. Pressure measurements in the vicinity of the upper and lower walls of the wind tunnel must also be carried out. Additional runs at several values of Mach number and angle of attack in the neighborhood of the important test conditions, as at the cruise condition, must also be carried out.

In Step 3 calculations are carried out at the cruise condition where the measured pressures are now prescribed as boundary conditions in the region aft of the shock waves where the viscous displacement effects are significant. Elsewhere the original slopes are prescribed. The measured wall pressures are also prescribed to simulate the wind tunnel environment. The results then yield the viscous displacement shape where the pressures were prescribed, and the pressures where the shape was prescribed. The agreement of the latter with the measured pressures will serve as a check. The above calculations are now repeated at several of the test points about the cruise condition to enable a more reliable modeling of the viscous ramps applicable for neighboring shock configurations.

With the resulting viscous ramp model, calculations are repeated at the cruise condition, recontouring the wing in deficient regions by prescribing more desirable pressures in these regions. Here it must be remembered that due to the presence of an extended supersonic region on the wing upper surface, changing the wing contour in a given region will also affect the flow in the corresponding domain of influence. In the latter calculation the measured wall pressures are replaced by the free stream conditions, and if suitable scaling laws are available, the viscous ramps would then be scaled to full scale Reynolds number. Needless to say, a fluid mechanically experienced designer is essential in this step. After a satisfactory configuration is evolved, confirmation of the design is obtained by a final wind tunnel test. For this purpose calculations for the final configuration are performed in the wind tunnel environment by prescribing the measured wall pressures and using the proper viscous ramps.

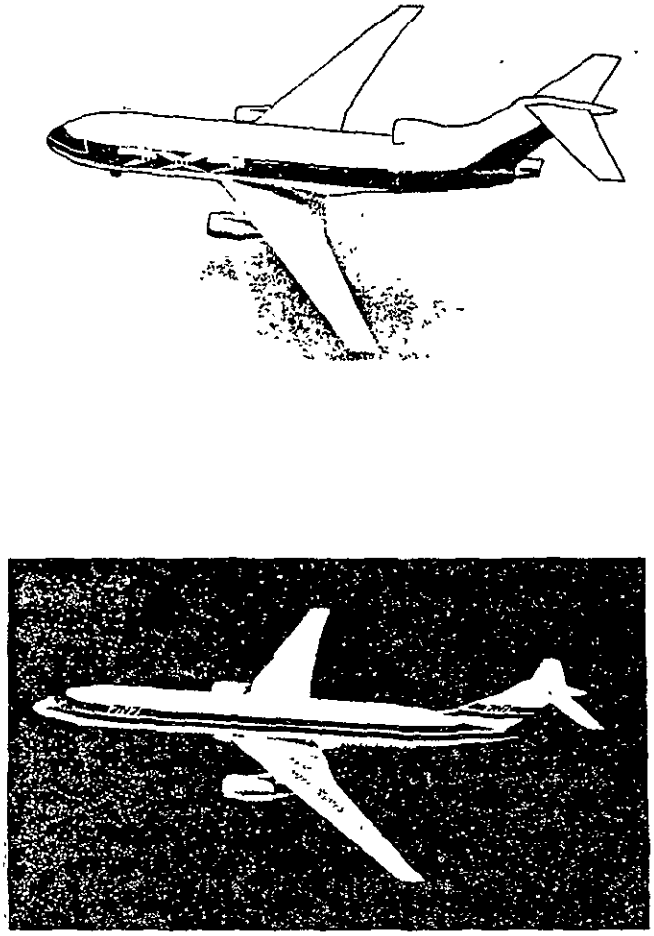
In summary, in the above simple case of the wing-fuselage design of a transonic commercial airliner, combined use of the wind tunnel and computer was suggested to model the strong viscous interaction, and the computer then used to tailor the design without wall interference. Here a crude level of modeling the viscous interaction was suggested, permitting the continued use of the inviscid equations. The resulting model should be reasonably reliable since it was applied only to cases closely neighboring the empirical data base.

The above approach was necessitated by the limitations of existing 3D boundary layer codes. Such codes cannot bridge the shock properly to yield the necessary initial conditions for the calculation of the boundary layer downstream of the shock, in particular the velocity profiles. The use of the 3D boundary layer codes, though appearing superficially to be more exact, in fact can lead to less accurate solutions. Most seriously, they cannot handle separated flows.

The present approach emphasized the near-term. What then are the longer range prospects. Clearly the dominant obstacle still remains the development of a suitable model for the turbulence in the generality required for practical problems. Such models can range from those based on molecular transport resulting in the unsteady (laminar) Navier-Stokes equations to those based on a coarser averaging. The unsteady Navier-Stokes equations require no empirical inputs, have universal applicability, but have their well known limitation in their numerical analogue as the result of truncation errors. Moreover, in this highly resolved representation, boundary conditions may not be a priori known in the required consistent manner.

particularly in the wind tunnel environment when experimental verification is sought. In the more coarsely grained representation, an experimental data base is necessary, and the generality of the latter will define the versatility of the resulting phenomenological equations. It is the result of the anticipated difficulty of generating such data base that an approach as described above combining the use of the wind tunnel and the computer might have to suffice for an extended period.

On the other hand for the long term, experimentation must be intensified, not only to seek to unravel the complexities of relevant turbulence at various time scales, but to generate a meaningful data base. The latter will be used to model phenomenologically the turbulent transport as well as to furnish a validation base for the resulting theories. Here the laser velocimeter and other non-intrusive instrumentation will play a key role. Hand in hand the development of more powerful computers must proceed with the above experimentation



BASIC OBSTACLE

MODELING OF THE
3D SHOCK-BOUNDARY LAYER
INTERACTION

- Type B Interactions
- Spanwise Contaminations on Swept Wings

From BMA Manager, August 1977, Vol. 7, No. 7

Figure 1. Design of a Transonic Commercial Airliner

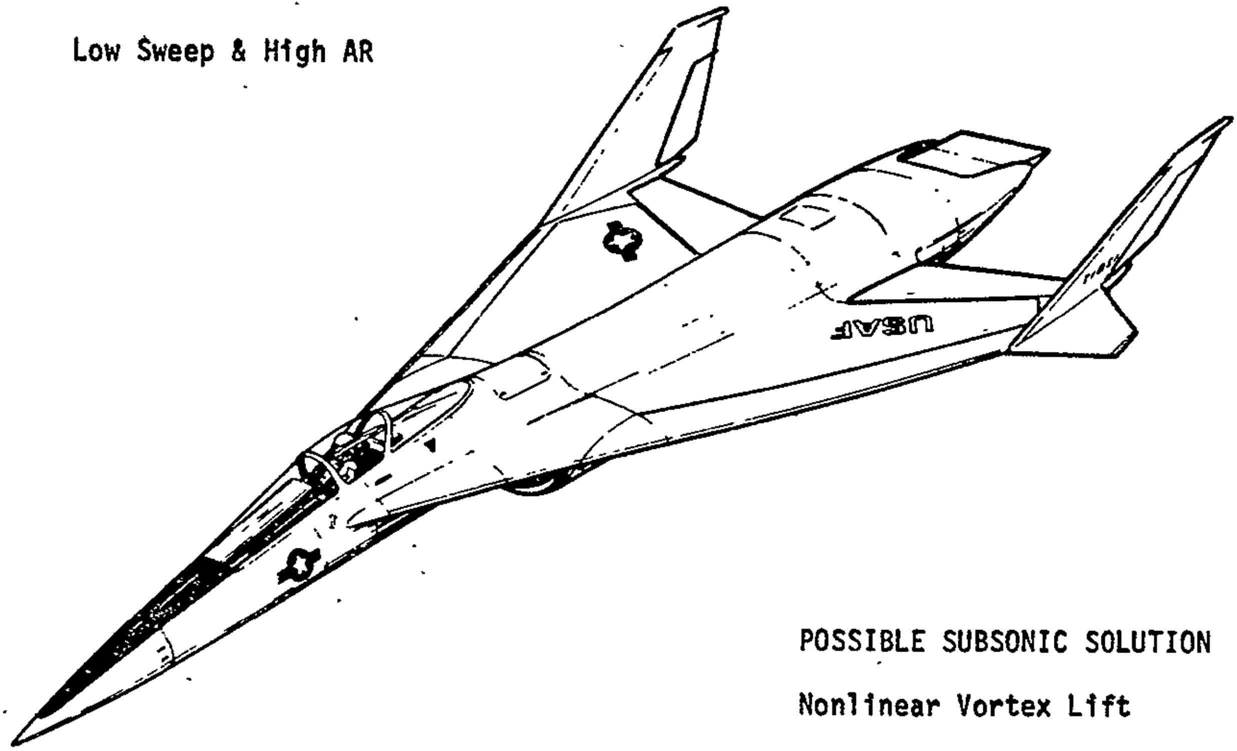
- EFFICIENT SUPERSONIC CRUISE

  Low C_{D_o} (Sleek Area Distribution)

  High L.E. Sweep - Low AR

- GOOD TRANSONIC MANEUVERABILITY & LANDING AND TAKEOFF CHARACTERISTICS

  Low Sweep & High AR



POSSIBLE SUBSONIC SOLUTION

Nonlinear Vortex Lift

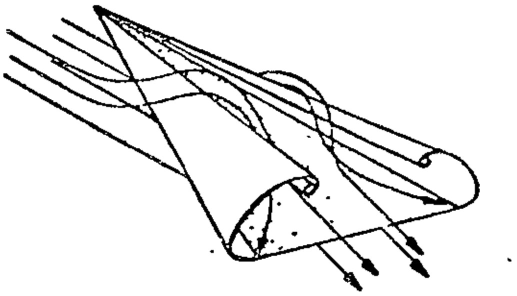


Figure 2. An Aerodynamic Dilemma - The Case of the Supercruiser

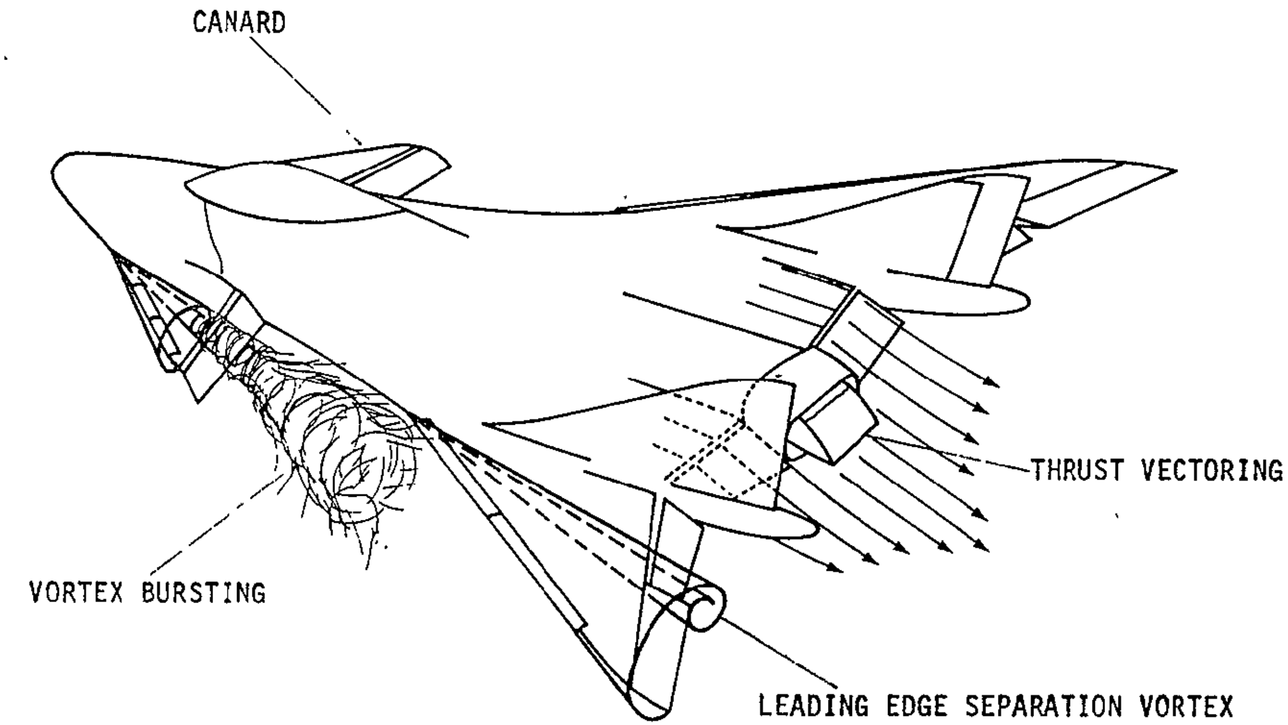


Figure 3. Transonic Lift Generation

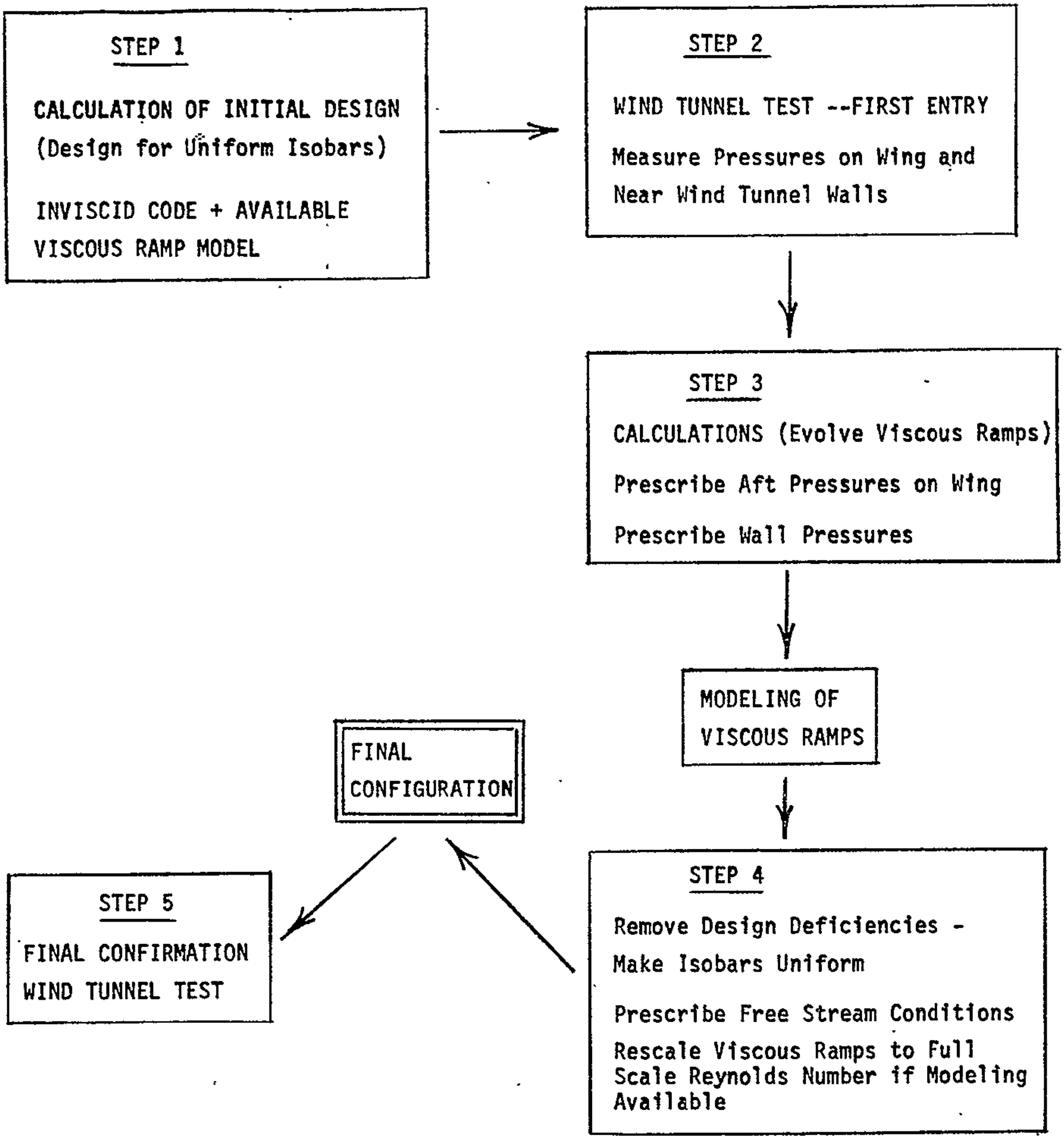

STEP 1
CALCULATION OF INITIAL DESIGN
(Design for Uniform Isobars)
INVISCID CODE + AVAILABLE
VISCOUS RAMP MODEL
STEP 2
WIND TUNNEL TEST --FIRST ENTRY
Measure Pressures on Wing and
Near Wind Tunnel Walls
STEP 3
CALCULATIONS (Evolve Viscous Ramps)
Prescribe Aft Pressures on Wing
Prescribe Wall Pressures
MODELING OF
VISCOUS RAMPS
STEP 4
Remove Design Deficiencies -
Make Isobars Uniform
Prescribe Free Stream Conditions
Rescale Viscous Ramps to Full
Scale Reynolds Number if Modeling
Available
FINAL
CONFIGURATION
STEP 5
FINAL CONFIRMATION
WIND TUNNEL TEST

Figure 4. Complementing Use of Wind Tunnel and Computer - Design of the Transonic Commercial Airliner

SESSION 5

Panel on VISCOUS FLOW SIMULATIONS

Robert W. MacCormack, Chairman

THE STATUS AND FUTURE PROSPECTS FOR VISCOUS FLOW SIMULATIONS

Robert W. MacCormack

Ames Research Center, NASA
Moffett Field, California

The Navier-Stokes equations adequately describe aerodynamic flows at standard atmospheric conditions. If we could efficiently solve these equations there would be no need for experimental tests to design flight vehicles or other aerodynamic devices. Unfortunately, at high Reynolds numbers, such as those existing at flight conditions, these equations become both mathematically and numerically stiff.

Reynolds number is a measure of the ratio of the inertial forces to the viscous forces of a fluid. The viscous terms which cause the system to be parabolic are of the order of the reciprocal of the Reynolds number. At high Reynolds number the system is almost everywhere hyperbolic; the viscous terms are negligible except in thin layers near body surfaces. Within these thin layers viscous effects are significant and control the important phenomenon of boundary layer separation. Because of the disparity in magnitude at high Reynolds number between the inertial and viscous terms and their length scales, such systems of equations are difficult to solve numerically. Although we have made much progress toward their solution, the calculation of flow fields past complete aircraft configurations at flight Reynolds numbers is far beyond our reach. They await substantial progress in developing reliable and powerful computer hardware, in devising accurate and efficient numerical methods, and in understanding and modeling the physics of turbulence.

During the past two decades rapid progress has been made in computer hardware development. Computer technology has increased computing speeds by a factor of ten approximately every five years. This has resulted in a reduction of the computation cost of a given problem by a factor of ten approximately every seven years. During the next decade it appears that this trend will continue and that computers more than two orders of magnitude faster than present machines and with memories as large as 32 million words can be built for fluid dynamics applications.

The availability of powerful computers has spurred on the development of numerical methods for solving the Navier-Stokes equations. We have witnessed during the past decade dramatic progress in computational fluid dynamics which has reduced the required computation time to solve a given problem on a given computer by one and two orders of magnitude. During the next decade we can expect that this trend will continue and that numerical methods an order of magnitude faster will be devised.

Finally, we can expect the availability of fast computers and methods to spur on the development of the third essential element -- the understanding and modeling of the physics of turbulence. Turbulent flows contain eddies that cause rapid fluctuations about the mean flow solution, which itself may also be varying in time. Because of present and foreseeable computer speed and memory limitations, the computational mesh cannot be made fine enough to resolve all significant eddy length scales. Thus, the instantaneous solution is impossible to determine. However, because mean flow quantities such as lift, drag, and heat transfer are of primary interest to aeronautical design, solutions to the Reynolds or "time averaged" Navier-Stokes equations are sought. To solve these equations, however, mesh size and small-scale turbulence effects must be accounted for by modeling. Such models exist now for compressible attached flows with mild pressure gradients and for Mach numbers as high as ten. There are no models, however, that can be applied with confidence to predict turbulence effects for flows separated by strong adverse pressure gradients. There is presently much experimental, computational, and theoretical activity toward the development of such models. During the past few years much progress has been made. We can expect much more in the next decade. Where today we can calculate some complex unsteady two and three-dimensional flows about simple but arbitrary geometries at high Reynolds numbers, perhaps a decade from now we will be routinely calculating for design purposes, in computation times measured only in minutes, flows past complete aircraft configurations at flight Reynolds numbers.

COMPUTATIONAL REQUIREMENTS FOR THREE-DIMENSIONAL FLOWS*

F. G. Blottner
Sandia Laboratories
Albuquerque, New Mexico 87115

For the prediction of steady viscous flow over complex configurations the needed computational requirements are considered. The desired predictions must be made at reasonable expense, require a reasonable amount of storage, and result in solutions that are sufficiently accurate. The information needed to estimate the cost of Navier-Stokes solutions is not available to the author and does not appear to be available. Therefore, some experience with the solution of the three-dimensional boundary layer equations will be utilized to help illustrate the needed information and what can be expected for Navier-Stokes solutions.

The cost of a computation can be estimated from the following relation:

$$C = T \cdot E \quad , \tag{1}$$

*This work was supported by the U.S. Energy Research and Development Administration.

where

T = total computation time (s),

E = expense of computer per unit time (\$/s).

The value of E appears to have remained nearly constant with time, and a value of $E = 10^{-1}$ is assumed. Also, it is assumed that a reasonable cost for a prediction is \$1000 which gives $T = 10^4$ s. Therefore, the computation time should be less than this number unless computer expenses can be sufficiently reduced. The total computation time is estimated from

$$T = N\ t/S \quad , \tag{2}$$

where

N = number of grid points - $N_x \cdot N_y \cdot N_z$,

t = time to compute one grid point on reference computer (CDC 7600)

S = machine speed relative to reference computer.

Next, it is assumed that the number of grid points in each direction is the same, which gives $N_x = N_y = N_z = n$, or $N = n^3$. The time to compute one grid point is expressed as the following:

$$t = \tau\ I \quad , \tag{3}$$

where

τ = time to compute one grid point for one time step or one iteration step on reference computer (CDC 7600),

I = number of time or iteration steps.

When the above relations are combined, the cost of a computation becomes

$$C = n^3 \tau \; I \; (E/S) \quad , \tag{4}$$

where the term in the bracket is determined from the computer being used. Perhaps this expression oversimplifies things, but hopefully it indicates the important parameters which determine the cost. The value of some of these parameters for boundary layer flows will be investigated next.

As can be seen from Eq. (4), the number of grid points required is extremely important in determining the cost of a computation. Also, one cannot state the number of grid points required until the desired accuracy of the solution is given. For incompressible, two-dimensional, turbulent boundary-layer flows the accuracy of the wall shear stress has been determined for various number of grid points by Blottner[1] and Wornom.[2] These results are given below for two desired accuracies and for second- and fourth-order schemes.

	Number of Grid Points		
	Blottner	Wornom	
Accuracy	2nd Order	2nd Order	4th Order
1.0%	25	30	8
0.1%	70	100	13

For an incompressible, laminar, three-dimensional boundary-layer calculation by Blottner,[3] the following results were obtained in the cross-flow direction for the indicated accuracy of the streamwise velocity:

Accuracy	Number of Grid Points for 2nd Order Scheme
1.0%	25
0.1%	80

For a compressible, two-dimensional, laminar boundary-layer flow with linearly retarded edge velocity, the following results are given by Blottner[3] for the accuracy of the wall shear stress for the number of grid points in the flow direction:

Accuracy	Number of Grid Points for 2nd Order Scheme
1.0%	10
0.1%	25

With the above results it is estimated that the number of grid points required for three-dimensional boundary-layer solutions is the following:

Accuracy	Number of Grid Points 2nd Order Scheme	 4th Order Scheme
1.0%	25^3	10^3
0.1%	80^3	15^3

These estimates assume that equal number of grid points can be used in each coordinate direction and the difference scheme is of the accuracy indicated in each coordinate direction. Also, it is assumed that a variable grid or coordinate transformation is utilized to obtain the desired accuracy with a minimum number of grid points.

The time to compute one grid point with various difference schemes needs to be known. The value of τ for a variety of problems and solution techniques is given in Table I. The explicit schemes are generally faster than implicit schemes but the solutions in some cases are obtained without time marching or a relaxation procedure. It appears that a value of $\tau = 10^{-3}$ s is a reasonable value for three-dimensional problems and cannot be changed too much with various numerical schemes. The important parameter is I as far as the numerical scheme is concerned. For boundary layer flows $I \approx 1$, for semi-direct methods $I \approx 10$, while time marching and relaxation procedures require $I = 10^2$ or more. Development of techniques which reduce the value of I while obtaining a steady-state solution is a worthwhile task.

With the foregoing information some estimates for the cost of performing 3-D boundary-layer computations are now made for a CDC 7600 computer. The results are the following:

3-D BOUNDARY LAYER SOLUTION

Accuracy	2nd Order Scheme Cost	2nd Order Scheme Time (s)	4th Order Scheme Cost	4th Order Scheme Time (s)
1.0%	$ 1.60	16	$0.10	1.0
0.1%	51.00	512	0.34	3.4

For the fourth-order scheme the value of τ has been assumed the same as for the second-order scheme which is too optimistic. For two-dimensional boundary layer solutions with fourth-order accuracy in the direction normal to the surface, the value of τ is increased only 10 or 20%. Since fourth-order accurate boundary layer solutions in all coordinate directions do not exist, the correct value of τ remains to be determined. If the complete Navier-Stokes equations are used to solve for the 3-D boundary layer flows, what cost would one expect? For the same accuracy of the results, the same number of grid points would be required. The main difference is in the solution procedure required for the two cases since a time marching or relaxation scheme is needed for the Navier-Stokes equations. Therefore, $I = 10^3$ is a reasonable value. The 3-D Navier-Stokes solutions could become unreasonably expensive with a second-order accurate scheme, while a fourth-order method might result in a reasonable cost as shown below:

3-D NAVIER-STOKES SOLUTION OF A BOUNDARY LAYER FLOW

Accuracy	Cost 2nd Order Scheme	Cost 4th Order Scheme
1.0%	$1,600	$100
0.1%	$51,000	$340

The cost to compute the flow field around a complete aerodynamic shape could be estimated if the cost of the various parts of the flow field are known. At the workshop the various participants should be able to help provide the various estimates needed. The total cost will probably be 10 to 100 times more expensive than the above computation. Such computations would be unreasonably expensive on present-day computers with present computational techniques. It would appear possible to solve the complete flow around aerodynamic shapes if the following items are achieved:

1. Develop higher-order accurate finite-difference schemes that can provide reasonably accurate solutions with a minimum number of grid points required. This is also a very important concern with storage requirements.

2. Develop coordinate transformations and variable grid techniques which result in the need for less grid points. Especially, multidimensional self-adaptive grid techniques are needed.

3. Determine numerical schemes that can obtain the steady-state solutions without a large number of time steps or iterations.

4. Utilize cheaper and faster computers.

If improvements can be made in each of these items, then the need for drastic improvements in any one item will not be required.

TABLE I

COMPUTATION TIME/GRID POINT/STEP
(ON CDC 7600)

PROBLEM	TIME/GRID POINT (ms)	REF.
1-D Unsteady (MacCormack Scheme)	0.64	Authòr
2-D Unsteady (MacCormack Scheme)	0.36	4
(Beam & Warming)	0.46	5
3-D Poisson Eq. (Direct Solution)	0.10	6
2-D Compressible Boundary Layer (Uncoupled)	0.16	7
2-D Incompressible Channel Flow (Coupled)	1.2	8
3-D Incompressible Boundary Layer (Uncoupled)	1.0	9
3-D Compressible Boundary Layer		
(McLean)	2.4	10
(Cebeci, et al)	0.3	11
3-D Navier-Stokes (MacCormack)	0.53	12
3-D Navier-Stokes (Briley & McDonald)	1.4	13

References

1. F. G. Blottner, "Variable Grid Scheme Applied to Turbulent Boundary Layer," Computer Methods in Applied Mechanics and Engineering, Vol. 4, pp. 179-194 (1974).

2. S. F. Wornom, "A Critical Study of Higher-Order Numerical Methods for Solving the Boundary-Layer Equations," AIAA Paper No. 77-637 (June 1977).

3. F. G. Blottner, "Computational Techniques for Boundary Layers," AGARD Lecture Series 73 (February 1975).

4. R. W. MacCormack and B. S. Baldwin, "A Numerical Method for Solving the Navier-Stokes Equations with Application to Shock-Boundary Layer Interactions," AIAA Paper 75-1, January 20-22, 1975.

5. R. M. Beam and R. F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA 3rd Computational Fluid Dynamics Conference, June 27-28, 1977.

6. U. Schumann, "Fast Solution Methods for the Discretized Poisson Equation," GAMM Workshop, April 1977.

7. F. G. Blottner, "Investigation of Some Finite-Difference Techniques for Solving the Boundary Layer Equation," Computer Methods in Applied Mechanics and Engineering, Vol. 6, pp. 1-30 (1975).

8. F. G. Blottner, "Numerical Solution of Slender Channel Laminar Flows," Computer Methods in Applied Mechanics and Engineering, Vol. 7 (1977).

9. F. G. Blottner and Molly Ellis, "Three-Dimensional, Incompressible, Boundary Layer on Blunt Bodies, Part I: Analysis and Results, Sandia Laboratories, SLA-73-0366 (April 1973).

10. J. D. McLean, "Three-Dimensional Turbulent Boundary Layer Calculations for Swept Wings," AIAA Paper No. 77-3 (January 1977).

11. T. Cebeci, K. Kaups, and J. A. Ramsey, "A General Method for Calculating Three-Dimensional Compressible Laminar and Turbulent Boundary Layers on Arbitrary Wings," NASA CR-2777 (January 1977).

12. C. M. Hung and R. W. MacCormack, "Numerical Solution of Supersonic Laminar Flow Over a Three-Dimensional Compression Corner," AIAA Paper No. 77-694 (June 1977).

13. W. R. Briley and H. McDonald, "Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, pp. 372-397 (1977).

D,13

Viscous Flow Simulations in VTOL Aerodynamics*

W. W. Bower
McDonnell Douglas Research Laboratories
St. Louis, Missouri 63166

Abstract

The critical issues in viscous flow simulations, such as boundary-layer separation, entrainment, turbulence modeling, and compressibility, are discussed with regard to the ground effects problem for vertical-takeoff-and-landing (VTOL) aircraft. A simulation of the two-dimensional incompressible lift jet in ground proximity is based on solution of the Reynolds-averaged Navier-Stokes equations in conjunction with a turbulence-model equation which are written in stream function-vorticity form and are solved using Hoffman's augmented-central-difference algorithm. The resulting equations and their shortcomings are discussed when the technique is extended to two-dimensional compressible and three-dimensional incompressible flows.

Nomenclature

a	grid spacing in ξ direction
b	grid spacing in η direction
C_D	empirical constant in turbulence model
c_p	specific heat at constant pressure normalized by $\overline{c}_{p,o}$
c_μ	empirical constant in turbulence model
$\overline{D}$	jet slot width at exit plane (used as normalizing parameter for all lengths)
F	conformal mapping function
Fr	Froude number, $\overline{V}_o/\sqrt{\overline{g}_o\overline{D}}$
H	height of jet exit plane above ground normalized by $\overline{D}$
i	$\sqrt{-1}$
k	turbulent kinetic energy normalized by $\overline{V}_o^2$; thermal conductivity normalized by $\overline{k}_o$
ℓ_D	length scale for dissipation normalized by $\overline{D}$
ℓ_μ	length scale for viscosity normalized by $\overline{D}$
p	static pressure normalized by $\overline{\rho}\overline{V}_o^2/2$
Pr	Prandtl number, $\overline{c}_{p,o}\,\overline{\mu}_o/\overline{k}_o$
Q	mapping modulus
Re	Reynolds number, $Re = \overline{\rho}_o\,\overline{V}_o\,\overline{D}/\overline{\mu}_o$
u	velocity component in x direction normalized by $\overline{V}_o$
v	velocity component in y direction normalized by $\overline{V}_o$
$\overline{V}_o$	jet centerline velocity at exit plane
w	velocity component in z direction normalized by $\overline{V}_o$
W	width of solution domain normalized by $\overline{D}$
x	Cartesian coordinate normalized by $\overline{D}$
y	Cartesian coordinate normalized by $\overline{D}$
z	Cartesian coordinate normalized by $\overline{D}$

*This research was conducted under the Office of Naval Research Contract N00014-76-C-0494.

α	coefficient in general form of transport equation
β	coefficient in general form of transport equation
γ	coefficient in general form of transport equation; ratio of specific heats
η	mapping coordinate normalized by $\overline{D}$
θ	vector angle
δ	coefficient in general form of transport equation
μ	molecular viscosity normalized by $\bar{\rho}\ \overline{V}_0\ \overline{D}$ for incompressible flow and by $\bar{\mu}_0$ for compressible flow
μ_{eff}	effective viscosity normalized by $\bar{\rho}\ \overline{V}_0\ \overline{D}$
μ_{turb}	turbulent (eddy) viscosity normalized by $\bar{\rho}\ \overline{V}_0\ \overline{D}$
ξ	mapping coordinate normalized by $\overline{D}$
ρ	mass density normalized by $\bar{\rho}_0$
σ	source term in general form of transport equation
$\sigma_{k,turb}$	turbulent Prandtl number
ϕ	general flow variable; function in compressible flow equations
$\vec{\psi}$, ψ	stream function normalized by $\overline{V}_0\ \overline{D}$ for incompressible flow and by $\bar{\rho}_0\ \overline{V}_0\ \overline{D}$ for compressible flow
$\vec{\Omega}$, ω	vorticity normalized by $\overline{V}_0/\overline{D}$
$\rightarrow$	(arrow) vector quantity
$-$	(overbar) dimensional quantity
∞	ambient conditions

Introduction

With the growing interest in jet and fan-powered vertical-takeoff-and-landing (VTOL) military aircraft, there has been an increasing demand for improved performance-prediction methods. This demand is greatest for techniques to predict propulsion-induced aerodynamic effects in the hover mode of VTOL flight.

This task is a challenge to the computational aerodynamicist. As the schematic of Fig. 1 illustrates, the hover mode of a VTOL aircraft is characterized by complex flow phenomena. Ambient air is entrained into the lift jets and the wall jet, leading to an induced down-flow of air around the aircraft and a resulting suckdown force. In addition, the inward jet flows merge and create a stagnation region from which a hot-gas fountain emerges and impinges on the lower fuselage surface. The fountain is a source of positive induced forces which, to some extent, counteract the large suckdown forces near the ground. However, the fountain flow also heats the airframe surface and can result in the reingestion of hot gas into the inlet.

Clearly, the VTOL ground effect flow illustrated in Fig. 1 is characterized by three-dimensionality, high turbulence levels, compressibility, strong pressure gradients, and regions of stagnation-point and separated flow. These problem areas are critical in viscous flow simulations and cannot be adequately treated through inviscid-flow calculation techniques coupled with simple empirical or boundary-layer corrections. Rigorous treatment of this problem requires solution of the Navier-Stokes equations.

This paper discusses modeling the VTOL hover flowfield, concentrating mainly on the required computational algorithms. Treatment of the two-dimensional, incompressible ground effect problem is presented in detail, and extension of this method to compressible and three-dimensional flows is discussed. Although specific attention is given to VTOL aerodynamics, the conclusions related to the numerical algorithms apply to a variety of external and internal viscous flows of practical interest.

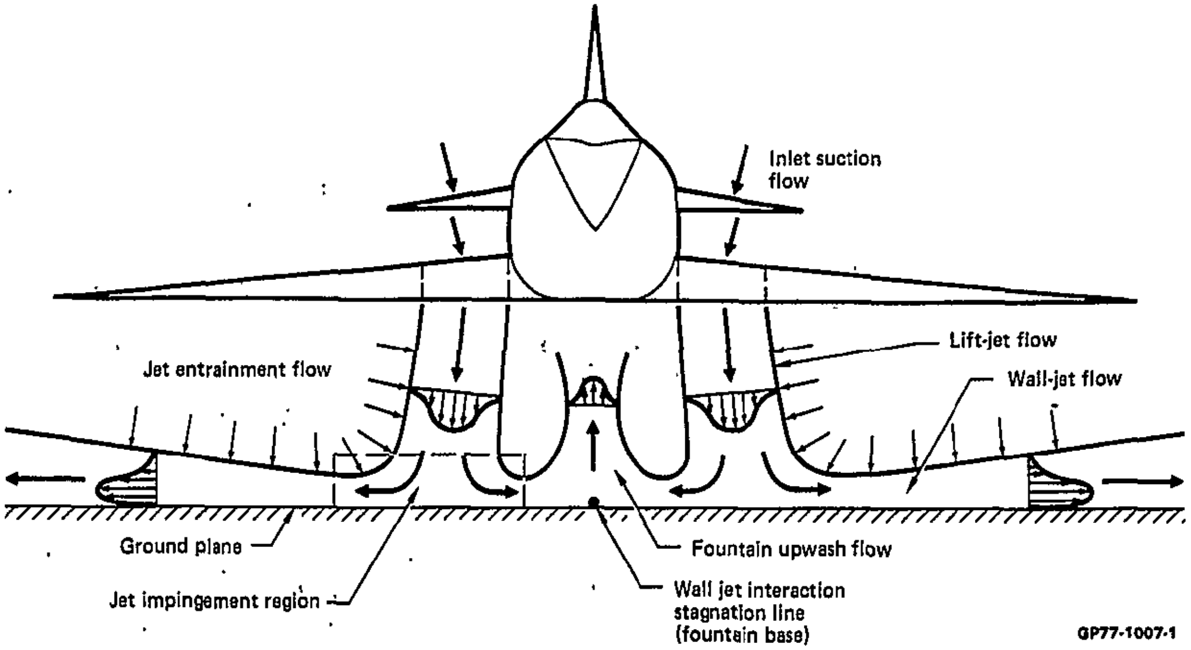


Fig. 1 Flowfield about a VTOL aircraft hovering in ground effect

Viscous Flow Simulations

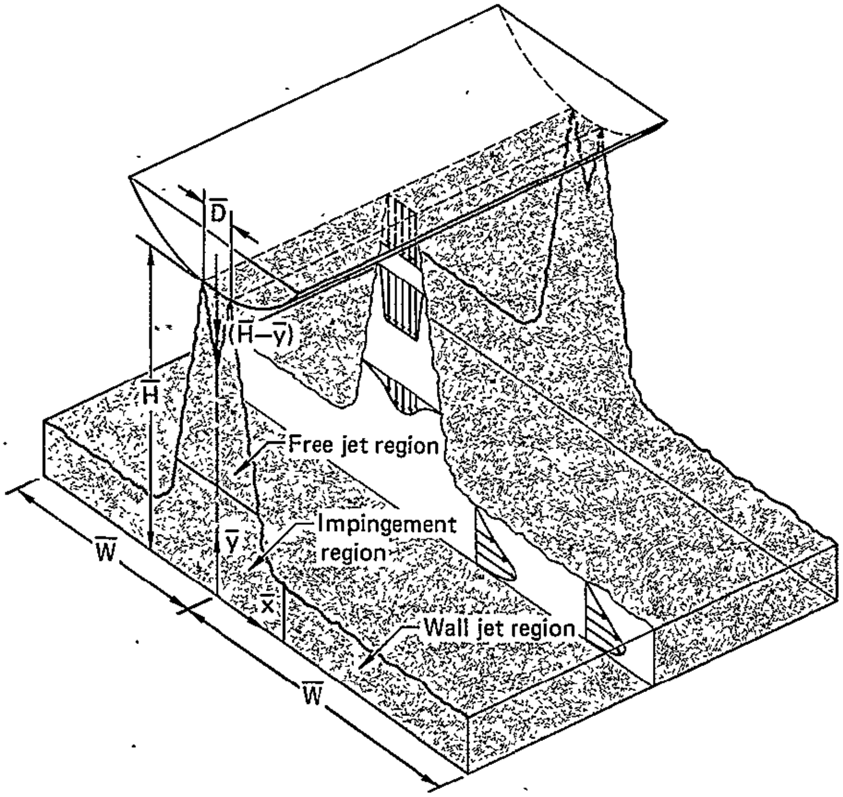
At the McDonnell Douglas Research Laboratories (MDRL), a flowfield model based on the Reynolds-averaged Navier-Stokes equations has been applied to the ground effect problem for steady, planar, incompressible, turbulent flow. In this section details of the model and solution algorithm for the governing equations are described, and the extension of this approach to two-dimensional compressible and three-dimensional incompressible flows is discussed.

Two-Dimensional Incompressible Flow

In order to gain a fundamental understanding of a lift-jet induced flow less complex than that shown in Fig. 1, MDRL has conducted both theoretical and experimental investigations of the flowfield created by a single planar lift jet in ground effect. The planar geometry was selected for the initial study instead of an axisymmetric geometry since the vectored planar jet flowfield can be computed with a two-dimensional analysis, while the vectored axisymmetric jet presents a fully three-dimensional problem.

The planar unvectored impinging jet flow is shown schematically in Fig. 2. The jet exits from a slot of width $\overline{D}$ in a contoured upper surface a distance $\overline{H}$ above the ground plane. The region of interest extends a distance $\overline{W}$ on each side of the jet centerline.

In the present approach, the time-averaged continuity and Navier-Stokes equations for steady, planar, incompressible flow are used to describe the mean motion of the fluid. Through the averaging procedure, unknown turbulent stress terms arise which are computed using a turbulent-kinetic-energy equation proposed by Wolfshtein[1] in combination with a phenomenological equation that relates the square root of the turbulent kinetic energy to turbulent viscosity.



GP77-1007-2

Fig. 2 The planar impinging jet

The governing equations are not written in primitive variable (velocity-pressure) form but rather in stream function-vorticity form to take advantage of the accurate and efficient numerical methods currently available to solve this system of equations. The stream function is defined by

$$\psi_y = u, \;\; \psi_x = -v, \tag{1}$$

and the vorticity is defined by

$$\omega = v_x - u_y. \tag{2}$$

Details of the derivation of the vorticity/stream-function form of the time-averaged conservation and turbulence model equations are presented in Ref. 2. The resulting equations are given below.

Poisson equation for stream function:

$$\psi_{xx} + \psi_{yy} = -\,\omega, \tag{3}$$

Vorticity transport equation:

$$\begin{aligned}(1 + \mathrm{Re}\,\mu_{turb})\,\omega_{xx} - \mathrm{Re}\,(\psi_y - 2\,\mu_{turb_x})\,\omega_x + (1 + \mathrm{Re}\,\mu_{turb})\,\omega_{yy}\\ + \mathrm{Re}\,(\psi_x + 2\,\mu_{turb_y})\,\omega_y = \mathrm{Re}\,(4\,\psi_{xy}\,\mu_{turb_{xy}} + \psi_{xx}\,\mu_{turb_{xx}}\\ + \psi_{yy}\,\mu_{turb_{yy}} - \psi_{yy}\,\mu_{turb_{xx}} - \psi_{xx}\,\mu_{turb_{yy}}),\end{aligned} \tag{4}$$

Turbulent kinetic energy equation:

$$(1/\sigma_k + \mathrm{Re}\,\mu_{turb}/\sigma_{k,turb})\,k_{xx} + \mathrm{Re}\,(\mu_{turb_x}/\sigma_{k,turb} - \psi_y)\,k_x$$
$$+ (1/\sigma_k + \mathrm{Re}\,\mu_{turb}/\sigma_{k,turb})\,k_{yy} + \mathrm{Re}\,(\mu_{turb_y}/\sigma_{k,turb} + \psi_x)\,k_y$$
$$= \mathrm{Re}\left\{C_D k^{3/2}/\ell_D - \mu_{turb}\left[4\,\psi_{xy}^{\;2} + (\psi_{yy} - \psi_{xx})^2\right]\right\}, \qquad (5)$$

Poisson equation for static pressure:

$$p_{xx} + p_{yy} = 4[\psi_{xx}\,\psi_{yy} - \psi_{xy}^{\;2} - \mu_{turb_x}\,\omega_y + \mu_{turb_y}\,\omega_x$$
$$+ \psi_{xy}\,(\mu_{turb_{xx}} - \mu_{turb_{yy}}) - \mu_{turb_{xy}}\,(\psi_{xx} - \psi_{yy})], \qquad (6)$$

where

$$\mu_{turb} = c_\mu\,k^{1/2}\,\ell_\mu \qquad (7)$$

and

$$\mu_{eff} = 1/\mathrm{Re} + \mu_{turb}. \qquad (8)$$

The turbulence modeling constants, C_D and c_μ, and the length scales, ℓ_D and ℓ_μ, are specified in Ref. 2. The length scales are an important element of the one-equation turbulence model in that they significantly influence the level of the turbulent viscosity throughout the field.

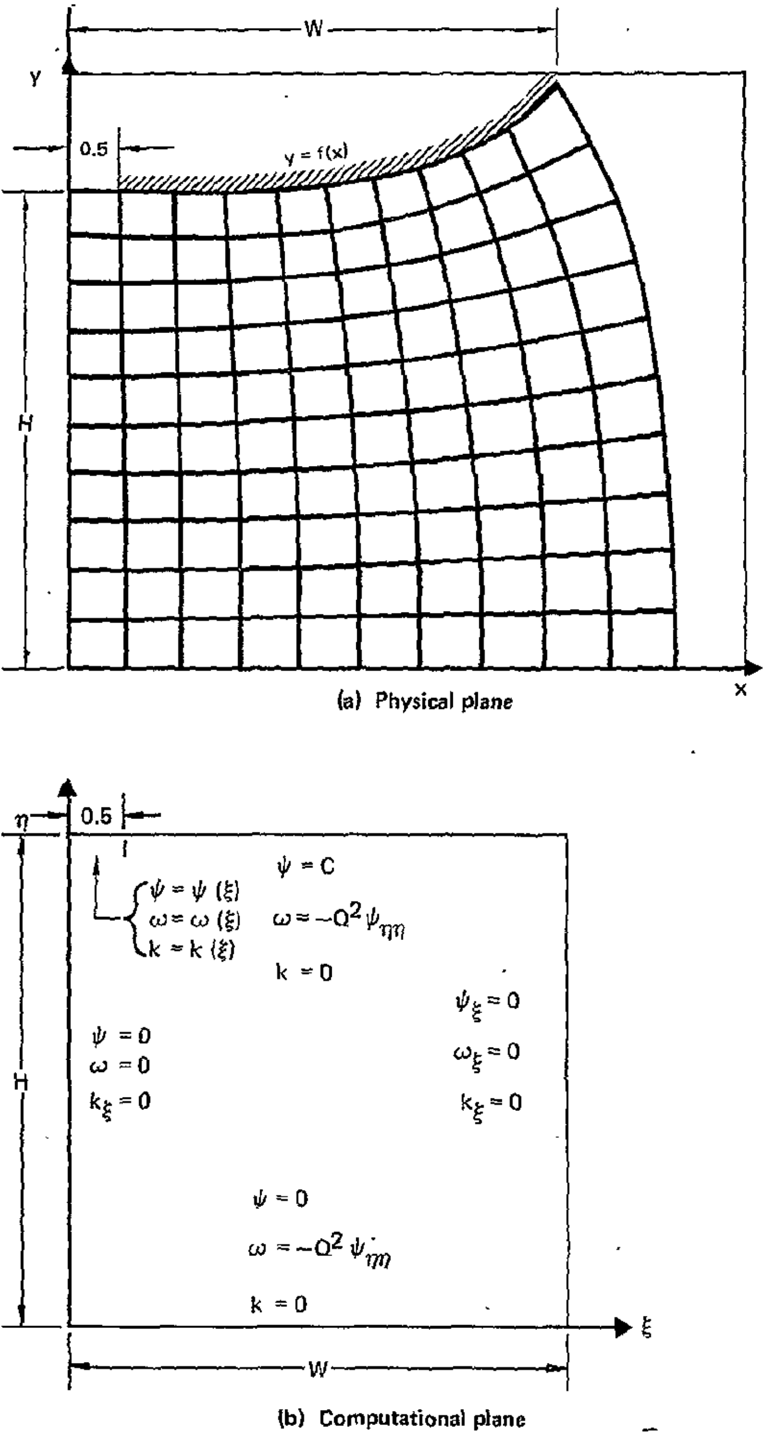
Equations (1) through (8) have been written in dimensionless form by using the normalizing parameters $\overline{D}$ (the jet width at the exit plane), $\overline{V}_o$ (the jet centerline velocity at the same station), and $\bar{\rho}$ (the constant fluid density). This normalization introduces the Reynolds number based on properties at the jet exit plane, $\mathrm{Re} = \bar{\rho}_o\,\overline{V}_o\,D/\bar{\mu}_o$.

To solve the governing equations for a flow with the contoured upper boundary used to simulate the lower surface of a fuselage (Fig. 2), a conformal mapping procedure is introduced. In this technique, which was originally devised at MDRL by G. H. Hoffman, a finite-difference computational plane with coordinates (ξ,η) is specified. The distance between nodes in the ξ direction is a, and the distance in the η direction is b, where a and b are not necessarily equal. A conformal mapping given by

$$\xi + i\eta = F\,(x + iy) \qquad (9)$$

is introduced which determines the physical plane (x,y). Laplace's equation is satisfied by both x and y and is solved for each variable subject to the required boundary conditions. The latter follow from physical constraints when they are known at the boundaries and from integration of the Cauchy-Riemann relations for x and y when the boundary distributions are not known. The derivatives in these equations are rewritten in terms of the computational plane coordinates and a mapping modulus Q.

Figure 3 illustrates the physical and computational planes used in the calculation of the two-dimensional ground effect flowfields, along with the boundary conditions imposed on the primary flow variables (stream function, vorticity, and turbulent kinetic energy). Since only normal impingement is considered, geometric symmetry about the jet centerline exists so that only half the flowfield need be solved. The stream function and vorticity are asymmetric about the centerline, and the turbulent kinetic energy is symmetric. Boundary conditions imposed on ψ, ω, and k follow



(a) Physical plane

(b) Computational plane

Fig. 3 Specification of the boundary conditions for the primary flow variables

GP77-1007-3

from the no-slip, impermeable wall constraint at the solid surfaces, from symmetry at the jet centerline, and from the assumption of no gradients in the ξ -direction at the right boundary. The last boundary condition is not accurate for relatively small values of W; in these cases experimental data should be used to better define the flow properties.

With conformal mapping, the elliptic partial-differential equations that describe the flow can be written in the form

$$\alpha\phi_{\xi\xi} + \gamma\phi_{\xi} + \beta\phi_{\eta\eta} + \delta\phi_{\eta} = \sigma, \tag{10}$$

where α, γ, β, and δ denote the nonlinear coefficients, and σ denotes the source term. For the two Poisson equations, $\phi = \psi$ or $\phi = p$, Eq. (10) can be solved numerically without difficulty using the conventional central-difference (CD) finite-difference algorithm, which is accurate to second order. For the vorticity transport equation, $\phi = \omega$, and for the turbulence model equation, $\phi = k$, the CD algorithm presents problems. The coefficients for these equations contain the Reynolds number as a multiplicative factor, and, as a result, with the standard CD algorithm, the discretized system is diagonally dominant for only a limited range in the coefficients γ and δ. Diagonal dominance is necessary to obtain convergence in the iterative solutions of the discretized system of equations.

One approach for obtaining convergent solutions at high Reynolds numbers uses a one-sided finite-difference scheme to represent the convection terms appearing in Eq. (10). However, this technique is only first-order accurate as opposed to the second-order accuracy for central differencing. Consequently, in the present work the vorticity transport equation and the turbulent-kinetic-energy equation are solved using the augmented-central-difference (ACD) algorithm developed by G. H. Hoffman at MDRL[3]. The essence of this method can be illustrated by considering the derivative ϕ_{ξ} of Eq. (10). Using the five-point finite-difference stencil shown in Fig. 4 and point-of-the-compass notation, this derivative can be evaluated at point P using the following Taylor-series representation and standard CD approximation to the first derivative:

$$\phi_{\xi}|_{P} = (\phi_{E} - \phi_{W})/2a - (a^2/6)\,\phi_{\xi\xi\xi}|_{P} - (a^4/5!)\,\phi_{\xi\xi\xi\xi\xi}|_{P}. \tag{11}$$

In the ACD scheme, the derivative $\phi_{\xi\xi\xi}$ is retained and is expressed in terms of lower-order derivatives by differentiating Eq. (10) with respect to ξ. The derivative ϕ_{η} in Eq. (10) is represented in an analogous fashion with the ACD algorithm.

The finite-difference forms of the flow equations are solved iteratively using point relaxation. First, a convergent solution of the Poisson equation for stream function, the vorticity transport equation, and the turbulent-kinetic-energy equation is obtained. Then the primitive flow variables (static pressure and the velocity components) are calculated. The Poisson equation for static pressure is solved subject to the boundary conditions on the normal pressure gradients imposed by the time-averaged momentum equations, and the velocity components are computed from the defining equations for the stream function. For the case of incompressible flow, calculation of the pressure field can be deferred until after stream function, vorticity, and turbulent-kinetic-energy distributions have been evaluated.

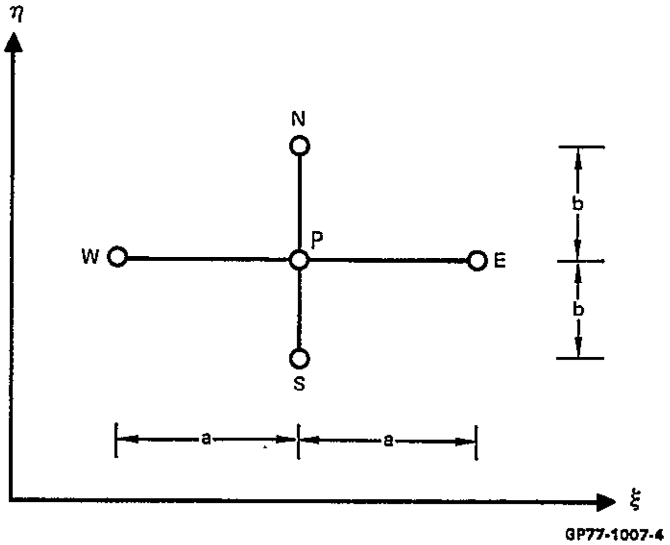


Fig. 4 Five-point finite-difference stencil

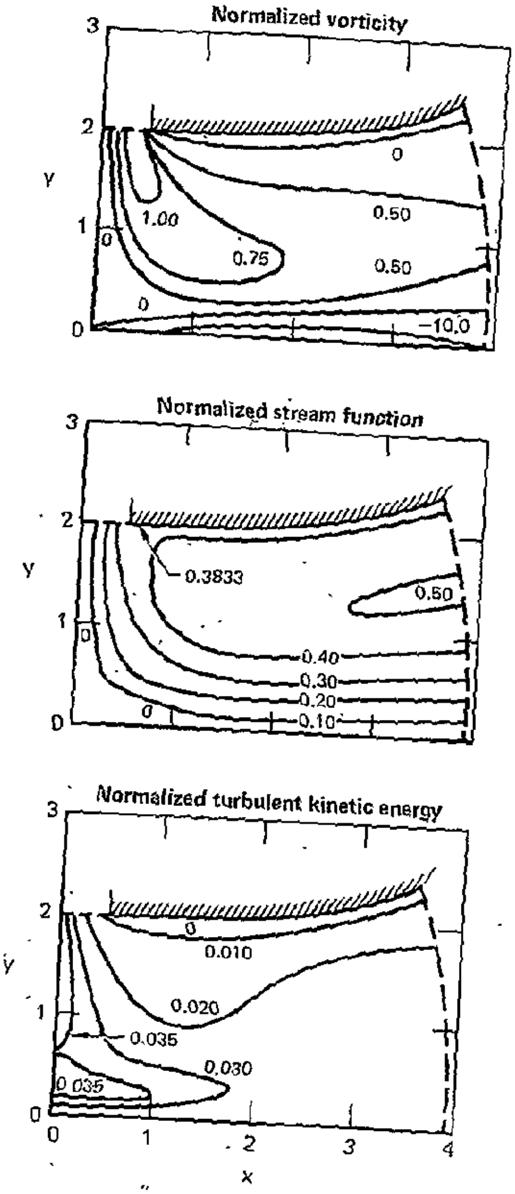
Flowfields have been computed for the planar impinging jet illustrated in Fig. 2 with various values of H and Re using the CYBER 173 system of the McDonnell Douglas Automation Company. Figures 5 and 6 contain the contour plots of the primary and primitive flow variables for the geometry of Fig. 3 with H = 2, W = 3.68, and Re = 100 000. The following basic flow characteristics can be observed in the solutions: a strong convection of vorticity toward the right boundary with separation near the slot edge, a region of recirculating flow with fluid entrainment into the free jet, and strong pressure gradients in going toward stagnation point along the jet centerline and the lower wall.

Specific comparisons between measured and computed data for this geometry are shown in Fig. 7. In the theoretical pressure distribution, Fig. 7(a), the pressure values at the end points of the right boundary have been used for p_∞ at each surface. The computed normalized profiles of $p - p_\infty$ reproduce the lower-wall pressure drop in the impingement region and the relatively constant, low-pressure level along the upper surface. Good agreement between the measured and computed centerline velocity variations is also obtained, Fig. 7(b).

Two-Dimensional Compressible Flow

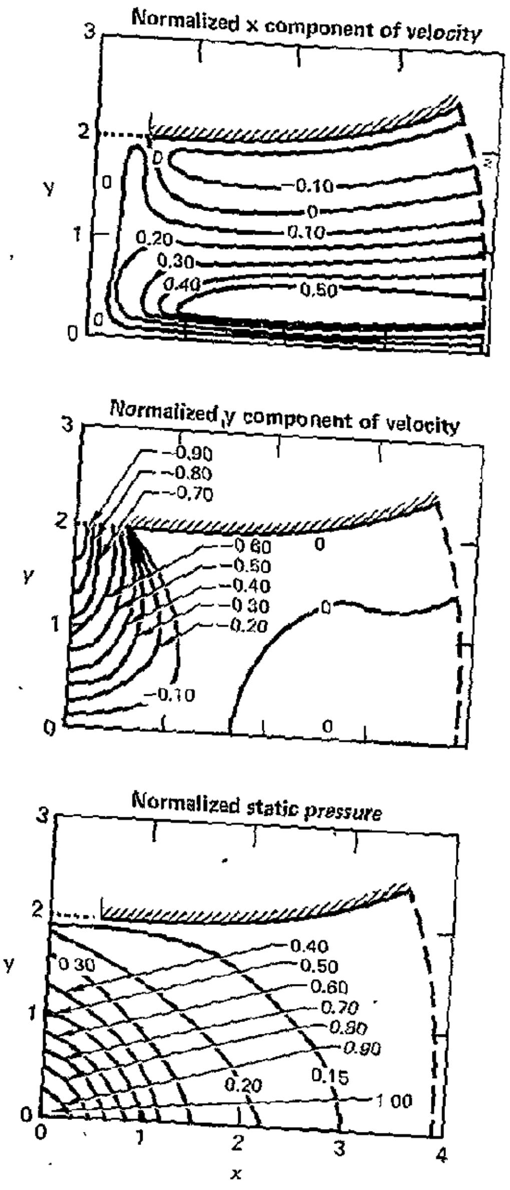
Currently work is in progress at MDRL to solve the compressible flowfield associated with a two-dimensional lift jet which is at a temperature much higher than that of the surrounding air. Density variations between the ambient air and the less dense lift jet have an influence on the entrainment of air at the free boundaries of the free jet and the ground wall jets. In addition, mixing of the ambient entrained air with the hot lift jet fluid thickens the free jet and the wall jets. The latter will eventually separate from the ground because of buoyant forces.

The geometry of interest remains that shown in Fig. 2. The governing equations are the time-averaged continuity and Navier-Stokes equations for steady, planar, compressible flow in conjunction with an extension of Wolfshtein's turbulence model[1] to account for compressibility. The equations are again solved in stream function-vorticity form to use the numerical algorithm developed for the transport-type equations. For simplicity in explaining the numerical procedure, the case of the laminar impinging jet is considered here.



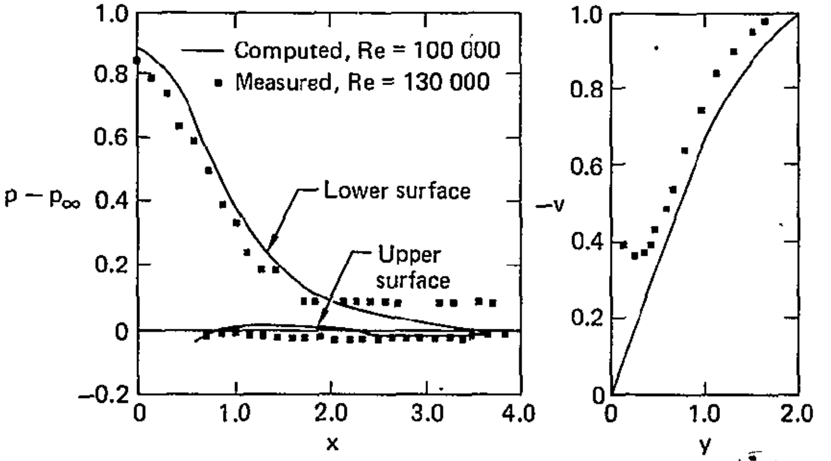
GP77-1007-5

Fig. 5 Primary flow variables for the curved-plate geometry (H = 2, W = 3.68, Re = 100 000)



GP77-1007-6

Fig. 6 Primitive flow variables for the curved plate geometry (H = 2, W = 3.68, Re = 100 000)



(a) Surface static pressure variations

(b) Centerline velocity profile

GP77-1007-7

Fig. 7 Comparisons of computed and measured flow properties for the planar impinging jet, H = 2

A compressible stream function is introduced,

$$\psi_y = \rho u, \quad \psi_x = -\rho v, \tag{12}$$

and the defining relation for the vorticity, Eq. (2), remains the same. The governing equations are given below.

Poisson equation for stream function:

$$\psi_{xx} - \rho_x \psi_x/\rho + \psi_{yy} - \rho_y \psi_y/\rho = \rho\omega \tag{13}$$

Vorticity transport equation:

$$\mu(\omega_{xx} + \omega_{yy}) + (2\,\mu_x - \mathrm{Re}\,\psi_y)\omega_x + (2\mu_y + \mathrm{Re}\,\psi_x)\,\omega_y$$

$$= \mathrm{Re}\,\phi_1 - \phi_2 - (\mathrm{Re}/\mathrm{Fr}^2)\rho_x \tag{14}$$

Poisson equation for static pressure:

$$p_{xx} + p_{yy} = (2/\mathrm{Re})\,(\mu_{xx}\,\phi_3 + \mu_x\,\phi_4 + 2\,\mu_{xy}\,\phi_5 + \mu_y\,\phi_6$$

$$+ \mu_{yy}\,\phi_7 + 4\,\mu\,\phi_8/3) - 2\phi_9 + 2\rho_y/\mathrm{Fr}^2 \tag{15}$$

Thermal energy equation:

$$(1/\mathrm{Pr})\,(k/c_p)\,(h_{xx} + h_{yy}) + [(1/\mathrm{Pr})\,(k/c_p)_x - \mathrm{Re}\,\psi_y]\,h_x$$

$$+ [(1/\mathrm{Pr})\,(k/c_p)_y + \mathrm{Re}\,\psi_x]\,h_y = (\mathrm{Re}/2\rho)\,(\psi_x\,p_y - \psi_y\,p_x)$$

$$- \phi_{10} - (\mathrm{Re}/\mathrm{Fr}^2)\psi_x \tag{16}$$

Equation of state:

$$p = 2\rho h\,(\gamma - 1)/\gamma \tag{17}$$

Transport properties:

$$\mu = \mu_1 (h) \qquad (18)$$

$$k = k_1 (h). \qquad (19)$$

Equations (12) through (19) are written in dimensionless form. Two additional parameters enter the problem for the case of compressible flow. These are the Froude number, $Fr = \bar{V}_o / \sqrt{\bar{g}_o \bar{D}}$, and the Prandtl number,

$$Pr = \bar{c}_{p,o} \bar{\mu}_o / \bar{k}_o.$$

The terms ϕ_1 through ϕ_{10} appearing in the equations involve derivatives of the stream function, vorticity, and density and are omitted here for brevity.

The conformal mapping and finite-difference procedures described previously can be directly applied for solution of the governing equations subject to the required boundary conditions. Since these terms are rather lengthy, calculation of the source terms in the governing equations requires more machine computation time for the case of compressible flow than for the case of incompressible flow. In addition, the Poisson equation for static pressure must be solved in combination with the remaining equations since the density depends on the static pressure. Calculations of the latter cannot be deferred until the end of the computations as is the case for incompressible flow.

Three-Dimensional Incompressible Flow

Work is also in progress at MDRL to solve the flowfield associated with a three-dimensional impinging jet in ground effect. This configuration is of practical significance since it is representative of the actual lift jets in VTOL aircraft.

To generate this geometry, an axisymmetric jet which impinges normal to the ground, Fig. 8, is rotated through some angle θ with regard to the normal. The governing equations are the time-averaged continuity and Navier-Stokes equations for steady, planar, incompressible flow in combination with an appropriate turbulence model. An extension of the stream function-vorticity concept to three dimensions is introduced to take advantage of the numerical algorithm described previously for transport-type equations. As before, the laminar impinging jet is considered here to simplify the numerical procedure.

Following Aziz and Hellums[4], for a three-dimensional velocity field

$$\vec{V} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \qquad (20)$$

a vorticity vector $\vec{\Omega}$ is defined by

$$\vec{\Omega} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \nabla \times \vec{V}, \qquad (21)$$

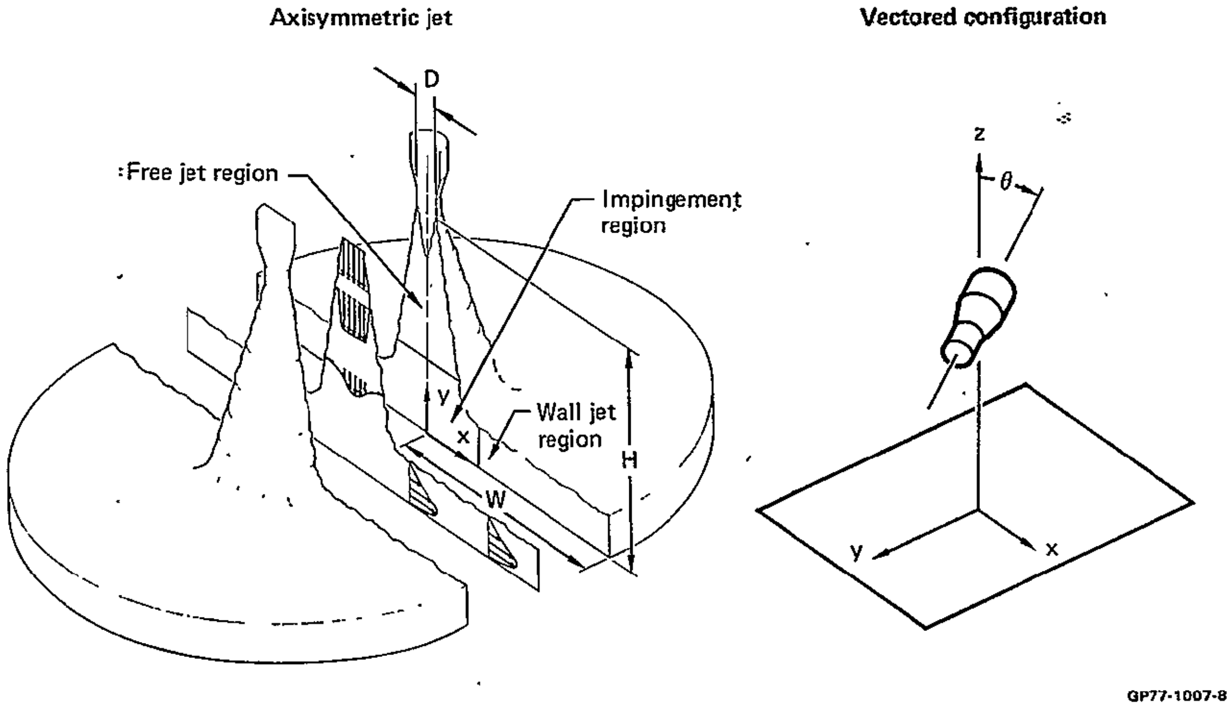


Fig. 8 Three-dimensional impinging jet geometry

and a three-dimensional counterpart $\vec{\psi}$ of the two-dimensional stream function is defined by

$$\vec{V} = \nabla \times \vec{\psi} \qquad (22)$$

with

$$\vec{\psi} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix}. \qquad (23)$$

With the constraint $\nabla \cdot \vec{\psi} = 0$, the following governing equations describe the three-dimensional incompressible flow:

Poisson equations for the stream functions:

$$\psi_{1_{xx}} + \psi_{1_{yy}} + \psi_{1_{zz}} = -\omega_1 \qquad (24)$$

$$\psi_{2_{xx}} + \psi_{2_{yy}} + \psi_{2_{zz}} = -\omega_2 \qquad (25)$$

$$\psi_{3_{xx}} + \psi_{3_{yy}} + \psi_{3_{zz}} = -\omega_3 \qquad (26)$$

Vorticity transport equations:

$$\begin{pmatrix} \psi_{3_y} - \psi_{2_z} \\ \psi_{1_z} - \psi_{3_x} \\ \psi_{2_x} - \psi_{1_y} \end{pmatrix} \cdot \nabla \omega_1 - \nabla(\psi_{3_y} - \psi_{2_z}) \cdot \vec{\Omega} - \nabla^2 \omega_1/\mathrm{Re} = 0 \tag{27}$$

$$\begin{pmatrix} \psi_{3_y} - \psi_{2_z} \\ \psi_{1_z} - \psi_{3_x} \\ \psi_{2_x} - \psi_{1_y} \end{pmatrix} \cdot \nabla \omega_2 - \nabla(\psi_{1_z} - \psi_{3_x}) \cdot \vec{\Omega} - \nabla^2 \omega_2/\mathrm{Re} = 0 \tag{28}$$

$$\begin{pmatrix} \psi_{3_y} - \psi_{2_z} \\ \psi_{1_z} - \psi_{3_x} \\ \psi_{2_x} - \psi_{1_y} \end{pmatrix} \cdot \nabla \omega_3 - \nabla(\psi_{2_x} - \psi_{1_y}) \cdot \vec{\Omega} - \nabla^2 \omega_3/\mathrm{Re} = 0 \tag{29}$$

Poisson equation for static pressure:

$$\begin{aligned} p_{xx} + p_{yy} = -2 \Big[& (\psi_{3_{yx}} - \psi_{2_{zx}})^2 + (\psi_{1_{zy}} - \psi_{3_{xy}})^2 \\ & + (\psi_{2_{xz}} - \psi_{1_{yz}})^2 + 2(\psi_{1_{zx}} - \psi_{3_{xx}})(\psi_{3_{yy}} - \psi_{2_{zy}}) \\ & + 2(\psi_{2_{xx}} - \psi_{1_{yx}})(\psi_{3_{yz}} - \psi_{2_{zz}}) + 2(\psi_{2_{xy}} - \psi_{1_{yy}})(\psi_{1_{zz}} - \psi_{3_{xz}}) \Big] . \end{aligned} \tag{30}$$

Equations (20) through (29) have been written in dimensionless form, introducing the Reynolds number into the problem. The ACD finite-difference algorithm can be extended to the three-dimensional case for solution of Eqs. (27) through (29) with specification of the appropriate boundary conditions. However, the terms which appear in the discretized forms of these equations are rather lengthy.

Summary

A finite-difference technique has been developed for solving the stream function-vorticity form of the governing equations describing a VTOL aircraft ground-effect flowfield. For the case of two-dimensional incompressible flow, the method provides an accurate and efficient means of solution. But as the stream function-vorticity formulation is extended to two-dimensional compresssible and three-dimensional incompressible flows, the algorithm becomes less efficient.

Numerical algorithms are required which are based on solution of the governing equations in primitive-variable form. For example, an investigation should be made of the feasibility of extending the box method of Keller[5] to the elliptic case. This scheme applied to parabolic equations has been used successfully by Cebeci and Smith[6] for calculation of the boundary-layer equations.

Acknowledgment

The author acknowledges Dr. G. H. Hoffman who originated the conformal mapping and finite-difference procedures used in the analysis, Dr. Galen R. Peters who formulated the three-dimensional flow equations, and Dr. D. R. Kotansky who acquired the experimental data shown in Fig. 7.

References

1. M. Wolfshtein, Convection Processes in Turbulent Impinging Jets, Report SF/R/2, Department of Mechanical Engineering, Imperial College of Science and Technology, November 1967.

2. W. W. Bower and D. R. Kotansky, A Navier-Stokes Analysis of the Two-Dimensional Ground Effects Problem, AIAA Paper No. 76-621, 1976.

3 G. H. Hoffman, Calculation of Separated Flows in Internal Passages, Proceedings of a Workshop on Prediction Methods for Jet V/STOL Propulsion Aerodynamics (1975), Vol. 1, pp. 114-124.

4. K. Aziz and J. D. Hellums, Numerical Solution of the Three-Dimensional Equations of Motion for Laminar Natural Convection, Phys. Fluids 10, 314 (1967).

5. H. B. Keller, in Numerical Solution of Partial Differential Equations, ed. B. Hubbard (Academic Press, New York, 1970) Vol. II.

6. T. Cebeci, and A. M. O. Smith, Analysis of Turbulent Boundary Layers (Academic Press, New York, 1974).

CRITICAL ISSUES IN VISCOUS FLOW COMPUTATIONS

W. L. HANKEY
Air Force Flight Dynamics Laboratory
Wright-Patterson AFB, Ohio

In developing computer programs to numerically solve the Navier-Stokes equations, the purpose of the computation must be clearly kept in mind. In the Air Force, our purpose is to provide design information on "non-linear" aerodynamic phenomenon for aircraft that perform throughout the flight corridor. This translates into the requirement for a computer program which can solve the time averaged compressible Navier-Stokes equations (with a turbulence model) in three dimensions for generalized geometries. The intended application of the results then controls the priorities in addressing critical issues.

In our investigations of viscous flows, several problem areas keep recurring. (Most of these are topics for subsequent discussions.) They are as follows:

1. Grid generation for arbitrary geometry
2. Numerical difficulties
3. Turbulence models
4. Accuracy and efficiency
5. Smearing of discontinuities

GRID GENERATION FOR ARBITRARY GEOMETRY

It is generally accepted that viscous flow problems require a surface-oriented coordinate system. Also for arbitrary geometries, automation of a numerical transformation (as opposed to an analytic transformation) is necessary. In addition, some optimization of the distribution of grid points throughout the flow field is necessary to economically solve practical problems. Conceptually, this implies that higher order derivatives (in the transformed plane) of the primary dependent variable be minimized.[1] The distribution of the grid points greatly influences the requirement of the number of field points necessary to achieve a desired accuracy. Considerably more attention is needed in this area to improve the economics of the viscous flow computations.

NUMERICAL DIFFICULTIES

This is a "catch all" term to cover the reasons a program "bombs out". Given a proven algorithm and an experienced user with a properly formulated problem, program failures are still common during the initial phase of the investigation. The problems are most frequently due to large truncation errors which eventually swamp the true solution. The cause of the problem is that the grid cannot truly be established until the flow-field is determined. A redistribution or increase in the number of grid points often permits success. Artfully changing the damping coefficients in the region of discontinuities has also been successful. In addition, alternate approaches for expressing the boundary conditions can have a dramatic effect on the success or failure of a problem. A requirement exists for a method in which the flowfield modifies its own numerical grid where needed. Also, additional program guidelines are needed to ensure a more robust code.

TURBULENCE MODELS

In time-averaging the Navier-Stokes equations information is lost. Information must be re-inserted into the governing equations by resorting to experimental observation. The engineer needs empirically determined transport properties to proceed with the numerical computation. A large body of data exists for flat plate boundary layers and good correlations have evolved which generally permit calculations to be performed that fit the data to within ±10% for skin friction and boundary layer thickness[2] (see Fig. 1). Unfortunately, the agreement for the pressure gradient case is not nearly as good. Higher order closure schemes have not greatly improved the prediction capability. There is a need for the measurement of turbulent Reynolds stresses under pressure gradient for a wide range of flow conditions to permit correlations comparable to the flat plate case. Without this data, progress in the field will be limited.

Many skeptics are pessimistic of our ability to compute turbulent flows in the near future. Turbulence is felt to be too complex and the progress has been slow in developing a thorough understanding. To counter these skeptics, an encouraging viewpoint is offered. First, the good design predictions of flat plate properties are possible without fully understanding the true mechanism of turbulence. Secondly, in some cases it may be possible to bracket the extremes of flows with pressure gradient by computing the frozen and equilibrium states[3], thereby, providing useable design information (Fig. 2). Thirdly, remarkable results are possible[4]in the prediction of gross turbulent properties by simply treating the eddy viscosity as a constant ($\frac{\rho U x}{\varepsilon} = Re_t = \text{const.}$) Turbulence is limited and confined, and these approximate results are easy to compute; the difficulty is in reducing the error bounds to satisfy the scientist. Fourthly, in most applications, only displacement effects which influence the pressure distribution (separation point location) are significant. Skin friction and heat transfer, which require greater numerical resolution, are often of secondary importance.

One last point concerning the future development of turbulence models the models to date have been analytical in nature. New models have an additional requirement to be compatible with numerical computation. We need something like "digital turbulence".

ACCURACY AND EFFICIENCY

Accuracy and efficiency should be addressed concurrently because of their interrelationship. Given a stable algorithm, the greatest control on spatial accuracy is the number and distribution of grid points. Figure 3[10] shows the error in drag coefficient vs number of points in one coordinate direction in an airfoil flowfield. The computational time increases with N^2 (for a two dimensional problem) and hence it is very expensive to obtain the last few percent accuracy. The accuracy requirements of any design problem must be very carefully defined in order to avoid excessive computer cost.

Once satisfactory spatial accuracy is achieved, a convergence criterion must be selected which produces comparable accuracy. A time dependent approach is generally used to solve the Navier-Stokes equations in which the computation proceeds from an arbitrary initial condition until a steady state solution is achieved. In the past, several (maybe 5) characteristic times $(\frac{L}{U_\infty})$ have been sufficient for the initial transient to decay. However, based upon the analytical solution of an impulsively started flat plate, the error between the transient value and steady state decays as $t^{-½}$. This slow convergence rate implies that to cut the error in half, the computer time must be increased by a factor of four[5] (for the same Δt). (See Fig. 4) Another discouraging aspect is that for some flows, periodic values are legitimate steady state solutions. For example, subsonic airfoils near stall shed vortices in a regular manner[6] (Fig. 5 and movie). Computations must be accomplished for many characteristic times to achieve mean and rms values for design application. Slow convergence could well be our most critical problem in our goal to economically produce aerodynamic design data.

Paramount to all of these issues is the fact that a good finite difference algorithm is used to solve the governing equations. Considerable success has been achieved with MacCormack's method[7] to solve supersonic viscous flows. MacCormack's explicit method possesses many desirable features with the exception of efficiency. The CFL stability limit requires small time steps where small spatial steps are required to resolve viscous regions. To relieve this restriction, implicit methods have been developed which are conceptually unconditionally stable. However, our experience shows a gain in efficiency only in the viscous region. Accuracy (not stability) requirements in the inviscid region can be achieved only for the CFL time step. Hence, the hybrid method[5,8] (explicit in the inviscid and implicit in the viscous region) is at present probably the most efficient method available.

SMEARING OF DISCONTINUITIES

In examining viscous flow problems, two scale lengths appear. One is the mean free path, $\lambda \sim \frac{\nu}{V}$; the other, which is introduced through the boundary conditions, is a characteristic geometric length, L. One can also derive another scale length, $\delta \sim \sqrt{L\lambda}$, which is a combination of the previous two lengths. In numerically solving any viscous flow problem, the grid size, Δy, should be sufficiently small to accurately resolve these three scale lengths (L, δ , λ). This, of course, is impossible to achieve in nearly any practical problem today. Slip lines, shock waves and leading edges are examples where the characteristic lengths are too small to be honored. As a consequence, these discontinuities are incorrectly computed. Large errors exist in the immediate vicinity of these regions and numerical smearing results. Based on both wind tunnel and computational experience, it is believed that these local errors near singularities do not totally invalidate the global results. Figure 6 shows a Navier-Stokes computation[9] of a high speed inlet flow indicating good agreement with experiment with the exception of the shock jump and the entropy layer generated by the cowl lip leading edge. More effort is required to minimize the smearing of these discontinuities.

CONCLUSION

Although additional research is required, we believe all the necessary components for the numerical wind tunnel exist. The main requirement is the need for a computer larger than presently available. It appears doubtful that the computer centers of most organizations can completely service the needs of all their users. Therefore, national facilities will be necessary to solve the few large problems each organization requires. Collectively, these users can justify the need for a huge computer. Computational fluid dynamics, weather modeling, aero-elastic-structural analysis and physical chemistry are fields that, to advance, require computers larger than currently exist. By joining forces we could share the cost and satisfy all of our needs.

REFERENCES

1. Ghia, U., Hodge, J.K. and Hankey, W.L., "Numerical Generation of Surface-Oriented Coordinates for Arbitrary Geometries-An Optimization Study" to be published as AFFDL T.R.

2. Shang, J.S., Hankey, W.L. and Dwoyer, D.L., "Numerical Analysis of Eddy Viscosity Models in Supersonic Boundary Layers", AIAA Journal, Vol. 11, Dec. 1973, pp 1677-1683.

3. Shang, J.S. and Hankey, W.L., "Numerical Solutions for Supersonic Turbulent Flow Over a Compression Ramp", AIAA Journal, Vol. 13, No. 10, pp 1368-1374.

4. Birch, S.F., "A Critical Reynolds Number Hypothesis and its Relation to Phenomenological Turbulence Models", Proceedings of the 1976 Heat Transfer and Fluid Mechanics Institute, Stanford University Press, 1976, pp 152-164.

5. Shang, J.S., "An Implicit-Explicit Method for Solving the Navier-Stokes Equations", submitted to AIAA.

6. Hodge, J.K. and Stone, A.L., "A Numerical Solution of the Navier-Stokes Equations in Body-Fitted Curvilinear Coordinates", submitted to AIAA for presentation.

7. MacCormack, R.W., "Numerical Solutions of the Interaction of a Shock Wave with a Laminar Boundary Layer", Lecture Notes in Physics, Vol. 8, Springer-Verlag, New York, 1971, pp 151-163.

8. MacCormack, R.W., "An Efficient Numerical Method for Solving the Time-Dependent Compressible Navier-Stokes Equations at High Reynolds Number", NASA TMX-73,129, July 1976.

9. Knight, D.D, "Numerical Simulation of Realistic High-Speed Inlets Using the Navier-Stokes Equations", to be published in AIAA Journal.

10. Thompson, J.F., and Thames, F.C., "Numerical Solution of Potential Flow about Arbitrary Two-Dimensional Multiple Bodies", to be published in NASA CTR.

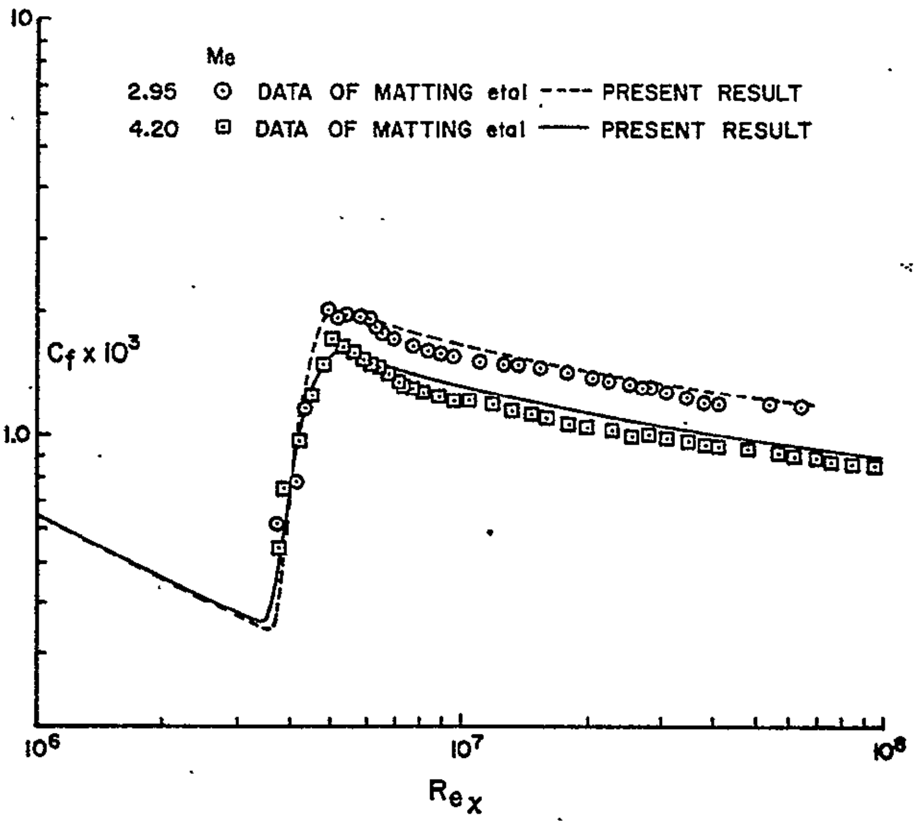


Fig 1. Comparison of Skin Friction Values for a Flat Plate Boundary Layer

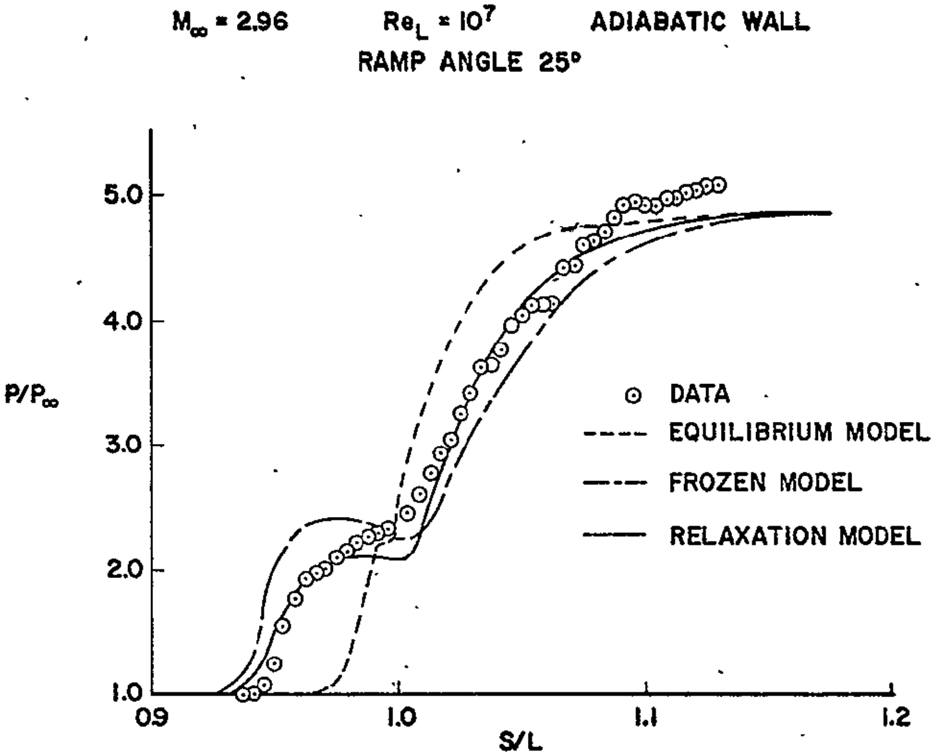


Fig 2. Comparison of Frozen and Equilibrium Turbulence Models for Compression Ramp

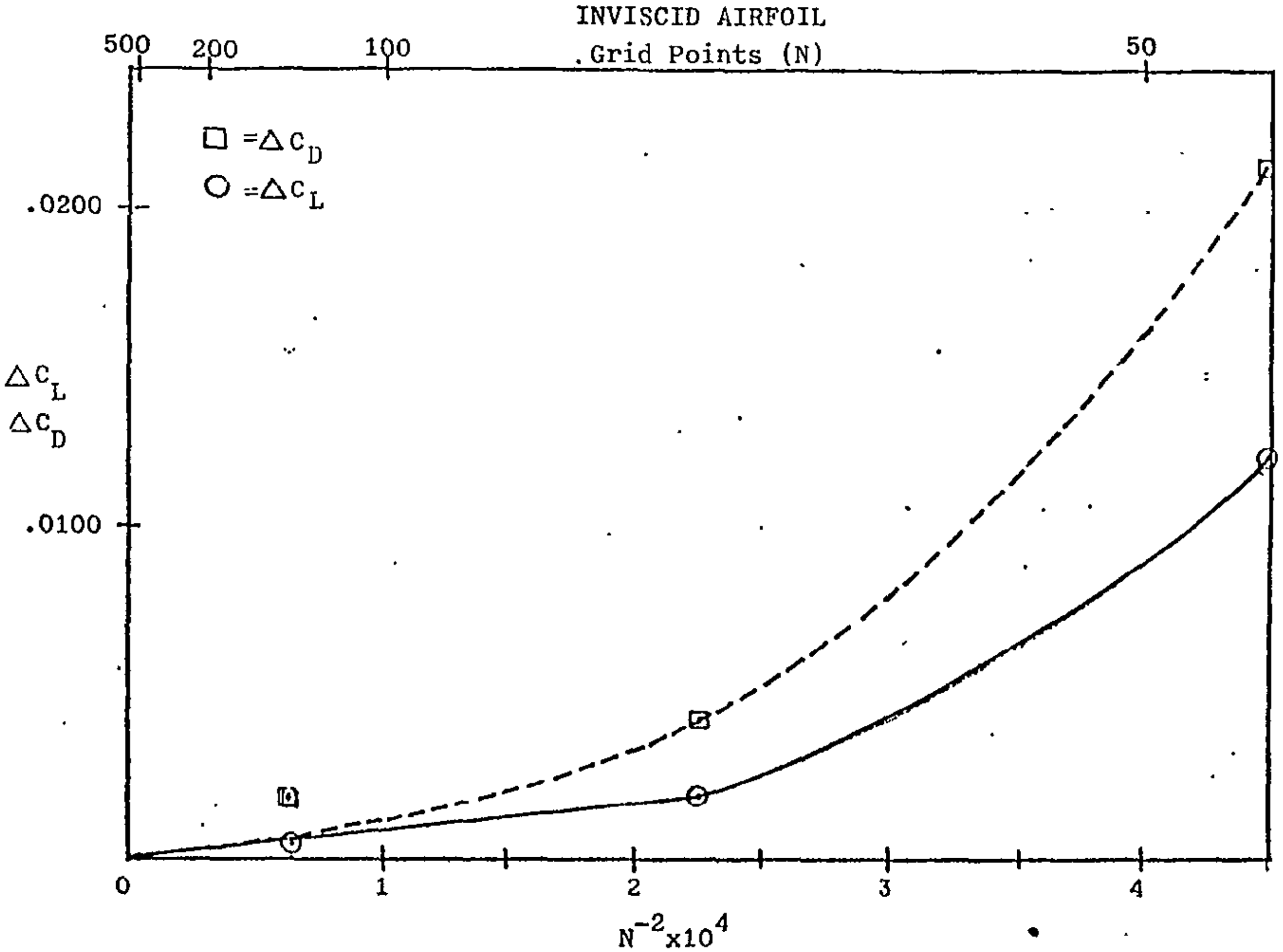


Fig 3. Error in Force Coefficients vs Number of Grid Points for Inviscid Flow Over an Airfoil[10]

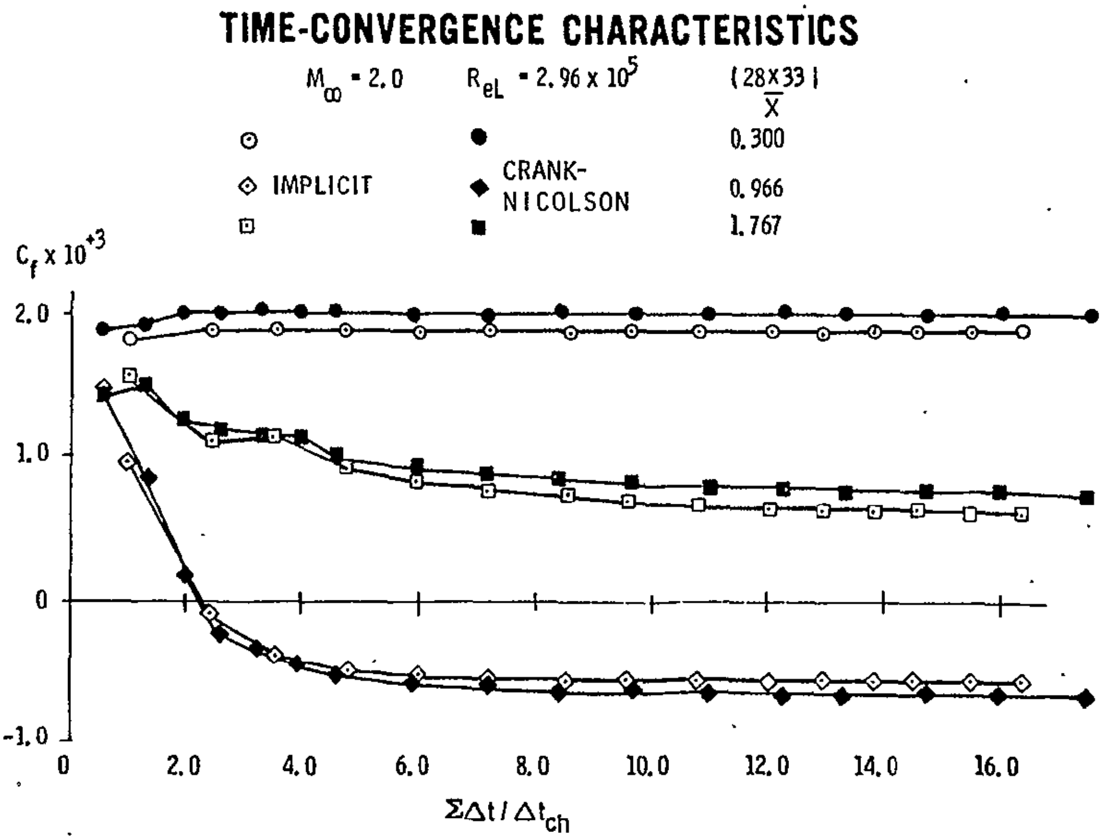


Fig 4. Convergence Characteristics for Shock Wave Impingement

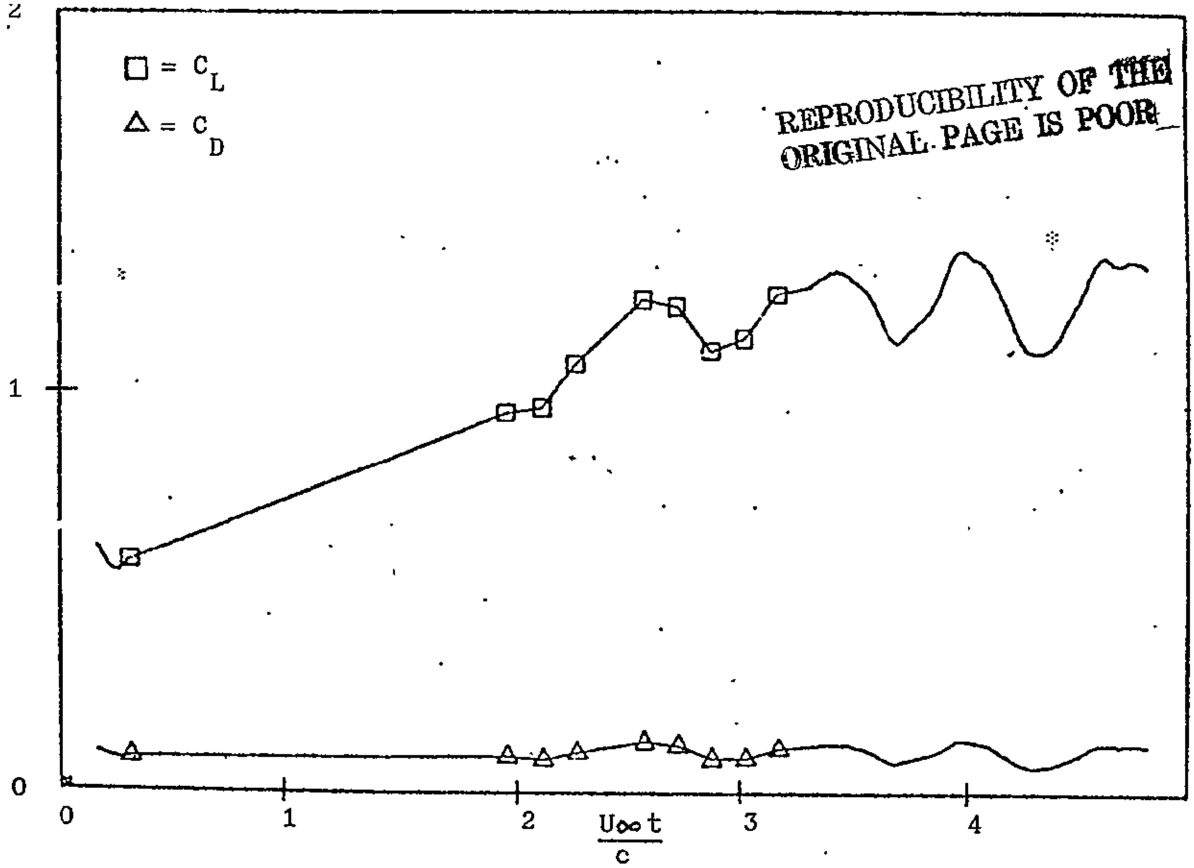


Fig 5. Time Dependent Variation of Force Coefficients for an Airfoil at Angle of Attack

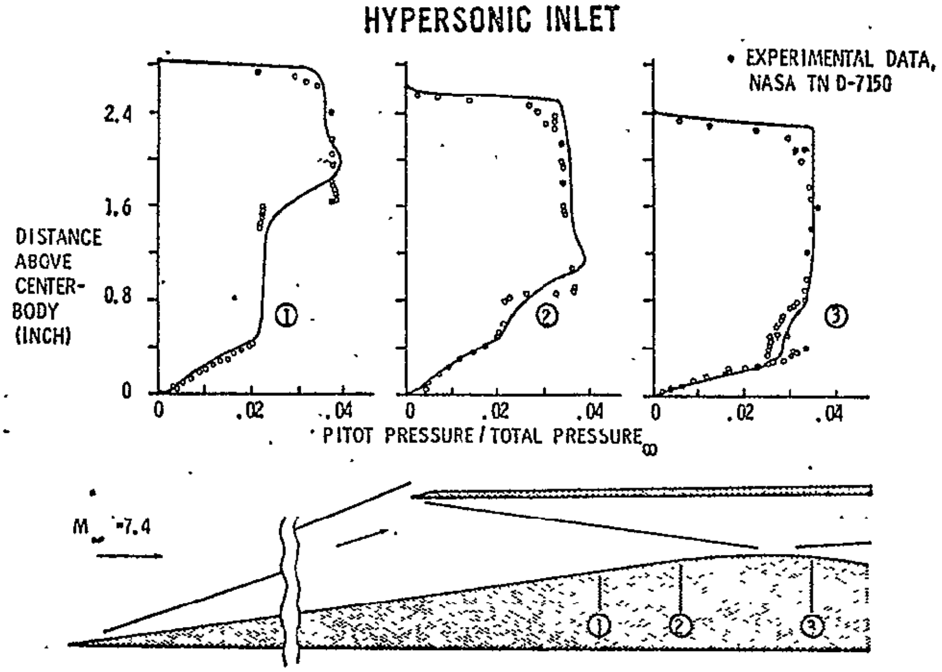


Fig 6. Pitot Pressure Distributions for a Hypersonic Inlet

D15

VISCOUS FLOW SIMULATION REQUIREMENTS

Julius E. Harris
Langley Research Center, Hampton, Virginia

INTRODUCTION

Simulation of two-dimensional compressible laminar viscous flows by numerically solving the compressible Navier-Stokes (N.S.) equations first began to appear in the literature during the mid 1960 time frame; since then significant advances have been made in this area of computational fluid dynamics (CFD). Research directed at the low Reynolds number (N_R), two-dimensional, incompressible laminar N.S. equations began much earlier and is still predominant in the literature today since the incompressible system is somewhat simpler to solve (for low N_R) and requires less computer resources than the compressible N.S. system. Reviews of the research area are presented in references (1) to (9). However, in spite of the research effort problem areas still remain to be solved before viscous flows requiring solution of the compressible N.S. equations can be efficiently and accurately simulated for flows of aerodynamic interest. These problem areas include turbulence (three-dimensional character), complex geometry, flow unsteadiness, placement of artificial boundaries relative to solid boundaries, specification of boundary conditions, and large flow gradients near surfaces and in the vicinity of shock waves for supersonic flows.

The cost of developing aircraft has risen dramatically over the past decade to the degree that it is estimated that approximately 100 million dollars of wind tunnel testing will be required in the 1980's for each new aircraft (ref. 10); it is obvious that this trend must be reversed. It appears that the only way that this trend can be reversed is by accelerating CFD capabilities for viscous flow simulation. The acceleration of CFD simulation depends upon (1) algorithm development coupled with (2) special purpose computers designed for processing these algorithms together with (3) coordinated programs (experimental/numerical) in turbulence closure techniques. The latter of these three research areas involves CFD studies in turbulence simulation with sub-grid scale closure, careful examination of modeled Reynolds stress equation closure concepts for separated three-dimensional flows, determination of the valid limits of algebraic closure concepts (eddy viscosity/mixing length) and "building-block" experimental programs for high Reynolds number, separated turbulent flows. The success achieved to date in simulation of turbulent boundary layer flows can be attributed to (1) the development of efficient implicit finite difference algorithms for solving the parabolic system of equations, (2) computer systems that efficiently and accurately process the resulting sequential codes, and (3) the large experimental data base available for developing/verifying the scalar eddy viscosity models for turbulence closure. It should be carefully noted that this data base is marginal for attached three-dimensional flows (ref. 11) and does not exist for three-dimensional flows with separation. The development of accurate turbulence closure models for three-dimensional separated flows appears at the present to be the main pacing item for aerodynamic simulation.

Considering the complex nature of general aerodynamic flows and the fact that the complexity in simulation is compounded by the interdependence of the various factors, one comes to the conclusion that no one single factor can be isolated and studied independently of the remaining factors. For example, it is absurd to evaluate the efficiency of a specific algorithm unless the evaluation is related to a specified computer architecture (parallel/pipeline/scalar, etc.). Transformation procedures employed to treat complex three-dimensional geometry cannot be evaluated independently of the viscous flow requirements which require careful placement of the grid points (nodes, for spectral methods) in order to capture the large gradients in regions of high shear (wall boundaries, shock waves, etc.) as well as minimize the number of required grid points. Consequently, while the purpose of the present paper is to address directly critical issues in flow simulation for flows with large regions of separation, it is not possible to accomplish this task without addressing to some degree the interrelationship between factors such as (1) transformation procedures for complex geometry, (2) coordinate systems and grid point distributions, (3) special requirements of flow regions with large gradients, (4) boundary placement and boundary condition specification, (5) algorithm structure and its relationship to (6) computer architecture, and (7) turbulence closure for three-dimensional, large N_R flows. The problems posed by the global nature of the pressure field for compressible subsonic and transonic flows is an area that has not received the required attention in CFD literature. Each of these problem areas will be addressed to some degree in the present paper while attempting to remain focused on large N_R turbulent flows with separated regions. Visual material used by the author during the workshop panel entitled "Viscous Flow Simulations" is presented in the Appendix of the present paper.

Transformation Procedures

One of the first and lasting impressions of the difficulty of three-dimensional flow simulation is the complex geometry associated with aerospace vehicles. Consequently, most of the CFD simulation research to date has centered on relatively simple geometrical shapes where coordinate lines could be chosen coincident with the boundary (see ref. 8, pp. 29-37). For these simplified geometric shapes it was generally possible to avoid interpolation between grid points not coincident with the boundary lines and thus avoid the introduction of interpolation errors into the region where the flow gradients were severe. Since the boundary conditions, especially on physical boundaries, are the dominant influence on the character of the solution, the use of grid points not coincident with the boundaries that required interpolation would place the most inaccurate difference representation in the region of maximum sensitivity. The generation of a curvilinear coordinate system with coordinate lines coincident with all boundaries thus becomes an important part of the simulation problem, especially for complex aerodynamic shapes. Such a system is often referred to in the literature as a "boundary-fitted" coordinate system.

The general method for generating a boundary-fitted coordinate system is to require that the coordinate lines be solutions of an elliptical partial differential system in the physical plane; Dirichlet boundary conditions are imposed on all boundaries. A method for the automatic generation of general two-dimensional curvilinear boundary-fitted coordinates is presented in reference (12). The curvilinear coordinate system will in general be nonorthogonal for the arbitrary spacing of the coordinate lines required in viscous flow simulation; however, the lack of orthogonality does not appear to present any serious problem in the specification of Neumann boundary conditions. However, the coordinate line stretching may introduce truncation errors due to the rapid variation of the coordinate line spacing in the physical plane.

The method of reference (12) has been applied successfully to two-dimensional flow simulation for multi-connected regions. The elliptic differential system for the coordinates are solved in finite-difference approximation by SOR iteration. The coordinate system can evolve with time without requiring interpolation of the dependent variables. Consequently, all computations can be performed on a fixed rectangular grid in the transformed plane without interpolation regardless of the time-history of the grid points in the physical plane.

The basic theory for the three-dimensional transformation is presented in reference (13). However, to date the method has not been carefully tested and will probably require detailed numerical experimentation on three-dimensional configurations before the desired grid distributions in the physical plane are achieved.

If simulation research is to be successful the three-dimensional body-fitted coordinate system will play an important role; research in this area must be continued. Careful assessment must be made of the truncation error effects introduced into the system by the coordinate line stretching in the physical plane.

Boundary Conditions

There appear to be two extreme philosophies concerning how much of the flow field surrounding a vehicle should be simulated by solving the N.S. equations: (1) only in regions where the N.S. equations are required, i.e., neighborhood of shocks, lee-side flows with separation, embedded subsonic regions, etc.; (2) use the N.S. equations for the complete configuration, i.e., enclose the vehicle in an elongated box. The former of these two extremes will most certainly require extremely complex logic with which the embedded regions could be isolated and enclosed in bounded regions. The interaction required between the boundary-layer like regions, N.S. regions, and external inviscid flow is at this point too complex to logically outline in diagram form for aerodynamic configurations. There is even some question as to whether such an approach would result in any saving of computer resources since for the two-dimensional compression corner with separation it has been shown to be more efficient to utilize the N.S. equations directly as opposed to the interactive procedures (ref. 14). The latter of the two extremes will without question require the most extensive computer storage ($0(10^9)$ grid points); however, in terms of computer time and manpower hours it may well be the most efficient of the two extremes. To date most flow simulations have involved solving the N.S. equations within truncated regions of the flow field as opposed to solving the complete flow field surrounding the aerodynamic vehicle. This course of action was chosen to reduce the computer resource requirements as well as simplify the problems associated with boundary conditions and geometry.

It is generally conjectured that the N.S. equations retain the mathematical properties of each of the individual equations in the set. Consequently, one can classify the set as hybrid parabolic-hyperbolic for unsteady flows and elliptic-hyperbolic for steady flows. The hyperbolic character is embodied in the continuity equation. The parabolic or elliptic character arises from the dissipative character of the remaining equations. For flow regions where dissipative effects are small (large N_R) the system tends to exhibit the characteristics of the Euler equations in regions removed from wall boundaries. The correct choice of boundary conditions depends upon the mathematical character of the equation set (higher order derivatives). Consequently, the global solution is a strong function of the dissipative terms even for large N_R separated flows where these terms are generally quite small. In general, the rigorous mathematical treatment of existence and uniqueness does not exist for a given set of boundary conditions and one is forced to rely almost entirely on heuristic arguments.

The specification of computational domains and their required boundary conditions for two-dimensional flows is presented in reference (8) (see also ref. (9), pp. 261-286); a detailed discussion of the material presented in reference (8) is beyond the scope of the present paper. However, it is important to note that most of the two-dimensional problems solved to date have had the following character: (1) truncate the flow field and bound only that part of the flow where the N.S. equations are required such that boundary-layer like flow occurs both upstream/downstream with supersonic external flow; (2) enclose the entire body being careful to place the downstream boundary sufficiently far from infinity so that infinity flow conditions have not been reached, but far enough removed from the body for its upstream influence to be negligible. Experience gained to date in numerically treating two-dimensional separation will be of value for general three-dimensional separation; however, the latter is much more complex and less understood (ref. 15).

For three-dimensional flows the option to isolate and bound only those regions of the flow field where the N.S. equations are required (as opposed to bounding the entire body) will result in extremely complex logic for specifying the boundary conditions over this bounding surface. Exceptions may be simple reentry type vehicles where separation occurs only on the lee surface or in the region of control devices. In general, for complex aerodynamic configurations the boundary conditions would depend upon solutions of boundary-layer like equations that had been interacted with the external flow field. For steady flow fields this option might be possible provided one could develop the logic to isolate these regions (highly doubtful); however, for unsteady flows this option appears to be impractical if not impossible. Consequently, it appears that the only current option is to enclose the entire vehicle and specify the boundary conditions on this closed surface.

Algorithm Selection

Based on current usage for two- and three-dimensional viscous flow simulation, only finite-difference methods can currently be considered as candidates for implementation on the proposed special-purpose computer. Integral methods, finite-element methods, and spectral methods have not been sufficiently tested to date for the compressible N.S. equations to be considered as possible candidates for a special-purpose computer for aerodynamic simulation. Candidate finite-difference methods can be explicit, implicit, or mixed explicit-implicit in character. If the flow under study is unsteady, then the numerical scheme must be consistent with the exact unsteady equations and sufficiently accurate in both time and space. For flows where turbulence closure is provided by either modeling or solving the Reynolds stress equations, the method must be a minimum of second order accuracy in time and space; whereas, for turbulence simulation with sub grid scale closure, fourth-order accuracy in space is required. If the flow under study is steady, then the numerical scheme need not be consistent with the unsteady equations unless the transient solution is of physical interest. The only requirement for the method is that it yield a steady solution for large time which is an approximation to the solution of the steady-state equations (N.S. equations with time derivatives equated to zero). There are several advantages to using nonconsistent schemes: (1) large time steps in comparison to a consistent scheme which results in (2) faster convergence to steady state. However, for large N_R three-dimensional viscous flow simulation for aerodynamic flows the method should be consistent with the exact unsteady equations since most flow fields will in general have embedded regions of unsteady flow.

Finite element methods. - Finite-element methods have received increasing attention in the literature over the past five year period as a possible substitute for finite-difference methods in fluid mechanics. The utility of the finite-element method for viscous-flow simulation has been questioned from several viewpoints (for example, see ref. 16). The most frequent claims of finite-element methods are: (1) elements can be fitted to irregular boundaries; (2) "natural" treatment of boundary conditions. In practice neither of these claims has proven to be true. The development of boundary-fitted coordinate systems (refs. 12 and 13) has essentially removed the problems associated with irregular boundaries for finite-difference methods. Furthermore, while in principle natural boundary-condition treatment may be possible in the finite-element method (problem dependent) it has not been so in practice (see ref. 16, pp. 233). One of the primary problems associated with the finite-element method is the complex matrix equations resulting from the formulation. Consequently, the method has large computer resource requirements (storage/processing time) in comparison to finite-difference methods. The complexity of the finite-element method as compared to finite-difference methods for the two-dimensional compressible N.S. equations is shown in reference (17).

Spectral methods. - Spectral methods are relatively new and have not been sufficiently tested for compressible viscous flow simulation; however, the method has been applied to incompressible flows with success (refs. 18 to 21). The method is optimum for flows with periodic boundary conditions (FFT), but the complex boundary shapes associated with flows of aerodynamic interest present problems. For more details the reader is referred to references (22) and (23).

Integral methods. - Integral relation procedures have been used extensively over the years for both inviscid and parabolic boundary-layer like flows; however, the methods do not appear feasible for the N.S. equations and to the author's knowledge there have been very few attempts to apply the method to the compressible N.S. equations (refs. 24 and 25).

The selection of the "class" of solution procedures, based on current experience then appears to be limited to finite-difference procedures. The potential error in this selection process centers around what is not known about the rapidly advancing state-of-the-art of algorithms. For example, if one had been faced with the decision prior to the publication of reference (26) the choice would still have been a finite-difference technique of the Lax — Wendroff type, but the subsequent advancements (ref. 27) made in the following few years would have negated this selection. The intensive research on algorithm developments and/or improvements in existing algorithms is far greater today than in the early 1970 time frame. Consequently, it is difficult to envision the state of the art in the mid 1980's. It is important that the process required to develop and test a special purpose computer for viscous flow simulation be initiated today if it is to have the desired impact on the aerodynamic design process by the mid 1980's; however, it is even more important that the resulting product not be a dinosaur incapable of evolving with the advancing state-of-the-art of solution procedures.

Finite-difference methods. - A review of the finite-difference schemes that have been applied to the two-dimensional compressible N.S. equations is presented in reference (7): both one-step and two-step methods are discussed for consistent and non-consistent schemes. The two-step scheme introduced by MacCormack (ref. 26) has been used extensively and has experienced several important modifications. The most important of these modifications were: (1) introduction of the splitting concept (ref. 27) originally introduced by Peaceman and Rachford (ref. 28) to replace the complex operators by a sequence of simpler ones while maintaining second-order accuracy as well as allowing larger Δt increments as compared to the original unsplit scheme; (2) splitting the equations into hyperbolic part with an explicit method based on characteristic theory and the parabolic part with an implicit method requiring simple tridiagonal inversion (ref. 29).

The "current" MacCormack scheme (ref. 29) yields computer time reductions of up to two orders of magnitude as compared with the earlier time split version. This increase in computational efficiency occurs with increasing N_R (see fig. 7, p. 16, ref. 29) as would be expected. With increasing N_R

the solution domain becomes less viscous dominated; consequently, the severe CFL limitation present in the former methods resulting from the fine grid distributions (Δy) required by the severe velocity gradients in the viscous region was replaced with an implicit boundary-layer like procedure having time steps that are orders of magnitude larger than those imposed by the CFL stability criteria.

The approximation of $v/c << 1$ required for the characteristic equation in reference (29) appears to be a severe penalty for general flows where v/c may be of $O(1)$. Shang (ref. 30) made an additional modification that eliminates (1) MacCormack's equation splitting between the inviscid and viscous terms and (2) the $v/c << 1$ restriction. Consequently, the method (ref. 30) appears to be simpler in structure and less restricted in its range of application for large N_R flows as compared with the method originally developed by MacCormack (ref. 29).

Implicit-finite difference methods have been extensively used for boundary-layer like flows (ref. 31) because of their stability characteristics; however, large arithmetic operation counts are required per incremental time step as compared with the explicit methods. For large N_R viscous flows the penalty of the large number of arithmetic operation counts per time step generally offsets the advantage of the larger time step allowed by the implicit methods in comparison with explicit schemes. However, improved non-iterative algorithms coupled with the trend of current computer hardware development has resulted in the development and implementation of efficient implicit procedures for the N.S. equations. Beam and Warming (ref. 32) present an efficient implicit finite-difference procedure for the Euler equations (inviscid) and have extended the procedure to the compressible Navier-Stokes equations (ref. 33). The extended algorithm is noniterative and retains the required conservation-law form. The method is a three-level scheme that requires only two levels of data storage. The three-level scheme effectively treats the cross derivative terms such that the unconditional stability of the algorithm is retained. The method has been applied to flat-plate shock-boundary-layer interaction (ref. 33) and compares favorably with the MacCormack rapid solver (ref. 29) (see fig. 5, ref. 33). From the data presented in reference (29) it is not possible to make comparisons of time required for solution between the two methods; however, the implicit method required less than 100 time steps to steady state at a maximum Courant number of approximately 170.

Steger (ref. 34) utilized the implicit method of reference (33) together with the boundary-fitted curvilinear coordinate system generation procedure of reference (12) to simulate the unsteady viscous flow field for two-dimensional airfoils. In the author's opinion this particular publication is one of the most, if not the most significant publication that has occurred to date for viscous flow field simulation. The approach presented in reference (34) with accurate turbulence closure may well be the candidate algorithm for the proposed special purpose computer for viscous flow simulation; the extension of the algorithm to three-dimensions appears to be reasonably straightforward.

Significant progress has been made in the simulation of viscous flows with simple turbulence models for relatively simple two-dimensional geometries where the boundary conditions could be correctly specified; this progress has been accelerated with the introduction of transformation procedures, especially for two-dimensional airfoils. However, the literature is sparse in three-dimensional applications (see ref. 8, pp. 35-37). The finite-difference algorithms previously discussed can be extended to three-dimensional flows provided the computer resource requirements are available. In 1977 two papers appeared in the literature to-date dealing with supersonic laminar flow over three-dimensional compression corners. Shang and Hankey (ref. 35) applied the finite-difference scheme of reference (36) to the problem cast in similarity coordinates. Convergence required 10 hours of CDC-7600 time for a 8 x 32 x 36 grid for 6000 time steps. Hung and MacCormack (ref. 37) solved the N.S. equations using the algorithm presented in reference (29) for a 30 x 30 x 30 grid in 1.2 hours on a CDC-7600 for 300 time steps. The method is now being extended to turbulent flow through scalar algebraic closure (ref. 38). Hopefully, the algorithm presented in reference (33) and applied to two-dimensional airfoils in reference (34) will also soon be applied to three-dimensional flows.

It then appears that no rational decision can currently be made pertaining to "the candidate" algorithm for the special purpose computer for large N_R aerodynamic flow simulation; one cannot eliminate either of the algorithms presented in references (29) and (33). However, if the system is hard-wired to optimally process mixed explicit-implicit procedures then it cannot be optimum for processing fully implicit schemes. Hopefully as the design procedure evolves the following two processes will occur: (1) accelerated development and testing of the algorithms (refs. 29, 33) for aerodynamic shapes requiring boundary-fitted curvilinear coordinate systems with careful evaluation of turbulence closure on performance; (2) a decision to maintain sufficient flexibility for programming the proposed special purpose computer so that advances in algorithm development can be implemented at minimum cost.

Turbulence

Turbulence closure is the most difficult problem area associated with large N_R viscous flow simulation. The increasingly finer length scales that develop for large N_R three-dimensional separated flows is the major pacing area for three-dimensional viscous flow simulation. These length scales must be properly treated if the simulation is to be correct. In principle turbulence can be numerically simulated without approximation from the time-dependent Navier-Stokes equations: this is not currently possible nor will it be possible in the foreseeable future. The resolution of all the scales of motion would require $O(N_R{}^{9/4})$ independent variables in time for which $O(N_R{}^3 \ln N_R)$ arithmetic operations would be required (ref. 39). Since the viscous aerodynamic simulation of problems of practical interest involves $N_R > 10^6$ the requirements are beyond projected computer system capabilities.

Turbulence simulation with sub-grid scale closure. - A possible but complex approach to turbulence closure is to utilize turbulence simulation with sub grid scale closure. In this approach the large-scale turbulence structure is obtained numerically from the time-dependent Navier-Stokes equations with appropriate models for the small-scale structure. This area of research is of fundamental importance since it provides bench-mark results against which more approximate modeling concepts can be compared and/or developed. To date, the concept has been partially successful only for low N_R, incompressible free flows. It is possible that certain compressible flows could be treated on the CDC STAR-100 system; however, it may well be that a special purpose computer system will have to be developed and dedicated to this area of CFD research.

Scalar closure. - The scalar, algebraic closure concept (eddy viscosity/mixing length) has been used with limited success for two-dimensional separated flows (see ref. 40). The use has been justified in part by the experimental data base developed for two-dimensional boundary layer flows and in part on being the only option available in relation to current computer limitations. The algebraic concepts are attractive from the viewpoint of the N.S. equations since they modify the system only through the addition of effective viscosity and conductivity terms, each of which tends to make the system more diffusive in character. However, the concept does not reflect the physical characteristics of the flow (for example, the nonequilibrium character in the vicinity of strong interactions) and cannot be extended to general three-dimensional large N_R flows with separation. Recent studies have shown that the concept is even highly suspect for attached three-dimensional boundary layer flows (ref. 41).

Two equation models. - Two equation turbulence closure models provide a possible approach to remove the obvious limitations associated with the scalar eddy-viscosity/mixing-length formulations without adding greatly to the complexity of the equation system. Second-order closure two equation turbulence models utilize two parameters to characterize the turbulence and define the eddy diffusivity: each parameter satisfies a nonlinear diffusion equation. Limited success appears to have been achieved for a wide variety of flows where conventional mixing length approaches have failed; for example, boundary layer separation (ref. 42) and transition (ref. 43). However, problems associated with the length scale equation (ref. 44) appear to limit the potential success of the approach; also, the near-wall region presents a severe problem since first-order wall models are generally used. The compilation of papers presented in reference (45) indicates that the two-equation model can provide adequate precision for many engineering applications; however, the approach does not yield the detailed physics of the flow (for example, see pp. 13.35-13.45, ref. 45) required for aerodynamic flow simulation. Considering the wide range of length scales present in three-dimensional large N_R separated flows together with the highly elliptic character of such flows, there appears to be little if any promise of utilizing the two-equation models for the simulation of general aerodynamic flows (a minimum of one additional Reynolds stress term must be modeled for three-dimensional flows).

Modeled Reynolds stress equations. - The modeled Reynolds stress equations currently appear to be the most promising means by which the problems associated with the scalar eddy viscosity/mixing length and two-equation models can be circumvented. However, the system results in a total of seven additional differential equations that must be solved with the averaged N.S. equations (Reynolds equations): a system of 12 equations in 12 unknowns. Furthermore, the "constants" appearing in the system (6-Reynolds stress equations; 1-dissipation equation) have not been shown to be universal and must be modeled by careful comparison of numerical results with experimental data; unfortunately, the required experimental data base for three-dimensional turbulent flows with large separated regions of flow does not exist.

The set of twelve governing equations, assuming that the modeling constants for the Reynolds stress and dissipation equations are known to a sufficient degree of accuracy presents a numerical problem in itself from the viewpoint of developing a special purpose computer system since they introduce stiffness into the system of equations. The stiffness is introduced into the system through the dissipation equation due to the sensitivity and interdependence of the dependent variables. Discussions of the Reynolds stress closure concept are presented in references (44), (46), and (47).

Computer System Architecture

To achieve the required processing speed and high-speed memory required for meaningful aerodynamic simulation the computer system architecture must be highly specialized. This improvement in speed will result from parallelism which is strongly dependent on software and the nature of the N.S. equations. It appears that the major problem that must be faced is not the design and/or cost of the processors: the primary problem is sufficient high-speed memory carefully matched to the processor speed.

Assuming that the algorithm chosen to solve the Navier-Stokes equations could be exploited to take maximum advantage of parallel architecture, then it follows that the system (algorithm plus architecture) could efficiently simulate three-dimensional, large N_R separated flows utilizing the averaged N.S. equations (Reynolds equations) with Reynolds-stress closure. As previously noted, the stiffness introduced through the dissipation equation would decrease the efficiency. However, for turbulence simulation where at a minimum second-order time and fourth-order space resolution with negligible phase error is required, it appears that the system designed for Reynolds-stress closure would not be optimum. Consequently, it appears that a minimum of two special architectures may be required; one for large N_R

aerodynamic flow simulation with Reynolds-stress equation closure and another for turbulence simulation with sub-grid turbulence closure. The projected cost of these special purpose systems is high (see ref. 39, pp. 41-52); consequently, care must be exercised to make certain that special purpose system(s) are as flexible as possible without compromising their performance to the degree that they approach large general purpose computer architecture. Several recent papers have been presented where design techniques promise the potential of reducing the cost associated with special purpose systems (refs. 48 and 49).

If one reviews the rapid evolution of algorithms for the two- and three-dimensional N.S. equations over the past decade, the doubt naturally arises as to whether a special purpose computer can be designed to adequately treat (grow with algorithm development) the potential algorithm improvements over the next decade (1977-1987). This poses a potentially serious problem in light of the large expense associated with the development of special purpose systems. The algorithm development/refinement that has taken place over the past decade has resulted from having to do the job on computer systems of the CDC-6600 and 7600 class; that is, systems with marginal speed and high-speed memory for two- and three-dimensional N.S. flows. However, the limitations imposed by the available computer systems resulted in research to do the job more efficiently within the constraints imposed by the existing and/or available computer systems. This work was carried out on serial machines that process and advance the data in a sequential mode (point by point) and as such complex boundary conditions could be efficiently studied together with modifications to the basic algorithm structure. It is important that we retain this capability on the proposed special purpose computer since complicated boundary conditions cannot be efficiently treated by efficient parallel procedures; it is also important that basic algorithm development continue and not be restricted to a single specialized architecture. Consequently, it is reasonable to project that general purpose scientific computers comparable to today's CDC-7600 will continue to be used for the foreseeable future, since good techniques still need to be made better and because the variety of problems is too diversified to specialize on one system architecture. The flexibility, programmability and inventory of software also dictates this conclusion. Furthermore, it is highly probable that large general purpose computers will be used in conjunction with the proposed special purpose machines.

The large general purpose computer still has a definite role to play in CFD development as well as complex viscous flow simulation. Basic ideas must first be developed and tested in order to evaluate their potential success for special-purpose machines. An advanced system like the CDC-7600 but with 10^6 high speed memory would fill these requirements and could be operated in either the sequential or vector mode; such a system would be an asset to the aerospace and basic research community for the foreseeable future. The system would foster the continued development of algorithms and applied codes for the aerospace industry thus leaving the proposed special purpose computer free for accelerated flow simulation research.

CONCLUDING REMARKS

The advances in CFD over the past decade clearly indicate that the computer will play an increasingly important role in reducing the cost and time associated with new aircraft development; this reduction will come through the ability to numerically simulate increasingly more complex three-dimensional viscous flows. The acceleration of our current ability to efficiently treat viscous flow simulation depends upon not only the development of more advanced specialized computer systems, but also upon a dedicated program of basic applied mathematics. It would be a serious error in judgment to assume that any of the numerical procedures now existing can efficiently (efficient in relation to potential developments) treat separation at large N_R or that our understanding of turbulence is sufficient to describe the complex flow. Consequently, the large general purpose computer still has a major role to play in the foreseeable future before maximum benefits can be obtained from any special purpose computer. Hopefully the developing microcomputer technology can do much to reduce the expense associated with this evolving process. In the near future it may be possible to interconnect hundreds or thousands of microprocessors into arrays of stand-alone systems dedicated to special problems as well as use them to augment the computational power of large computers.

Large N_R, three-dimensional viscous flow simulation with separation cannot be adequately treated without carefully addressing the three-dimensional turbulent character of the flow. The success enjoyed in two-dimensional turbulent boundary-layer simulation through first-order closure occurred because the assumptions made in the scalar eddy-viscosity models were not all that physically incorrect for quasi-parallel flows as well as the existence of an extensive experimental data base from which one could verify the modeling constants for various flow conditions. However, this success cannot be directly extended to general three-dimensional flows with separation since turbulence cannot be treated as a scalar quantity; also, as of this date the data base for three-dimensional flows does not exist. Consequently, success in three-dimensional viscous flow simulation depends strongly upon developing active experimental programs that are adequately funded and staffed with qualified experimentalists. The development of a special purpose computer (or computers) for large N_R three-dimensional flow simulation with separation will be of little real value unless experimental research in three-dimensional flows is accelerated.

In conclusion, as one reviews the current CFD literature it appears that there is an underlying belief held by some that faster, bigger and more specialized computer systems will provide the solution to the difficulties associated with three-dimensional large N_R viscous flow simulation; this is in part a delusion. It is agreed that larger, faster and more specialized machines are needed simply due to the large number of grid points required to adequately describe flow fields of aerodynamic interest; however, it should also be clearly understood that specific areas such as algorithm development (stability, accuracy, etc.), coordinate systems, and turbulence closure still require concentrated research effort before any dedicated special purpose "super computer" for viscous-flow simulation can have any real impact on the aerospace industry.

APPENDIX

Visual Material for Viscous Flow Simulation Panel

The material contained in the present Appendix was used during the oral presentation for the panel entitled "Viscous Flow Simulations."

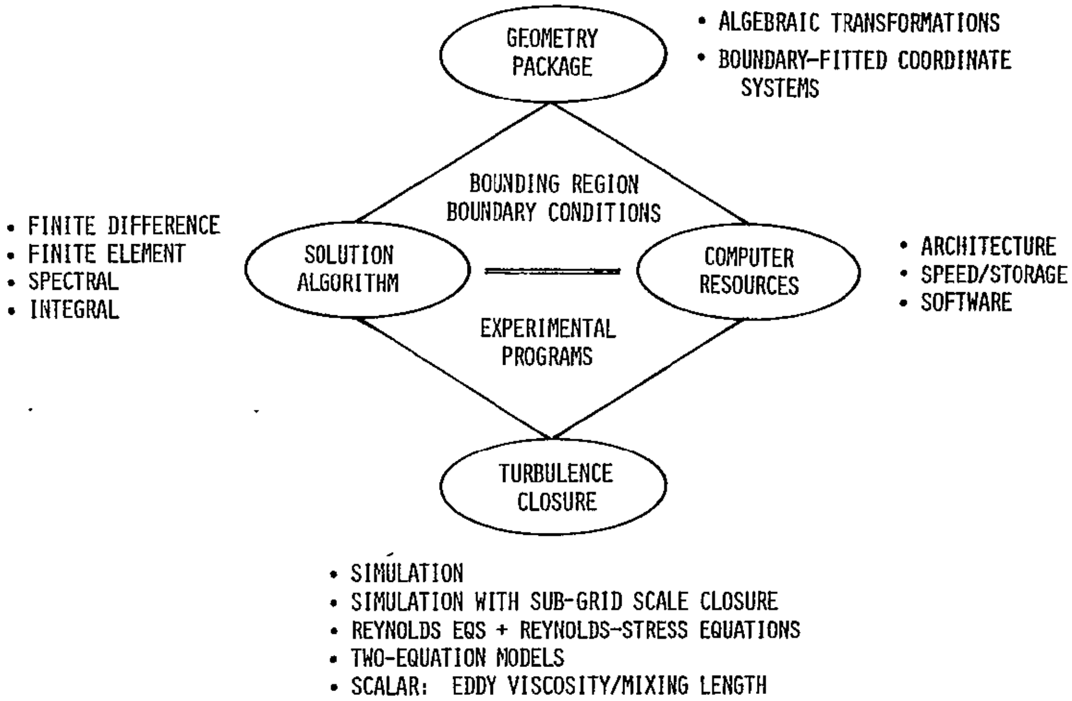


Figure 1. - The elements of three-dimensional viscous flow simulation.

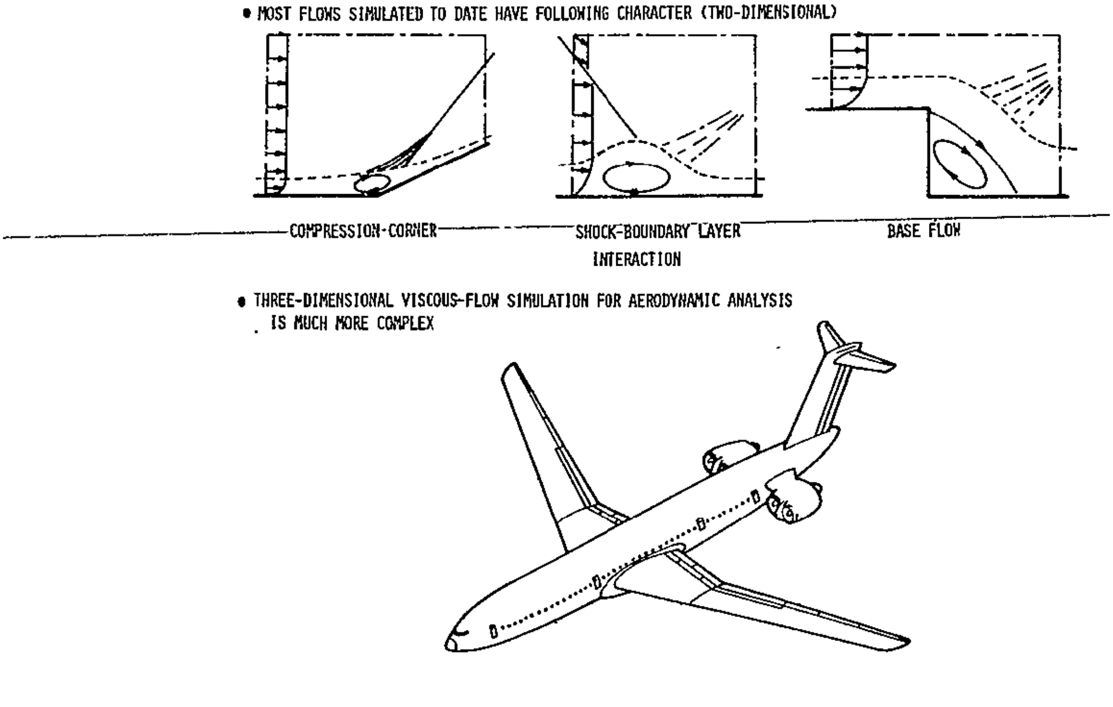
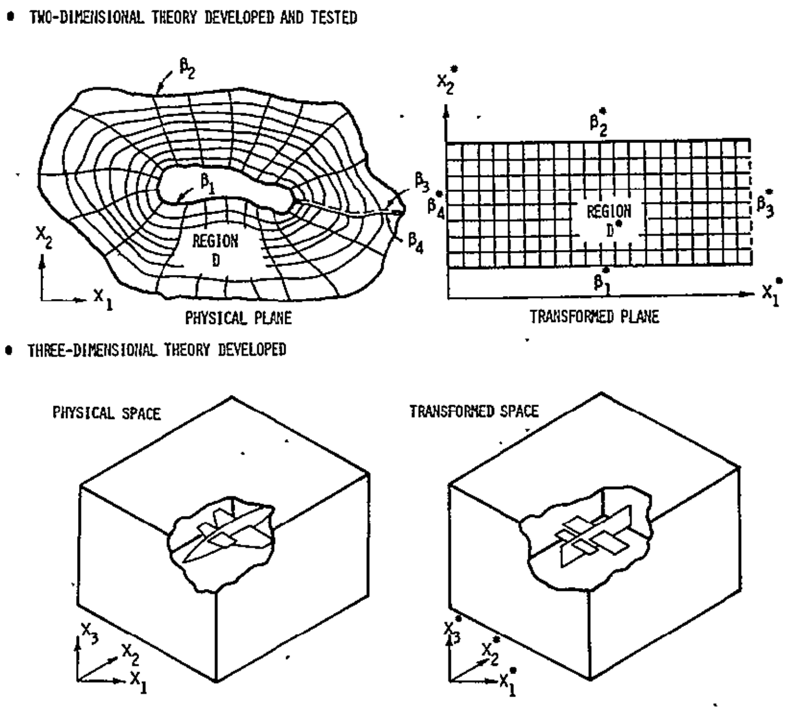

• MOST FLOWS SIMULATED TO DATE HAVE FOLLOWING CHARACTER (TWO-DIMENSIONAL)
COMPRESSION-CORNER
SHOCK-BOUNDARY LAYER INTERACTION
BASE FLOW
• THREE-DIMENSIONAL VISCOUS-FLOW SIMULATION FOR AERODYNAMIC ANALYSIS IS MUCH MORE COMPLEX

Figure 2. - Geometry.



- TWO-DIMENSIONAL THEORY DEVELOPED AND TESTED
- THREE-DIMENSIONAL THEORY DEVELOPED

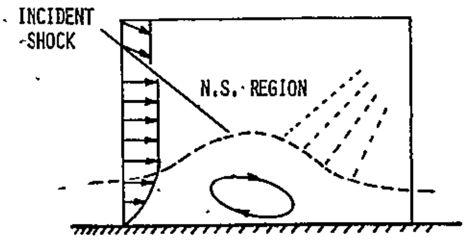
Figure 3. - Body-fitted curvilinear coordinate systems

- TWO-DIMENSIONAL FLOW
  - EASILY DEFINED
  - WALL VELOCITY GRADIENT VANISHES : $\frac{\partial u}{\partial y} = 0$ AT SURFACE
  - FLOW MAY/MAY NOT REATTACH (CLOSE) ON BODY

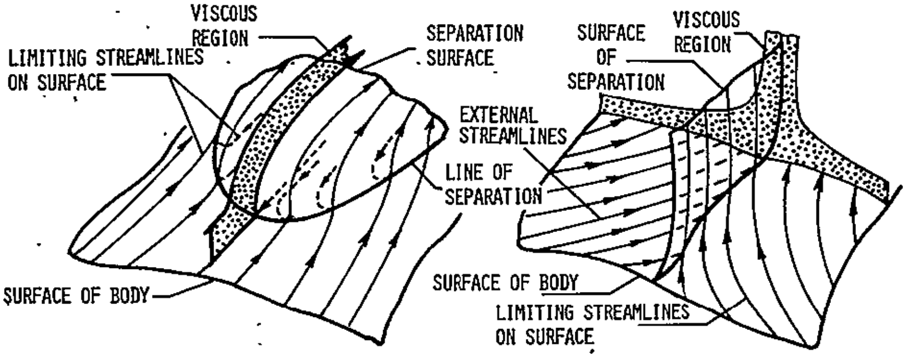
- THREE-DIMENSIONAL FLOW
  - SURFACE SHEAR NOT NECESSARILY ZERO
  - VELOCITY GRADIENT NORMAL TO SEPARATION LINE VANISHING IS A NECESSARY BUT NOT SUFFICIENT CONDITION FOR SEPARATION
  - TWO TYPES: BUBBLE; FREE SHEAR LAYER

(a) Basic definitions.

Figure 4. - Boundary-layer separation.



TWO-DIMENSIONAL SHOCK-BOUNDARY LAYER INTERACTION



BUBBLE SEPARATION

FREE SHEAR LAYER SEPARATION

(b) Three-dimensional separation.

Figure 4. - Concluded.

ACCEPTABLE TRUNCATION
- TURBULENCE CLOSURE: $O(\Delta t^2, \Delta X_i^2)$
- SIMULATION- SGC : $O(\Delta t^2, \Delta X_i^4)$ -- MAY BE OPTIMUM

● FINITE DIFFERENCE
- EXPLICIT (CFL LIMIT; EASY TO CODE; LOW STORAGE
- IMPLICIT (Δt>CFL; MORE COMPLEX CODE)
- MIXED EXPLICIT/IMPLICIT (TAKE ADVANTAGE OF FLOW CHARACTER)

} EXTENSIVE EXPERIENCE

● FINITE ELEMENT
- COMPLEX MATRIX ALGEBRA
- EASE OF TREATMENT OF COMPLEX BOUNDARIES
- NATURAL TREATMENT OF BOUNDARY CONDITIONS

} NOT PROVEN IN PRACTICE

● SPECTRAL
- CURRENTLY LIMITED TO SIMPLE GEOMETRY
- NATURAL TREATMENT OF BOUNDARY CONDITIONS
- POTENTIAL ACCURACY $\geq O(\Delta t^2, \Delta X_i^4)$
- EXCELLENT RESOLUTION IN REGIONS OF HIGH SHEAR

} INCOMPRESSIBLE FLOWS

● INTEGRAL
-LIMITED APPLICATIONS IN LITERATURE

} INSUFFICIENT APPLIED EXPERIENCE

Figure 5. - Solution procedures.

	ASSUMPTIONS	DIFFICULTIES	SUCCESSES
TURBULENCE SIMULATIONS (DIRECT TIME-WISE INTEG. OF 3-D NAVIER-STOKES EQS. FOR SMALL $\Delta t, \Delta x$)	CLOSURE SCHEME REQD. FOR SCALES SMALLER THAN GRID SPACING (SUB-GRID-SCALE)	-HUGE MACHINE TIME/STORAGE -ONLY LOW RE NO., SIMPLE GEOMETRY, DUE TO MACHINE SIZE & LIMITATIONS -REQUIRES TOP NOTCH NUMERICAL TALENT, NEED SMALL PHASE ERROR, EXCELLENT ACCURACY	ISENTROPIC & HOMOGENOUS SHEAR FLOWS, LOW SPEED, BOX-TYPE GEOMETRY RESULTS SHOW CLOSURE "CONSTANTS" IN 2ND ORDER METHODS REALLY "VARIABLES," EVEN FOR THESE SIMPLE CASES.
"REYNOLDS STRESS" EQUATIONS ($\overline{u'_i u'_j}$) (OBTAINED BY TAKING MOMENTS OF NAVIER-STOKES EQS. & REYNOLDS AVERAGING)	3RD-ORDER CORRELATIONS, PRESS. FLUCT., LENGTH SCALE EQ. TERMS CAN BE MODELED WITH "CONSTANT" COEFFICIENTS	-"CONSTANTS" VARY FROM FLOW TO FLOW -REAL DIFFICULTY (KNOWN FOR LAST 5 YEARS) IS FORM AND MODELING OF LENGTH-SCALE EQUATION -STIFF EQ. SYSTEM	GIVES REASONABLE ANSWERS FOR SEVERAL TYPES OF SEPARATED FLOWS, FURTHER DATA REQD. FOR GOOD EVALUATION

(a) Part 1.

Figure 6. - Turbulence modeling.

	ASSUMPTIONS	DIFFICULTIES	SUCCESSES
"TWO EQUATION MODEL" (USES TURB. KIN. EN. AND LENGTH SCALE EQS.)	$\overline{u'v'} = F(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}, L)$ PLUS USUAL MODELS FOR THIRD-ORDER TERMS	- 8 YEARS OF EXPERIENCE INDICATES LENGTH SCALE EQ. IS MAJOR SOURCE OF INACCURACIES - NOT SUITABLE FOR 3-D FLOWS	EVIDENTLY SUITABLE FOR MANY 2-D SEPARATED FLOWS EXCEPT NEAR WALLS
FIRST ORDER OR MIXING LENGTH CLOSURE	$\overline{u'v'} = F(\bar{u})$, LENGTH SCALE SPECIFIED FROM MEAS., PHYSICS	FOR SIMPLE FLOWS ONLY, LENGTH SCALE MUST BE WELL BEHAVED	EXCELLENT FOR MOST QUASI-PARALLEL SHEAR FLOWS, LARGE QUANTITY OF DATA FOR 2-D QPSF'S ALLOWS INCLUSION OF u'_∞, ROUGHNESS, DP/DX, $\dot{M}_W$, ETC., EFFECTS

NOTE: PROBLEM IN TURBULENCE MODELING FOR NON-QUASI-PARALLEL FLOWS IS SORTING OUT NUMERICAL AND TURBULENCE MODELING INACCURACIES.

(b) Part 2.

Figure 6. - Concluded.

ADVANTAGES	DISADVANTAGES
● SCALAR	
+ NO ALGORITHM LIMITATIONS + SOFTWARE WELL UNDERSTOOD/DEVELOPED + EVOLUTIONARY DEVELOPMENT ALLOWED	- CPU SPEED LIMITATION
● PARALLEL	
+ HIGH CPU SPEED FOR APPROPRIATE PROBLEMS	- ALGORITHMS NOT WELL DEVELOPED - SOFTWARE NOT UNDERSTOOD WELL - REQUIRES REVOLUTIONARY DEVELOPMENT
● PIPELINE	
+ SOFTWARE PROBLEMS NOT AS BAD AS PARALLEL BUT WORSE THAN SCALAR + CPU SPEED INCREASES WITH MULTIPLE PIPES BUT APPROACHES PROBLEM AREA OF PARALLEL	- CPU SPEED FROM SINGLE PIPE LIMITED
● MINI/MICRO	
+ LOW COST + POTENTIAL HIGH PERFORMANCE	- ORGANIZATION PROBLEMS - SOFTWARE DIFFICULT

Figure 7. - Computer architecture.

- SPECIAL PURPOSE COMPUTERS WILL HAVE AN INCREASINGLY IMPORTANT ROLE IN REDUCING THE COST/TIME ASSOCIATED WITH NEW AIRCRAFT DESIGN
- SUCCESS OF SPECIAL PURPOSE SYSTEM DEPENDS UPON:
  - ACCURATE TURBULENCE CLOSURE
  - EFFICIENT/ACCURATE TREATMENT OF COMPLEX GEOMETRY
  - ALGORITHM DEVELOPMENT FOR PARALLEL PROCESSING
  - HIGH-SPEED PARALLEL PROCESSOR WITH MATCHED, LARGE INCORE, HIGH-SPEED MEMORY
- DESIGN WITH FLEXIBILITY TO AVOID AN EXPENSIVE DINOSAUR
- LARGE GENERAL PURPOSE SYSTEMS WILL BE REQUIRED FOR THE FORESEEABLE FUTURE
- MICRO/MINI SYSTEMS REPRESENT AN AREA WHERE ADDITIONAL RESEARCH IS REQUIRED (POTENTIAL IS HIGH)

Figure 8. - Recommendations.

REFERENCES

1. Cheng, S. I.: Numerical Integration of Navier-Stokes Equations. AIAA Journal, Vol. 8, No. 12, 1970, pp. 2115-2122.

2. Wirz, H. J.; and Smolderen, J. J.: Numerical Integration of Navier-Stokes Equations. AGARD Lecture Series No. 64 on Advances in Numerical Fluid Dynamics, 1973, pp. 3.1-3.13.

3. Cheng, S. I.: A Critical Review of the Numerical Solution of Navier-Stokes Equations. Lecture Notes, Progress in Numerical Fluid Dynamics, von Karman Institute, Rhode-St-Genèse, Belgium, Jan. 1974.

4. Taylor, T. D.: Numerical Methods for Predicting Subsonic, Transonic, and Supersonic Flow. AGARDograph No. 187, January 1974.

5. Krause, E.: Numerical Solution of the Navier-Stokes Equations. Lecture Notes, International Center for Mechanical Sciences, Udine, Italy, October 1974.

6. Peyret, R.; and Viviand, H.: Resolution Numerique des Equations de Navier-Stokes pour les Fluides Compressibles. Lecture Notes in Computer Science, Vol. 11, pp. 160-184, Springer Verlag, 1974.

7. Peyret, R.; and Viviand, H.: Numerical Solution of the Navier-Stokes Equations for Compressible Fluids. AGARD Lecture Series No. 73 on Computational Methods for Inviscid and Viscous Two- and Three-Dimensional Flows, 1975, pp. 6.1-6.14.

8. Peyret, R.; and Viviand, H.: Computation of Viscous Compressible Flows Based on the Navier-Stokes Equations. AGARDograph No. 212, 1975.

9. Roache, Patrick J.: Computational Fluid Dynamics. Revised Printing, Hermosa Publishers, 1976.

10. Chapman, Dean R.; Mark, Hans; and Pirtle, Melvin W.: Computers vs. Wind Tunnels for Aerodynamic Flow Simulations. Astronautics and Aeronautics, April 1975, pp. 22-35.

11. Johnston, James P.: Experimental Studies in Three-Dimensional Turbulent Boundary Layers. Proceedings of the Lockheed-Georgia Company Viscous Flow Symposium, June 1976, pp. 239-289.

12. Thompson, Joe F.; Thames, Frank C.; and Mastin, C. Wayne: Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies. NASA CR-2729, July 1977.

13. Mastin, C. Wayne; and Thompson, Joe F.: Transformation of Three-Dimensional Regions Onto Rectangular Regions by Elliptic Systems. ICASE Report No. 76-13, April 1976.

14. Rose, William C.: Practical Aspects of Using Navier-Stokes Codes for Predicting Separated Flows. AIAA Paper No. 76-96, January 1976.

15. Peake, D. J.: Controlled and Uncontrolled Flow Separation in Three-Dimensions. National Aeronautical Establishment, Ottawa, Canada, Aeronautical Report LR-591, July 1976.

16. Roache, Patrick J.: Recent Developments and Problem Areas in Computational Fluid Dynamics. Lecture Notes in Mathematics, Computational Mechanics, 461, pp. 195-256.

17. Cooke, C. H.: A Numerical Investigation of the Finite-Element Method in Compressible Primitive Variable Navier-Stokes Flow. Department of Mathematics and Computing Sciences, Old Dominion University Research Foundation, NASA Grant NSG 1098, Final Report, May 1977.

18. Orszag, S. A.: 1971a Numerical Simulation of Incompressible Flows Within Simple Boundaries: Accuracy, J. Fluid Mech. 49, 75.

19. Orszag, S. A.: 1971b Galerkin Approximations to Flows With Slabs, Spheres, and Cylinders, Phys. Rev. Letters 26, 1100 (1971).

20. Orszag, S. A.: 1971c Numerical Simulation of Incompressible Flows Within Simple Boundaries: Galerkin (Spectral) Representations, Stud. in Appl. Math. 50, 395.

21. Orszag, S. A.: 1976a Turbulence and Transition: A Progress Report, Proc. Fifth Int'l Conf. on Numerical Methods in Fluid Dynamics (ed. by A. I. van de Vooren and P. J. Zandbergen), Springer-Verlag, Berlin, p. 32.

22. Gottleib, David; and Orszag, Steven A.: Theory of Spectral Methods for Mixed Initial-Boundary Value Problems - Part I, ICASE Report No. 76-32, November 1976.

23. Gottleib, David; and Orszag, Steven A.: Theory of Spectral Methods for Mixed Initial-Boundary Value Problems - Part II, ICASE Report No. 77-11, July 1977.

24. Molodtsov, V. K.: The Numerical Calculation of the Supersonic Circulation of a Current of Viscous Perfect Gas Around a Sphere. USSR Comput. Math. and Math. Physics, Vol. 9, No. 5, 1969, pp. 320-329.

25. Molodtsov, V. K.; and Tolstykh, A. N.: Calculation of Supersonic Viscous Flow Around a Blunt Body. Proceedings 1st International Conf. Numerical Methods in Fluid Dynamics, Novossibirsk 1969, Vol. 1, pp. 37-54.

26. MacCormack, Robert W.: The Effect of Viscosity in Hypervelocity Impact Cratering. AIAA Paper No. 69-354, 1969.

27. MacCormack, R. W.: Numerical Solution of the Interaction of a Shock Wave With a Laminar Boundary Layer. Lecture Notes in Physics, Vol. 8, Springer-Verlag, 1971, pp. 151-163.

28. Peaceman, D. W.; and Rachford, H. H., Jr.: The Numerical Solution of Parabolic and Elliptic Differential Equations. SIAM Journal, Vol. 3, 1955, pp. 28-41.

29. MacCormack, Robert W.: An Efficient Numerical Method for Solving the Time-Dependent Compressible Navier-Stokes Equations at High Reynolds Number. NASA TM X-73,129, July 1976.

30. Shang, J. S.: An Implicit-Explicit Method for Solving the Navier-Stokes Equations. AIAA Paper No. 77-646, June 1977.

31. Blottner, F. G.: Computational Techniques for Boundary Layers. AGARD Lecture Series No. 73, Computational Methods for Inviscid and Viscous Two- and Three-Dimensional Flow Fields, 1975, pp. 3.1-3.51.

32. Beam, R. M.; and Warming, R. F.: An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form. Journal of Computational Physics, Vol. 22, 1976, pp. 87-110.

33. Beam, Richard M.; and Warming, R. F.: An Implicit Factored Scheme for the Compressible Navier-Stokes Equations. AIAA Paper No. 77-645, June 1977.

34. Steger, J. L.: Implicit Finite-Difference Simulation of Flow About Arbitrary Geometries With Application to Airfoils. AIAA Paper No. 77-665, June 1977.

35. Shang, J. S.; and Hankey, W. L.: Numerical Solution of the Compressible Navier-Stokes Equations for a Three-Dimensional Corner. AIAA Paper No. 77-169, January 1977.

36. MacCormack, R. W.; and Baldwin, B. S.: A Numerical Method for Solving the Navier-Stokes Equations With Application to Shock-Boundary Layer Interactions. AIAA Paper No. 75-1, January 1975.

37. Hung, C. M.; and MacCormack, R. W.: Numerical Solution of Supersonic Laminar Flow Over a Three-Dimensional Compression Corner. AIAA Paper No. 77-694, June 1977.

38. Hung, C. M.; and MacCormack, R. W.: Numerical Solution of Three-Dimensional Shock Wave and Turbulent Boundary Layer Interaction. Paper to be presented at the AIAA 16th Aerospace Sciences Meeting, Huntsville, Alabama, January 16-18, 1978.

39. Case, K. M.; Dyson, F. J.; Freeman, E. A.; Grosch, C. E.; and Perkins, F. W.: Numerical Simulation of Turbulence. Stanford Research Institute, Technical Report JSR-73-3, 1973.

40. Horstman, C. C.: A Turbulence Model for Nonequilibrium Adverse Pressure Gradient Flows. AIAA Paper No. 76-412, 1976.

41. East, Lionel F.: Computation of Three-Dimensional Turbulent Boundary Layers. Euromech 60, Trondheim 60, FFA TN AE-1211, September 1975.

42. Wilcox, D. C.: Numerical Study of Separated Turbulent Flows. AIAA Journal, Vol. 13, No. 5, pp. 555-556, 1975.

43. Wilcox, D. C.: Turbulence-Model Transition Predictions. AIAA Journal, Vol. 13, No. 2, pp. 241-243, 1975.

44. Wolfshtein, M.; Naot, D.; and Lin, A.: Models of Turbulence. Ben-Gurion University of the Negev. Dept. of Mechanical Engr., Report ME-746(N), June 1974.

45. Proceedings of a Symposium on Turbulent Shear Flows, Vol. 1, held at the Pennsylvania State University, University Park, Pennsylvania, April 18-20, 1977.

46. Hanjalic, K.; and Launder, B. E.: A Reynolds Stress Model of Turbulence and its Application to Thin Shear Flows. Journal of Fluid Mechanics, Vol. 52, part 4, pp. 609-638, 1972.

47. Launder, B. E.; Reece, G. J.; and Rodi, W.: Progress in the Development of a Reynolds-Stress Turbulence Closure. Journal of Fluid Mechanics, Vol. 68, part 3, pp. 537-566, 1975.

48. Orszag, S. A.: Minicomputers vs. Super Computers; A Study in Cost Effectiveness for Large Numerical Simulation Programs. Flow Research Note No. 38, 1973.

49. Gritton, E. C.; King, W. S.; Sutherland, I.; Gaines, R. S.; Gazley, G., Jr.; Grosch, C.; Juncosa, M.; and Petersen, H.: Feasibility of a Special Purpose Computer to Solve the Navier-Stokes Equations. Rand Report R-2183-RC, June 1977.

COMPUTING VISCOUS FLOWS

J. D. Murphy

Ames Research Center, NASA

Moffett Field, CA 94035

Due to the short time scale for the preparation of these remarks together with the restricted space available for presentation, I am taking the liberty of doing substantial violence to the usual NASA format for the presentation of technical information. Rather than the usual order of analysis, results, discussion, and conclusions this presentation will be simply a sequence of statements, each one followed by supporting material.

Statement 1

Computational aerodynamics is a discipline distinct from computational fluid dynamics in its goals and to a degree its techniques.

Computational fluid dynamics is, in general, the application of numerical analysis to the solution of the equations of fluid mechanics. As such it is primarily concerned with the mathematical structure of these equations and the generation of stable accurate algorithms for their solution.

Computational aerodynamics, on the other hand, is an engineering science, directed to the generation of useful information, applicable to the design of aircraft and aircraft components, predominantly through the application of numerical methods.

With these definitions it becomes clear that the major differences arise from the fact that computational aerodynamics is not concerned with what is "true," but rather what is "close enough" and what is "cheap enough."

Statement 2

To perform efficient aerodynamic computations the most attractive approach is the use of hybrid methods where the equations treated and the solution algorithms used reflect the local character of the flow.

It is becoming increasingly clear that, except for hypersonic flows with significant curvature, i.e., ref. 1, and for flows with large separation bubbles, e.g., ref. 2, boundary layer theory provides a perfectly adequate predictive capability for laminar flows at Reynolds numbers of importance to aerodynamicists. Figure 1, for example, shows a comparison of the skin-friction coefficient as obtained from boundary-layer theory, ref. 3, with that from a solution to the full Navier-Stokes equations, ref. 4, for laminar flow over a flat plate. Such differences as arise between the two solutions are

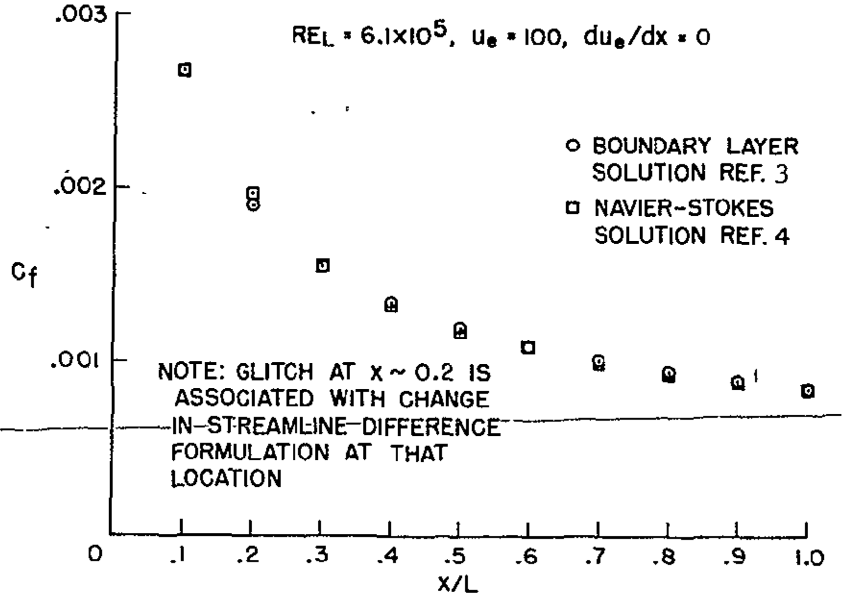


Fig. 1. Comparison of skin friction coefficients as obtained from boundary layer and Navier-Stokes calculations.

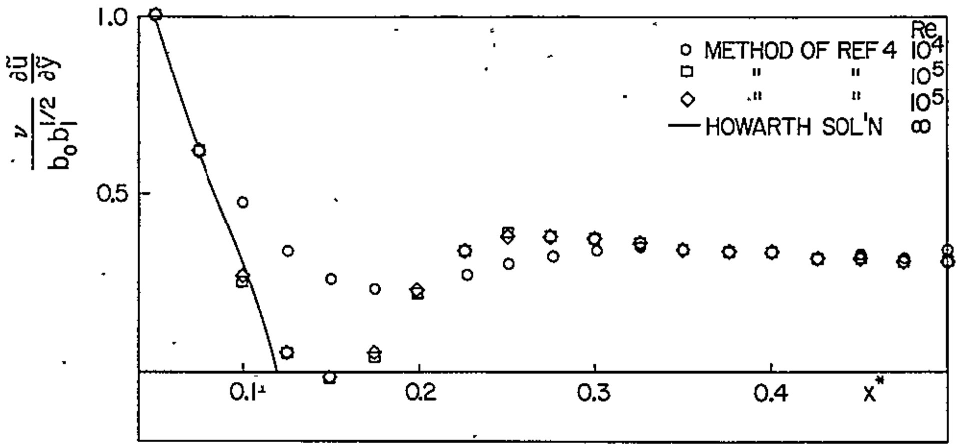


Fig. 2. Effect of Reynolds number on predicted nondimensional skin friction distribution.

almost totally numerical. Figure 2 conveys a similar message, although somewhat less directly. Here we see a comparison of three solutions to the Navier-Stokes equations, ref. 4, at increasing Reynolds number with the boundary-layer solution of Howarth, for a separating and reattaching flow. It is obvious that for the attached portion of the flow and for $RE_L \geq 10^5$, boundary-layer theory satisfies our criterion of "close enough." More importantly, however, we see that for high Reynolds numbers, the solution is independent of Reynolds number and hence it is the ellipticity of the Navier-Stokes system, and not the existence of normal pressure gradients which is significant. Further, this ellipticity can be artificially introduced into the boundary-layer equations to permit treatment of slender separation bubbles, e.g., refs. 5 - 8. Figure 3, taken from ref. 8, compares an inverse boundary-layer solution with the Navier-Stokes solution of MacCormack, ref. 9, for a Mach 2 laminar boundary-layer shock-wave interaction.

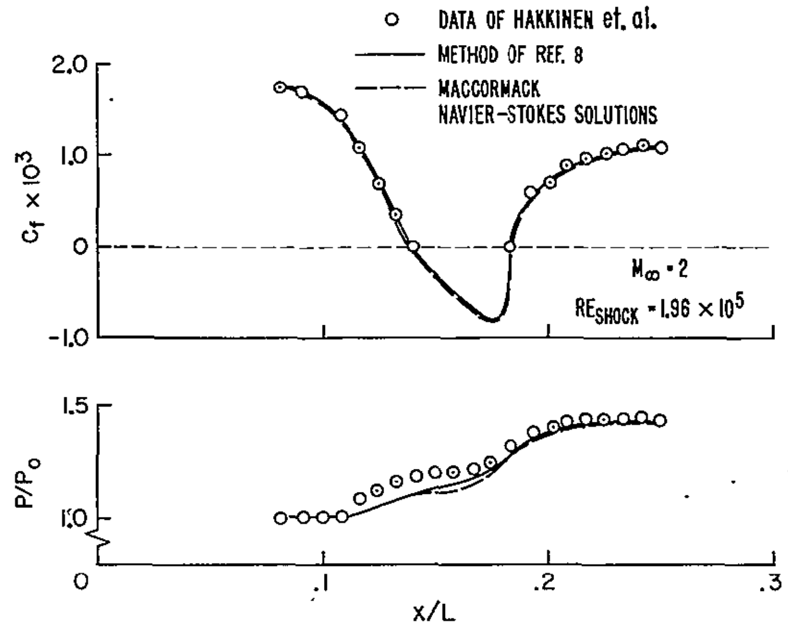


Fig. 3. Comparison of results of an inverse boundary-layer method with calculations of MacCormack.

This last figure is something of a "swindle" since in order to obtain the inverse boundary layer solution the skin-friction distribution must be input. The intent however, is to show that when the required ellipticity has been introduced, albeit artificially, the boundary layer equations represent the physics of quite a large variety of flows sufficiently to provide a "work-horse" calculation method for many computational aerodynamic needs. It is true that for some flow configurations, for example portions of military aircraft and off-design studies of commercial aircraft, solutions to the Navier-Stokes equations may be required. But even here it seems probable that hybrid calculation schemes offer the most promise for efficient computation. Examples of these kinds of methods using coupled (or patched) solutions of boundary layer, Navier-

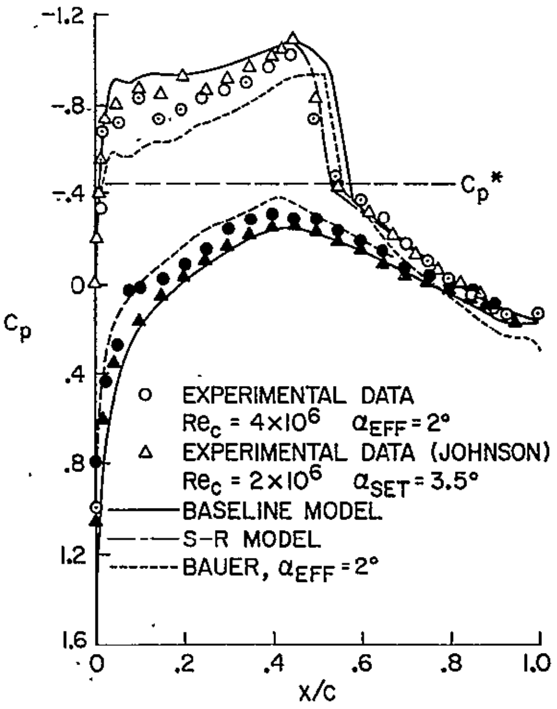


Fig. 4. Comparison of hybrid method results with experimental data; pressure distribution over a NACA 64A010 airfoil; M = 0.8, $Re_c = 2 \times 10^6$, $\alpha_{set} = 3.5°$.

Stokes and Euler equations are appearing with increasing frequency, e.g., refs. 10 - 13, and represent substantial economies in computation over the use of Navier-Stokes equations alone. Figure 4 (fig. 6 of ref. 12) shows a comparison of a hybrid method predicted and a measured pressure distribution on a NACA 64A010 at a Mach number of 0.8, $Re_c = 2 \times 10^6$ and $\alpha = 3.5$. The authors indicate an order of magnitude reduction in CPU time for the hybrid method as compared with a Navier-Stokes solution for the entire flow field.

Statement 3

The pacing item in obtaining a significant breakthrough in computational aerodynamics is a general turbulence model that works, and this breakthrough is only peripherally related to availability of large, fast computers.

Despite 100 years of study we have only a hazy qualitative idea of what is really going on in a turbulent flow. Fortunately, again our "close enough" criterion comes to the rescue. Figure 5 presents a comparison of the predicted skin friction distribution for turbulent flow over a flat plate with the data of Wieghardt, ref. 14. The turbulence model employed is a simple algebraic mixing length model embodying almost totally fictitious physics, but it works surprisingly well, not only for low speed flat plates, but for any flow for which the boundary conditions are not changing too rapidly. Even for more complicated flows such as an unseparated shock-wave boundary layer interaction, relatively minor modifi-

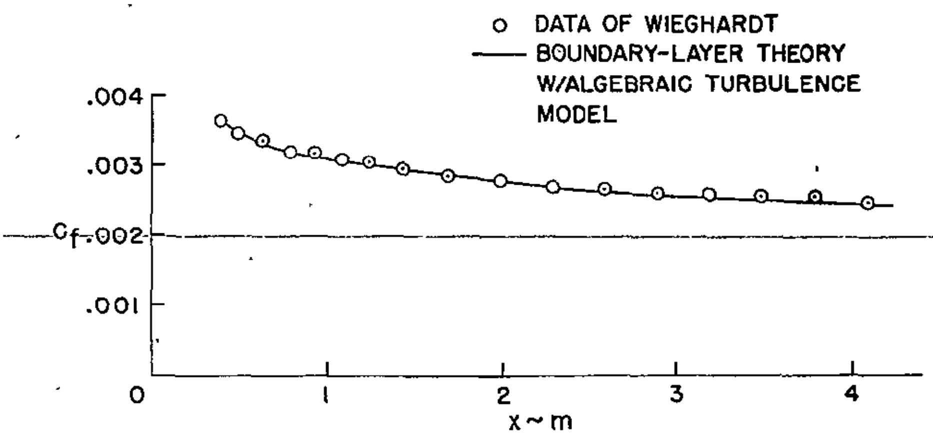


Fig. 5. Comparison of predicted skin friction distribution on a flat plate with the data of Wieghardt.

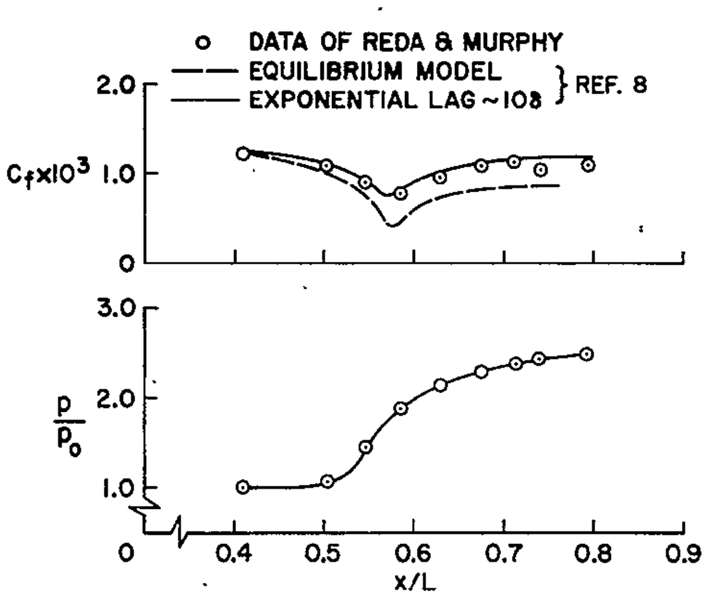


Fig. 6. Comparison of the present method with the data of reference 8; turbulent unseparated flow.

cations, such as an exponential lag governed by an ordinary differential equation, provide useful results, see fig. 6. For flows which are still more complicated, however, such as flows with large separation bubbles and three-dimensional and time-dependent flows, these models are not adequate and none of the proposed models have demonstrated significant generality.

To summarize this section one can do no better than to quote Peter Bradshaw. In ref. 15, he remarks that "It is not wise to distinguish—or choose—calculation methods on the basis of the numerical procedure employed, even though much of the work in developing a calculation method may be numerical analysis and computer programming: a numerical procedure without a turbulence model stands in the same relation to a complete calculation method as an ox does to a bull." Since the panel to follow is addressing itself exclusively to the subject of turbulence modeling there is no need to further belabor the point.

Statement 4

There is no unanimity of opinion as to what may be the optimum algorithm or even family of algorithms during the next decade.

The obvious direction for future efforts in both computational aerodynamics and fluid mechanics in general is toward the development of three-dimensional and time-dependent prediction methods. This is particularly true for the boundary layer equations which appear to lag inviscid methods in three-dimensions and Navier-Stokes methods in time-dependent flows, and are critical to the development of three-dimensional hybrid methods. At present I don't think we are capable of making a judgment as to which algorithms or even which family of algorithms may prove to be the most efficient for these classes of problems. Implicit methods including ADI, and various spline methods appear to offer significant promise for the future, but the ultimate determining parameter for useful calculations will remain the turbulence model. In fact a real possibility is that the most efficient numerical method will be determined by the character of the turbulence model.

Statement 5

It is premature to develop an optimum processor for computational aerodynamics, but such a machine, dedicated to the study of the structure of solutions to the three-dimensional time-dependent Navier-Stokes equations and to the computability of turbulence would be very valuable indeed.

It has been suggested that by optimizing the machine architecture about a specific computational algorithm one might pick up two or even three orders of magnitude in speed. This is very probably true, but even ignoring the very real problems associated with the design, fabrication, reliability, and software support for such a machine; we are not in a position today to determine what will prove to be the proper algorithm around which to optimize.

Since even in hybrid methods 80% of the time is spent on solving the Navier-Stokes equations it is clear that we should optimize about a Navier-Stokes solver, but over the past several years these solvers have been sped up by more than an order of magnitude so that we take the risk of producing (and paying for) a very powerful machine structured about an antique algorithm which is overall no more efficient than an off the shelf item at a fraction of the cost.

If, however, the decision is made to proceed with the procurement of such a machine, it would be only prudent to require that, in addition to the special purpose character of the machine, it be at least as fast in general computation as the best "off the shelf" computer at the time of delivery.

It strikes me that the real utility of a very large, very fast machine is in fundamental studies of the structure of solutions of the Navier-Stokes equations and in particular to investigations of the computability of turbulence. This has little to do with Computational Aerodynamics during the next ten years, but may well prove fundamental to our understanding of fluid mechanics in generations to follow.

Statement 6

From the foregoing it is clear that in order to make significant progress in computational aerodynamics we must continue to advance in both the physical and mathematical aspects of fluid mechanics. Here, as in all scientific endeavor, the primary motivation for advancement will be human curiosity; and the primary tools of advance will be human intelligence and creativity. If we lack these elements and an environment wherein they can prosper, arbitrarily large increases in computational power will be meaningless.

REFERENCES

1. Hung, C. M. and MacCormack, R. W.; Numerical Solutions of Supersonic and Hypersonic Laminar Compression Corner Flows. AIAA Journal Vol. 14, No. 4, April 1976, pp. 475 - 481.

2. Seetharam, H. O. and Wentz, W. J., Jr.; Experimental Investigation of Subsonic Turbulent Separated Boundary Layers on an Airfoil. Journal of Aircraft, Vol. 14, No. 1, January 1977, pp. 51 - 55.

3. Murphy, J. D. and Davis, C. B.; User's Guide—Ames Inlet Boundary Layer Program. NASA TM X-62,211, January 1973.

4. Murphy, J. D.; An Efficient Numerical Method for the Solution of the Incompressible Navier-Stokes Equations. AIAA Paper 77-171.

5. Murphy, J. D.; A Critical Evaluation of Analytic Methods for Predicting Laminar Boundary Layer Shock Wave Interaction, NASA TN D-7044, January 1971.

6. Klineberg, J. M. and Steger, J. L.; The Numerical Calculation of Laminar Boundary Layer Separation, NASA TN D-7732, July 1974.

7. Carter, J. E.; Solutions for Laminar Boundary Layers with Separation and Reattachment, AIAA Paper 74-584, 1974.

8. Murphy, J. D., Presley, L. L., and Rose, W. D.; On the Calculation of Supersonic Separating and Reattaching Flows, in AGARD CP 168 Flow Separation 1975.

9. MacCormack, R. W.; Numerical Solution of the Interaction of a Shock Wave with a Laminar Boundary Layer. Second International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, Vol. 8, Springer-Verlag, 1971.

10. Walitt, L. and King, L. S.; Computation of Viscous Transonic Flow About a Lifting Airfoil. AIAA Paper 77-679, 1977.

11. Seginer, A. and Rose, W. C.; A Numerical Solution of the Flow Field Over a Transonic Airfoil Including Strong Shock Induced Flow Separation, AIAA Paper 76-330, 1976.

12. Rose, W. C. and Seginer, A.; Calculation of Transonic Flow Over Supercritical Airfoil Sections. AIAA Paper 77-681.

13. Brune, G. W., Rubbert, P. E. and Forrester, C. K.; The Analysis of Flow Fields with Separation by Numerical Matching, in AGARD CP168 Flow Separation, 1975.

14. Wieghardt, K.; Proceedings, Computation of Turbulent Boundary Layers—1968 AFOSR IFP—Stanford Conference, Vol. 11, Compiled Data, Flow No. 1400, Dept. of Mech. Engineering, Stanford University, 1969.

15. Bradshaw, P.; The Understanding and Prediction of Turbulent Flow. Aeronautical Journal, July 1972.

PROSPECTS FOR COMPUTATIONAL AERODYNAMICS

J. C. Wu

Georgia Institute of Technology

Atlanta, Georgia 30332

During the past several years, my colleagues and I at Georgia Tech have been developing a new numerical approach, called the integral representations approach, for the solution of the Navier-Stokes equations. Our work is being supported by the Office of Naval Research, by the Army Research Office, and by the Georgia Institute of Technology under its academic research program. The theoretical basis of this approach as well as the detailed numerical procedures and computed results for various types of flow problems are presented in a series of articles prepared by my co-workers and myself (References 1 to 14). In some of our studies, the entire set of differential equations describing the fluid motion is recast into integral representations. The desired solutions are then obtained by numerical quadrature procedures. In other studies, only some of the differential equations are recast into integral representations. The formulation of the problem is then called the integro-differential formulation.

My remarks are based on our own experience in the development of the integral representation approach, our experience in applying available finite-difference and finite-element techniques, as well as our knowledge about the current work of many other researchers whom we keep in touch with continually.

Computational aerodynamicists participating in this workshop were asked to consider the following two questions:

1. What computational capability, in terms of arithmetic speed and memory size and access rate, is required for routinely solving three-dimensional aerodynamic problems including those with embedded separated turbulent flows?

2. What types of three-dimensional solution algorithms, turbulence models, and automatic grid generation methods are likely to be available by the early 1980's?

A year ago, I prepared an article (Reference 12) assessing the prospects for the routine numerical solution of two- and three-dimensional flow problems involving appreciable regions of separation at high Reynolds numbers. I find that the viewpoints expressed in that article are, for the most part, still current today.

In Reference 12, it was pointed out that for two-dimensional laminar flows the state of art permitted the development of a package of computer code that is efficient, reasonably universal, sufficiently accurate, and relatively simple to utilize. It was further suggested that such a package would have a relatively short life-span and would not see broad engineering usage more concerned with three-dimensional turbulent flows. Such a package nevertheless would be a highly valuable asset within the research community.

Recently, Dr. M. M. Wahbah, a member of our research team at Georgia Tech, prepared a general-purpose user-oriented package of computer code for internal steady laminar incompressible flows in two-dimensions using the integral representation approach (Reference 13). As input, a user assigns the locations and the sequence of the numerical data nodes to be used in the computation procedure, the velocity values at the boundary nodes, the Reynolds number of the specific problem, and several parameters such as a criterion for terminating the computation. The computer code then calculates, through the use of a computer, the numerical values of the velocity components and the vorticity at all data nodes as well as the pressure at all boundary nodes for the problem specified. Typically, CPU time for solving a problem at a Reynolds number of several thousand and using about a thousand data nodes is a few minutes on the CDC-6600 computer. This computer time requirement does not increase very rapidly with increasing Reynolds number.

Also recently, two of our Ph.D. students completed two separate studies of two-dimensional time-dependent laminar incompressible flows past airfoils. In one of these studies, S. Sampath considered an airfoil set into motion impulsively (Reference 1). In the other study, N. L. Sankar studied an airfoil oscillating in pitch at specified mean angles of attack, amplitudes, and frequencies of oscillation. (Reference 14). Both studies utilized the integro-differential formulation. In the impulsively started airfoil study, a transformation method is used to obtain a body-fitted grid system for the differential part of the solution procedure. (The integral representation part needs no special procedure for generating body-fitted grid systems). In the oscillating airfoil study, a hybrid finite difference-finite element grid system is used. Our experience indicates that it is now feasible to utilize the existing knowledge in computational fluid dynamics and construct a highly efficient general-purpose package of computer code for external laminar incompressible flows, either steady or time-dependent, in two-dimensions. For airfoil-type problems, such a package will require less than one minute of CDC-7600 CPU time to advance the solution by one dimensionless unit of physical time, i.e., the time interval during which the airfoil advances by one chord length relative to the freestream.

In contrast to the considerable experience that has been accumulated in recent years relating to laminar flow problems in two-dimensions, our own experience at Georgia Tech as well as those of our colleagues elsewhere are severely limited relating to three-dimensional solution algorithms and to turbulence models for separated flows. In our opinion, an accurate assessment of computer requirements for the routine solution of three-dimensional separated turbulent flow problems requires much more extensive experience in these two research areas than presently available.

Regarding three-dimensional solution algorithms, it is known that the extension of some of the more efficient numerical methods, which work well in two-dimensions, to three-dimensions presents some uncertainties. For example, in Reference 15, it is pointed out that plausible extensions of iterative ADI methods to three-dimensions frequently fail to converge. There appears to be little reason for doubting that, with extensive efforts devoted to the development of three-dimensional algorithms, some successful methods for treating three-dimensional separated laminar flows will be firmly established in the early 1980's. An uncertainty, however, does exist regarding the specific method that will eventually become the best candidate for a general-purpose three-dimensional code. In fact, judging from past experience, it is reasonable to expect that, during the next few years, some new numerical approaches will emerge and be demonstrated to be superior to the established approaches popularly considered today. The future development and general availability of more advanced and faster computers are important factors influencing the development of new methods. Conversely, planners of numerical flow simulation facilities should not overlook new numerical methods as they appear on the horizon.

At Georgia Tech, we conclusively demonstrated that, for the incompressible flow problem, the integral representation approach possesses the distinguishing ability of confining the solution field to the vortical regions of the flow. In an incompressible external flow, the inviscid portion of the flowfield, where the vorticity is negligible, is generally vastly larger in extent than the vortical region where viscous and Reynolds stresses are important. Because of the ability to confine the solution field to the vortical region, the integral representation approach requires drastically fewer numerical data nodes than other known methods which do not possess this ability. The advantages offered by this ability, in terms of computational requirements for two-dimensional problems, have been amply demonstrated. For three-dimensional problems, the factor of reduction of the number of data nodes tends to be the square of that in two-dimensions. Our estimate of the number of data nodes required for complex three-dimensional flow problems is about one tenth of that estimated by many other researchers. Therefore, we are convinced that the required arithmetic speed and central storage for the routine solution of three-dimensional laminar flow problems will be drastically smaller than those presently estimated by many other researchers.

At the present, our experience in treating three-dimensional problems using the integral representation approach is limited to flows involving very simple boundary geometries (Reference 7 and 10). For compressible flows, we have shown that the integral representation approach permits the solution field to be confined to the region where the vorticity and/or the dilatation is nonzero (Reference 4). We have yet to implement the approach for either the compressible flow or the three-dimensional flow involving complex geometries. Our estimate should be viewed, like those of our colleagues elsewhere, as educated guesses. There are a number of ways of increasing the solution efficiency. Some of these ways have been investigated reasonably thoroughly; others have merely been suggested. For example, a method of segmenting the solution field, which is already confined to the vortical region of the flow through the use of the integral representation approach, was demonstrated to offer substantial reduction in the amount of computation needed (References 1 and 11). It was shown that the segments can be of arbitrarily specified shapes and sizes, and each segment can contain any number of data nodes. The computation of field variable values within each segment can be performed independently of that in other compartments. This segmentation technique is therefore well-suited for parallel programming. Thus far, however, our own computations have all been carried out on older computers, such as the UNIVAC 1108 and the CDC-6400 and 6600, that do not possess a parallel programming capability. We have not yet demonstrated this well-suitedness by actually utilizing the parallel programming capability of a super computer such as the ILLIAC IV.

In our opinion, while drastic improvement in solution efficiency is no longer a critical factor in the routine computation of two-dimensional flows, it should be considered a pacing item for three-dimensional separated flows. We support the planning of a numerical aerodynamic simulation facility today. We wish to emphasize, however, that the development of more efficient algorithms will lessen the requirements on the facility. From a cost-effectiveness point of view, it will be important to stimulate worthy research in the area of three-dimensional algorithms while the flow simulation facility is being planned.

Our own experience in computing turbulent flows are at present limited to relatively simple two-dimensional problems, although we did explore the possibilities of using simple algebraic models, a two-equation model (Ref. 3) as well as a statistical distribution function approach (Ref. 16) on the basis of these simple problems. It appears that those of us who have devoted considerable amounts of efforts in computation of turbulent flows are in agreement that in the near future it will not be realistic to plan for a computing facility that permits routine numerical solution of the full Navier-Stokes equations for three-dimensional turbulent flows, including small-scaled motions, about complex solid geometries.

With Reynolds-averaged equations of motion, there is a great uncertainty regarding which, if any, of the presently proposed models of turbulence is sufficiently reliable or universal for the purpose of "routinely solving three-dimensional aerodynamic problems including those with embedded separated turbulent flows." The question as to which level of closure is adequate for the wide range of applications being considered has not been answered. Because of the empirical foundation of turbulence modelling, this question cannot be answered without extensive experimentation, both numerically and in the laboratory.

It is well known that turbulence research has been a most challenging activity in fluid mechanics for more than fifty years. Perhaps less well known is the fact that the concept of turbulent viscosity, which forms the basis of many of the algebraic and differential models of turbulence being studied today, was introduced by Boussinesq in 1877, precisely a century ago. The longevity, intensity, and ubiquity of interest in turbulent flow attested not only to its practical importance but also to the formidable difficulties attendant to the subject. For separated flows, the twin obstacles of (1) the lack of definitive experimental data of sufficiently high quality and fine detail and (2) the lack of tools powerful enough to accurately solve Reynolds averaged equations of motion, with any proposed model of turbulence, have in the past precluded the needed extensive numerical experimentation and calibration necessary for the firm establishment of turbulence models. It is natural for us to anticipate that the availability of modern instrumentation and computation facility will eventually remove these two obstacles. Bradshaw noted in the Sixth Reynolds-Prandtl lecture which he delivered in 1972 (Ref. 17) that we may hope for rapid progress in the future. His concluding paragraph of the lecture, quoted below, is of interest to us:

"What would our heroes say to all this, Reynolds who never saw hot-wire measurements of his turbulent stresses, Prandtl who never saw computer solutions of his turbulence models? Would they be amazed at the spectacular progress we have made? Perhaps they would be amused to find that with all our hot wires and computers we have still not achieved an engineering understanding of turbulence, and that it is still as important and fascinating and difficult a phenomenon as when the first steps in studying it were taken by Reynolds and Prandtl."

If we replace the words "hot-wire" and "computer" by "laser velocimeter" and "super computer", the above quotation is as worthy of note today as when Bradshaw delivered it five and a half years ago. There is no doubt that modern computing facilities and rapid response instrumentation have drastically expanded our horizon. We must point out, however, that the task involved in the establishment of suitable turbulence models is more enormous and longer-termed than some of us realize. Very few detailed and definitive measurements of a quality high enough to guide the development of turbulence models for separated flows exist today, even for "two-dimensional" flows. Chapman et al stated in 1975 (Ref. 18) that "...we strongly advocate that more carefully designed and thoroughly documented basic fluid dynamic experiments be conducted. These should cover a wide variety of flows of various degrees of complexity and encompass wide ranges of Mach and Reynolds numbers. More important, the documentation for each flow should include detailed measurements of such quantities as pressure distribution, skin friction, heat transfer, mean velocity and temperature profiles, and especially the fluctuating quantities which determine turbulent shear stress and energy transport. Few flows have been thoroughly documented to this requisite degree. But that documentation will be required in order to provide a basis for devising new and improved turbulence models..."

Chapman et al expressed optimism about more rapid development in turbulence modelling in the future (Ref. 18). While we share this optimism, we have in our minds a much longer time table than one presented by Chapman et al (Table 1 of Ref. 19). We feel that the magnitude of experimental efforts required is so immense that this task will not be completed before the mid 1980's. In fact, judging from the present pace, it appears to us it will be many years before adequate experimental information is accumulated and documented even for "two-dimensional" flows.

A computing facility designed specifically for aerodynamic simulation will be a highly valuable asset for computational aerodynamics. We support the early planning of such a facility. At the same time, we are of the opinion that many major obstacles, other than the absence of a bigger and faster computer, still exist. These obstacles require persistent long-term research activities to remove. Before they are removed, the aerodynamic simulation facility can only serve as a research tool and not a facility for the routine computation of complex three-dimensional separated turbulent flows.

The magnitude of the efforts required to develop turbulence models and three-dimensional algorithms indicates that computational fluid dynamic research needs to have a broad base. NASA can and should stimulate worthy research in computational fluid dynamics both within and outside its own research centers. Broader access to modern computing facilities that are in existence within NASA should be promoted for active researchers not affiliated directly with NASA. Funding for the development of turbulent models and of three-dimensional algorithms within and outside NASA should receive a higher priority than they are receiving at the present. A numerical wind tunnel with which we know neither the proper instrumentation nor how to install a test model is not an effective flow simulation facility. With additional emphasis on the numerical methods and the turbulence models, we can be reasonably certain that we will not end up with such a numerical wind tunnel.

REFERENCES

1. Sampath, S., "A Numerical Study of Incompressible Viscous Flow Around Airfoils," Ph.D. Thesis, Georgia Institute of Technology, 1977.

2. Wu, J. C. and Thompson, J. F., "Numerical Solutions of Time-Dependent Incompressible Navier-Stokes Equations Using an Integro-Differential Formulation," Vol. 1, No. 2, pp. 197-215, Journal of Computers and Fluids, 1973.

3. Wu, J. C., and Sugavanam, A., "A Method for the Numerical Solution of Turbulent Flow Problems," AIAA Paper No. 77-649, Proceedings of AIAA 3rd Computational Fluid Dynamics Conference, 1977.

4. Wu, J. C., "Integral Representations of Field Variables for the Finite Element Solution of Viscous Flow Problems," Proceedings of the 1974 Conference on Finite Element Methods in Engineering, Clarendon Press, 1974.

5. Wu, J. C. "Finite Element Solution of Flow Problems Using Integral Representation," Proceedings of Second International Symposium on Finite Element Methods in Flow Problems, International Centre for Computer Aided Design, Conference Series No. 2/76, June, 1976.

6. Wu, J. C., and Wahbah, M., "Numerical Solution of Viscous Flow Equations Using Integral Representations," Lecture Series in Physics, Springer-Verlag, Vol. 59, 1976.

7. Thompson, J. F., Shanks, S. P., and Wu, J. C., "Numerical Solution of Three-Dimensional Navier-Stokes Equations Showing Trailing Tip Vortices," AIAA Journal, Vol. 12, No. 6, pp. 787-794, June 1974.

8. Wu, J. C., "Numerical Boundary Conditions for Viscous Flow Problems," AIAA Journal, Vol. 14, No. 8, 1976.

9. Wu, J. C., and Sankar, N. L., "Explicit Finite Element Solution of the Viscous Flow Problem," Proceedings of the 1976 International Conference on Finite Element Methods in Engineering, 1976.

10. Wu, J. C., and Thompson, J. F., "Numerical Solution of Unsteady, Three-Dimensional Navier-Stokes Equations," Proceedings Project SQUID Workshop on Fluid Dynamics of Unsteady, Three-Dimensional, and Separated Flows, October, Purdue University, Lafayette, Indiana, October, 1971.

11. Wu, J. C., Spring, A. H. and Sankar, N. L., "A Flowfield Segmentation Method for the Numerical Solution of Viscous Flow Problems," Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, Vol. 35, Springer-Verlag, 1975.

12. Wu, J. C., "Prospects for the Numerical Solution of General Viscous Flow Problems," Proceedings of the Lockheed-Georgia Company "Viscous Flow Symposium," LG77ER0044, 1976.

13. Wahbah, M. M., "Computation of Internal Flows with Arbitrary Boundaries using the Integral Representation Method", Technical Report, School of Aerospace Engineering, Georgia Institute of Technology, October 1977.

14. Sankar, N. L., "Numerical Study of Laminar Unsteady Flow Over Airfoils," Ph.D. Thesis in preparation, Georgia Institute of Technology, October 1977.

15. Roache, P. J., "Computational Fluid Dynamics," Hermosa Publishers, 1972.

16. Scrinivasan, R., Giddens, D. P., Bangert, L. H., and Wu, J. C., "Turbulent Plane Couette Flow Using Probability Distribution Functions," The Physics of Fluids, Vol. 20, No. 4, April 1977.

17. Bradshaw, P. "The Understanding and Prediction of Turbulent Flow," Aeronautical Journal, July 1972.

18. Chapman, D., Mark, H., and Pirtle, M. W., "Reply to Bradshaw" Astronautics and Aeronautics, Vol. 13, No. 9, Sept. 1975, p. 6.

19. Chapman, D., Mark, H. and Pirtle, M. W., "Computers vs. Wind Tunnels," Astronautics and Aeronautics, Vol. 13, No. 4, April, 1975.

SESSION 6

Panel on TURBULENCE MODELING

Joel H. Ferziger, Chairman

LEVELS OF TURBULENCE 'PREDICTION'

by

Joel H. Ferziger and Stephen J. Kline
Department of Mechanical Engineering
Stanford University
Stanford, California

1. Introduction

Although the major purpose of this meeting is to look into the value of supercomputers in the 'prediction' of turbulent (and other) flows, it is well to begin by looking at the subject from a broader perspective. At the outset, a couple of important points need to be emphasized. The first is that, with the exception of a few very simple low Reynolds number turbulent flows, we can do almost nothing about predicting turbulent flows. (In this context, we are using prediction in the strong sense that the outcome of an experiment is calculated from nothing more than the fundamental equations of physics and the properties of matter.) In most cases, what we are really doing is what Saffman calls postdiction; i.e., we are having the computer use the results of a set of experiments to calculate the outcome of another experiment. Another way of looking at it is to say that we are performing interpolation, not extrapolation. In essence, many of our computer codes for turbulent flow computation are not much more than highly sophisticated versions of non-dimensional engineering correlation methods that have been in use for a long time. The second important point is that we may never be able to solve the Navier-Stokes equations for turbulent flows in the Reynolds number range of technological interest. Furthermore, there is no reason, other than aesthetic, why we should want to. In virtually every case, the information that is required is of a very low level compared to the complete details of a turbulent flow. All the engineer needs is certain simple data: for example, lift, drag and some important moments. The proper task for an engineer in design is to find a way to obtain this information with as little extraneous data and calculation as possible. In fact, we would argue that one of the principal aims of research in turbulent flow computation in the near term must be the establishment of a map that will tell the designer what level of description must be provided in a computation to produce a given level of results in terms of both accuracy and detail of information for each of various common types of problems.

There are a number of ways in which one can classify turbulent flow prediction methods. One is obviously in terms of the kind of flow: subsonic/transonic/supersonic, internal/external, free/bounded, and so forth. A second classification scheme, proposed by Bradshaw, is based on the complexity of the strains that the turbulence undergoes in the flow. This classification is particularly useful for 'modelers' constructing computation methods. However, our primary interest here is in knowing what type of program is necessary to compute the properties of a flow. For this purpose, a classification according to the level of detail of description the method provides is probably most useful. We emphasize, however, that all of the classification methods are tentative at the present time, and they are meant mainly to serve as the focus of much-needed further discussion.

We propose that flow calculations can be classified into five categories:

1. Correlations
2. Zonal methods
3. Time-averaged equations
4. Large-eddy simulation
5. Navier-Stokes solution

There are methods that fall into more than one category, and there are subdivisions of each category. This particular scheme seems to us to be the one that best sorts existing methods for the purpose of choice by an engineering user. The remainder of the paper is devoted to a discussion of the advantages and disadvantages of each of these five categories.

2. Correlations

It is well to remember that, even in this age of large computers and sophisticated numerical methods, the great bulk of engineering work involving fluids handling is still done via the use of relatively simple correlations. In situations in which the geometry is simple or where there are many devices with similar geometries, the most efficient and accurate approach to design is normally the use of empirical data in the form of non-dimensional correlations. Well-known examples of this approach are the friction factor charts for pipe flow and the rather extensive charts of non-dimensional heat transfer coefficients. More complex versions of the method are in use by almost all manufacturers, based on their own proprietary data..

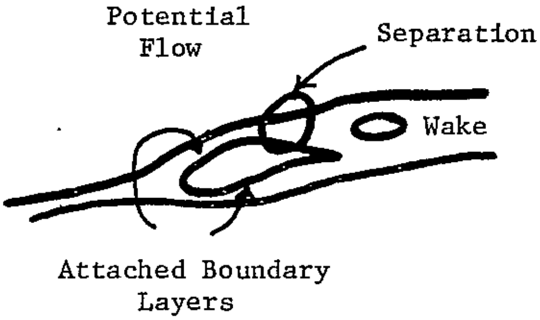
When the method is applicable, there is little question that it ought to be the preferred approach. The approach is simple, easily understood, very quick in application, and requires nothing more sophisticated than a set of charts and/or tables and a hand calculator. The difficulty with this method is that the data are available only for a set of standard cases, and any design that does not fall within the range covered by the data set requires new measurements; in aerodynamics this means a new wind tunnel test for almost every new shape. Also, because of the costs of data-gathering, correlations usually provide only a few kinds of simple information -- typically only average behavior for a few parameters. Thus the correlation approach is not one that is well adapted to the needs of an industry that relies on the continual introduction of new concepts or frequent and significant design changes from earlier practice.

3. Zonal Methods

A second category of flow 'prediction' is also quite old; it dates to the development of boundary layer theory in the early years of this century. In practice it also makes considerable use of empirical data in the form of correlations; however, the data are used in a more complex way that permits one to calculate the performance of devices for which direct experimental data are not available.

We shall define a 'zonal' method to be any approach in which the flow is divided into a number of 'flow modules', each of which is modeled by a different technique. Perhaps the simplest and best known example is Prandtl's original theory that divides a flow into a potential flow far from surfaces and a boundary layer in a thin region near the surface. The obvious advantage of such an approach is that the equations that one has to deal with in each region are simpler than the full Navier-Stokes equations. The difficulty in many cases is that of 'patching' the solutions together. In a typical calculation of the classical type, one first computes a potential flow about the body; then the pressure distribution at the surface, from the potential flow, is used to compute the boundary layer behavior. From the displacement thickness of the boundary layer a new potential flow is computed, and the process is iterated as required*.

The biggest drawback to this method from the point of view of the present-day designer is that it cannot adequately treat boundary layer separation. It is important to point out, however, that our understanding of the computation of flows near separation has improved considerably in the past several years, and it is now possible to compute at least some separated flows by modifications of Prandtl's original method. The number of flow modules used has to be greater than just the two in Prandtl's method. For example, the airfoil shown in the figure would require five zones: two attached boundary layers, a separation zone, a potential flow, and a wake.



Each flow module is computed using an appropriate approximate method. In most cases, it is advantageous to use the simplest method possible. (Our group has had some success with integral boundary layer methods combined with boundary integral methods for the potential flow.) Then some means must be found of patching the modules together, and this requires as much attention as the modules themselves. In particular, as Ghose and Kline [1] have pointed out, it is important to compute the potential flow and the boundary layer simultaneously in the region of separation.

It appears that, despite their relative crudity, zonal methods have the potential to be a useful design tool for some time to come. They offer the possibility of cheap computation (they require minutes on small machines, seconds on large ones) coupled with reasonable accuracy. They are thus well

* We omit here discussions of convergence and improved asymptotic matching, since it is a large topic and, although important in some cases, does not add much for the purpose of this discussion.

within the reach of the working engineer. Their most important shortcoming is that they usually must be redone for each important case, and the author of a program of this type needs to include all of the possibilities that might occur in the flow for which the program is designed. This is the price that must be paid for the simplicity of the equations in each region.

4. Time-Averaged Methods

We now come to an approach that is over a century old but, with a few important exceptions, saw little use until computers became widely available in the early 1960's. This method is based on averaging the Navier-Stokes equations and, largely for this reason, it has become a very popular approach. For flows which are steady in the mean, the averaging used is usually a long-term time average. Ensemble averaging is more appropriate for unsteady flows, while span averaging may be used in two-dimensional flows (Some of these terms require careful definition.)

No matter what averaging method is used, the major difficulty arises from the nonlinear term in the N-S equations. After the decomposition of the velocity field into a mean and a fluctuating part has been made, there always remains the Reynolds stress term $\rho\overline{u_i'u_j'}$. Although this term is typically small with respect to the other terms in the equation, its effects are usually profound on the parameters of design interest, and its accurate treatment is therefore often crucial. A number of methods of modeling this term have been tried. We will give only a very brief overview here; for further information, the reader is referred to the papers by Reynolds [2] and Rubesin [3].

The most popular approach to modeling the Reynolds stress is to make an analogy with the viscous stress and assume that it is proportional to the strain rate in the mean field $S_{ij} = (\partial\overline{u}_i/\partial x_j + \partial\overline{u}_j/\partial x_i)/2$. In the simplest models the proportionality parameter (eddy viscosity) is simply a prescribed function (either a constant or a function of the distance from a wall). Such models are called algebraic or zero-equation models. More complex models make the eddy viscosity a function of local properties of the turbulence, such as the kinetic energy or the length scale. New, auxiliary, partial-differential equations are required for the turbulence quantities used in these more complex models. These auxiliary equations are solved along with the equations describing the mean-flow field. We then have the so-called one- and two-equation turbulence models, depending on the number of additional quantities whose values are calculated. There is a great deal of effort on the development of models of this type at the present time.

The most sophisticated time-averaged models that are receiving attention at the present time are full Reynolds stress models in which partial differential equations are written for the Reynolds stresses themselves (three equations in 2-D, six equations in 3-D). These, too, are currently under intensive development.

The hope is that these more complex models will have a wider range of applicability than simpler models. To date, the evidence on this point is mixed; there is no clear proof either way. What seems to be reasonably clear is that, as a result of the flexibility of these models, they can probably be tuned to do an excellent job on a limited range of flows. It is the opinion of the authors that the most popular method for computing turbulent flows ten years from now will likely be two-equation models tuned for the particular type of flow; thus there will probably be several different models for different jobs.

Currently, the techniques are under intensive development in both modeling and algorithms. Using approximately 30 points in each dimension (a representative number), a program of this type typically requires on the order of 10 minutes on a machine of the CDC-6600 or IBM 370/168 size in two dimensions, and a few hours in three dimensions. This clearly means that programs of this type can be used only occasionally by designers at the present time, but one order of magnitude increase in available machine size will bring them to design feasibility. Experience with the methods is needed to determine their long-range value. Finally, there is a vital need for more experimental data of high quality that can be used to tune and test the models and algorithms. The need for data is likely to become more acute as time goes by. In a sense, computational methods are outrunning the data base from which they have historically been derived. In this connection, we emphasize two things. (i) At this level all methods known have been (and to date remain) postdictive, and thus require reliable data inputs covering a reasonable number of cases (in the 1968 Conference on computing turbulent boundary layers [4], this reasonable number was found to be at least a dozen).

(ii) Thus far, at least, the methods have not been found to extrapolate; whenever we have gone beyond the class of cases used to "tune" a method, we have found it necessary to introduce new data and modify or "retune" the model. This suggests that perhaps no single model, with a fixed set of constants, at this level of approximation, can "predict" all flows and therefore that we should seek a number of methods carefully classified regarding what problems they: (a) will do, (b) may do, and (c) won't do. We need to include estimates of uncertainty for types (a) and (b).

This suggests two further ideas. First, we need to be seriously sceptical of claims of universality -- of any single method purported to "predict" all turbulent flows at this level of approximation. Second, there is the possibility for using a combination of zonal ideas and more sophisticated models by using different closure models in different zones, e.g., in attached shear layers, near wakes, and so on within a given flow-field calculation.* This idea is not new but seems to the writers to be currently underexploited. It is not elegant, but may be very practical.

5. Large Eddy Simulation

This is a relatively new approach that has become feasible only since the introduction of the CDC-7600 and other machines of its size, speed, and cost per computation. The ideas behind the method are (i) the relatively well-established experimental result that the large eddies in any turbulent flow are dependent on the nature of the flow and vary greatly from flow to flow; (ii) the generally accepted hypothesis that the large eddies 'carry' most of the Reynolds stresses. The large eddies are difficult to model, and this is probably a central reason why turbulence modeling is difficult. On the other hand, the small eddies are nearly universal and isotropic and are not responsible for much of the overall transport of mass, momentum, and energy in a turbulent flow. (Most researchers believe the main effect of small eddies is to produce dissipation; however, some workers now believe that small eddies play an important role in creating new large eddies in turbulent boundary layers -- this area is also the focus of much current research.)

In large eddy simulation, one tries to compute the large eddies explicitly and model only the small eddies. This is accomplished by filtering or local averaging. These processes result in a set of equations for the large-eddy field which contains terms analogous to the Reynolds stresses of the models described earlier. They are

* In this light, the distinction between zonal methods and time-averaged methods begins to become unclear. It is possible to use time-averaged methods for some of the modules of a zonal method, e.g., the boundary layers, and it is possible to use different time-averaged methods in different zones.

called the sub-grid scale Reynolds stresses, and can be modeled by the methods mentioned in the previous section. To date, almost all calculations have been done with algebraic, i.e., zero equation, models.

The method has been applied only to relatively simple flows to date, but has shown itself to be extremely promising. Good results have been obtained in all cases tried to date; the evidence so far is that the simple sub-grid scale model used is adequate. Much more work needs to be done before this method can be applied to geometrically complex flows. Work on wall-bounded flows is only now beginning.

Large eddy simulation necessarily requires three-dimensional time-dependent calculation. Consequently, even a 16 × 16 × 16 mesh point calculation currently requires about 10 minutes on the 7600, and a 64 × 64 × 64 calculation (the largest yet attempted) requires a few hours. This means that large eddy simulation will remain a research tool even on next-generation computers. However, it mav become a verv valuable tool in providing information to be used in constructing and checking time-averaged methods.

Large eddy simulation provides a considerable amount of information about a turbulent flow. As a result, the output of a large eddy simulation program must be processed considerably before it can be useful. Typically the data are processed in a manner similar to that for experimental data; averages of various kinds are computed and computer graphics are used to provide 'flow visualizations'. If large eddy simulation is to be used to its full capacity in the future, considerable effort will be needed in developing three-dimensional computer graphics.

Finally, large eddy simulation can be used to check time-averaged models. From the output, one can compute the time-averaged Reynolds stresses and, simultaneously, the model approximations to them. One can then test the model directly by using correlation coefficients and, if the models are found valid, the constants in them can be evaluated. The remaining problem is that the contribution of the sub-grid scale turbulence to average quantities may be difficult to assess.

6. Navier-Stokes Equations

'Exact' solutions to the Navier-Stokes equations can be computed. Unfortunately, a well-known result due to Kolmogoroff shows that the number of mesh points required, scales like $Re^{9/4}$ in turbulent flows, where Re is the Reynolds number. Thus it is unlikely that there will ever be a computer with the capacity needed for calculating turbulent flows of engineering interest in complete detail, nor is it clear that one would want to do the calculation. The information that would be produced is not needed for most (perhaps all) engineering design work.

The role that exact simulations will play is likely to be in the area of model checking. Exact simulation does not suffer from the difficulty of estimating the effect of the sub-grid terms that arises in large-eddy simulation. It can therefore give unambiguous results as to the validity of a model. Furthermore, it can be used to check both the sub-grid scale models of large-eddy simulation and the Reynolds stress models of time-average calculations.

The major drawback in the exact solutions is a severe limit on the accessible range of Reynolds numbers, and one has to be cautious about extending results obtained outside the range of Reynolds numbers for which they are valid. Despite this, exact simulation is likely to be an important complement to experimental data in the area of model validation. Larger computers will, of course, extend the accessible range of Reynolds numbers.

7. Conclusions

1. A wide variety of methods for 'predicting' turbulent flows exists, and each method has an important contribution to make in its range of applicability.

2. The engineering designer should use the lowest-level method consistent with the accuracy desired. Higher-level methods can then be used to verify the results.

3. The development of computational methods will require ever-increasing amounts of experimental data. Since the lead time for experimental work is typically much larger than the lead time for computer program development, it is essential that the sponsorship of high-quality experimental work be made a high priority item and begun as soon as possible.

4. The computation of turbulent flows is an area that can fully occupy any computer that is likely to be built in the next 20 years. An increase in computer capacity of an order of magnitude yields only a twofold increase in the range of available Reynolds number for direct simulations but offers qualitative improvements at lower levels of computation. This increase is of considerable importance, however, and new computers can make a substantial contribution to the art and science of turbulent flow computation.

5. For technologies in which the use of correlations is not an open option, the computational methods in use ten years from now are likely to be found at what we have called levels two and three. Level two offers cheaper computation and allows the use of intuition to a greater degree than level three, but requires separate programming for every case. Level three allows the possibility of a single code that covers <u>some</u> variety of situations.

6. Given that in ten years the effective cost of computing will be considerably reduced from what it is now, we believe that the commonest design tools are likely to be two-dimensional computation at level three. Two equation models tuned to the particular type of flow are the most likely choice, but this is highly speculative. Zonal modeling will continue to be an important tool and should be used whenever a code applicable to the problem at hand is available. Three-dimensional zonal programs may be available at reasonable cost, but three-dimensional, two-equation programs will probably remain in the research and verification domain for this period.

References

1. Ghose, S., and Kline, S. J., "Prediction of Transitory Stall in Two-Dimensional Diffusers," Report MD-36, Dept. of Mech. Engrg., Stanford Univ., 1976.

2. Reynolds, W. C., "Computation of Turbulent Flows," Ann. Rev. Fluid Mech. 8, 193 (1976).

3. Rubesin, M., paper in this volume.

4. Kline, S. J., Morkovin, M. V., Sovran, G., and Cockrell, D. J., "Computation of Turbulent Boundary Layers - 1968," Thermosciences Div., Mech. Engrg. Dept. Stanford Univ., 1968.

MODELING OF THE REYNOLDS STRESSES

By

Morris W. Rubesin
Ames Research Center, NASA

It is generally accepted that for the next decade, or so, the computation of complex turbulent flow fields will be based on the Reynolds averaged conservation equations. In their most general form, these equations result from ensemble or time averages of the instantaneous Navier-Stokes equations or their compressible counterparts. For these averaging processes to be consistent, the averaging time period must exceed the periods identified with the largest time scales of the turbulence, and yet be shorter than the characteristic times of the flow field. With these equations long-period variations in the flow fields are deterministic, provided initial conditions are known. The averaged dependent variables are sufficiently smooth to be resolvable by finite difference techniques consistent with the size and speed of modern computers.

The difficulty with these equations is that they contain second-order moments of dependent variables as well as the first-order variables themselves. When equations for these moments are derived, these equations contain additional higher order moments. As the process is continued, the numbers of dependent variables grow at a faster rate than numbers of the equations. This proliferation of dependent variables and the need to truncate the process at a reasonable level is called the "closure problem." In first-order closure, these second-order moments, called the Reynolds stresses, are expressed algebraically as functions of the coordinates and the first-order dependent variables of the conservation equation, i.e., the mean fluid velocity and physical properties. Since these quantities are related algebraically, an equilibrium between turbulence stress and strain is implied. The process closes the problem at the level of the conservation equations. As no supplementary differential equations are introduced, first-order closure is sometimes called a zero-equation model. In second-order closure; third-order moments and moments other than Reynolds stresses are expressed algebraically in terms of the Reynolds stresses and the flow-field variables. The differential equations for the second-order moments are "closed" by this process. Currently, most of the modern modeling employs such second-order closure. The main differences between the methods are in the number of second-order equations employed. When a single turbulence kinetic energy equation is used to establish the intensity of the turbulence, it is called a one-equation model. In this case the length scales of the turbulence are defined algebraically in terms of the first-order variables. An eddy viscosity is defined that depends on the intensity and length scale. When both the scale and intensity are established with differential equations, the turbulence model is called a two-equation model. Finally, when the individual Reynolds stresses are expressed with differential equations, the models are called Reynolds stress models. For compressible flows, these latter models involve approximately 10 differential equations in addition to the conservation equations.

Examples of computations based on representative examples of these various classes of turbulence models are shown in the figures that follow. The boundary-layer experiments identified by the experimenters' names from Zwarts through Lewis et al. are described in Fig. 1. On Figures 2 through 5 the lines identified by: "Marvin-Sheaffer" represent a first-order, algebraic model, by "WT" a second order, two- equation model, and by "ARAP" a full Reynolds stress model. A comparison of the computed results and the data indicates that the more complex models are generally a little better at predicting the data than is the first-order, algebraic model. Although, the improvements of the newer models are not dramatic for these examples, the newer models also possess the decided advantage of being applicable, with minimum change, to flow fields other than attached boundary layers. The Reynolds stress model, which shows no significant advantage over the two-equation model in these examples, seems to possess this generality to a greater extent than does the two-equation model. These advantages, however, are not without cost. For similar marching techniques, the computer times required to solve a boundary-layer flow are roughly in the ratio of 1:2:5 for the algebraic, two-equation, and Reynolds stress models, respectively.

Examples of application of zero-, one-, and two-equation models to problems that must use the full Navier-Stokes (compressible) equations rather than boundary-layer equations are shown in Figures 6 and 7 for separated flow fields induced by a standing shock wave and a compression corner, respectively. The full Reynolds stress approach has not yet been tried in such a complex flow. Also, the two-equation results shown here are rather preliminary. For the two examples shown, the second-order closure models utilizing one and two equations, essentially unchanged from their attached boundary-layer forms, seem to capture the downstream skin friction rather significantly better than does the zero-equation model, though there is insufficient basis for choosing between the second-order closure models with the limited data shown. Upstream of separation, the zero-equation model is about as good as the two-equation model results, whereas the one-equation model lags the data. The relative costs of performing these calculations are indicated in the following table for the corner-flow problem.

TABLE I

Corner Flow Problem, 50x32 mesh

MODEL EMPLOYED	COMPUTER SPACE	TIME
0-Eq.	186K WORDS	2.7 SEC/ITER
1-Eq.	254K WORDS	4.1 SEC/ITER
2-Eq.	208K WORDS	6.7 SEC/ITER

It can be concluded from this brief examination of turbulence modeling that for two-dimensional attached boundary layers the newer second-order closure models on the whole, provide somewhat better agreement with data but at higher computer costs. For two-dimensional separated flows, computations with time-dependent solutions of averaged Navier-Stokes equations show serious shortcomings in skin-friction predictions by the 0-eq. model and potential with the 1-eq. and 2-eq. models. For the newer models, the computation costs, at least up to two-equation models, are at acceptable levels.

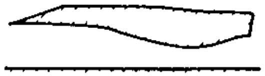
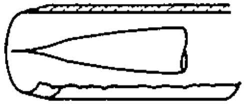
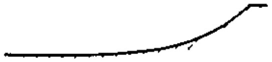
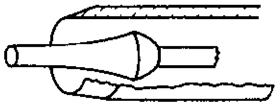
REF.	EXPERIMENTAL CONFIGURATION	M_∞	$Re_{\theta_\infty} \times 10^{-4}$	T_w/T_o	P^+_{max}
ZWARTS		4.02	3.5	1	0.004
PEAKE, BRAKMANN AND ROMESKIE		3.93	1.1	1	0.006
STUREK AND DANBERG		3.54	2.0–2.8	1	0.0085– 0.0125
LEWIS, GRAN AND KUBOTA		3.98	0.5	1	0.011

Figure 1. Experiments Used As Standards

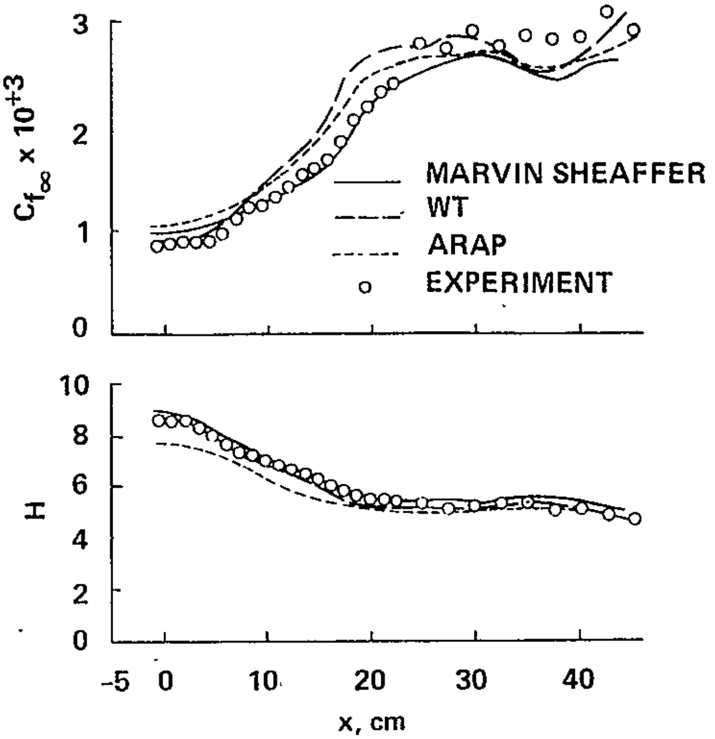


Figure 2. Comparison of Computations with Data of Zwarts

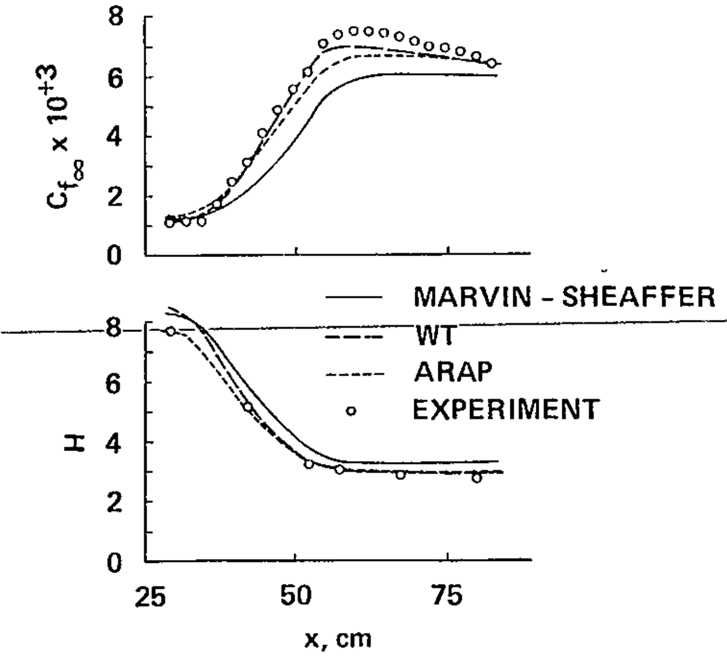


Figure 3. Comparison of Computations with Data of Peake et al.

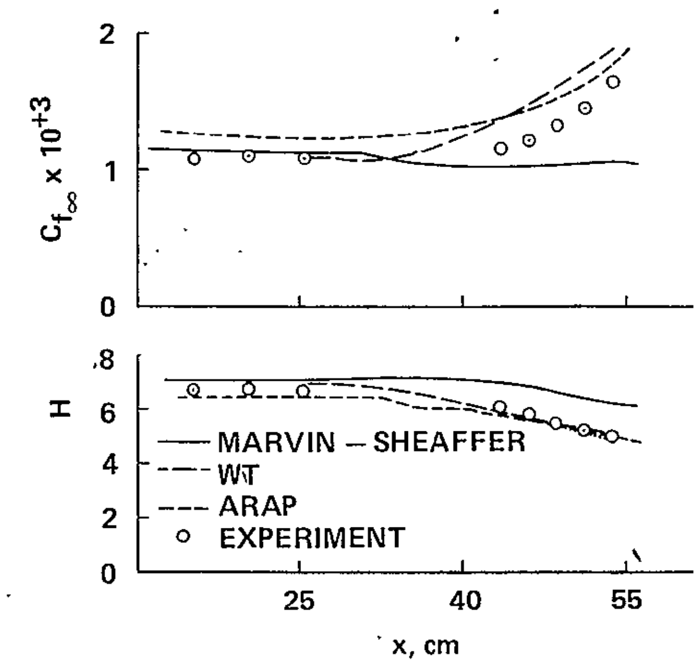


Figure 4. Comparison of Computations with Data of Sturek and Danberg

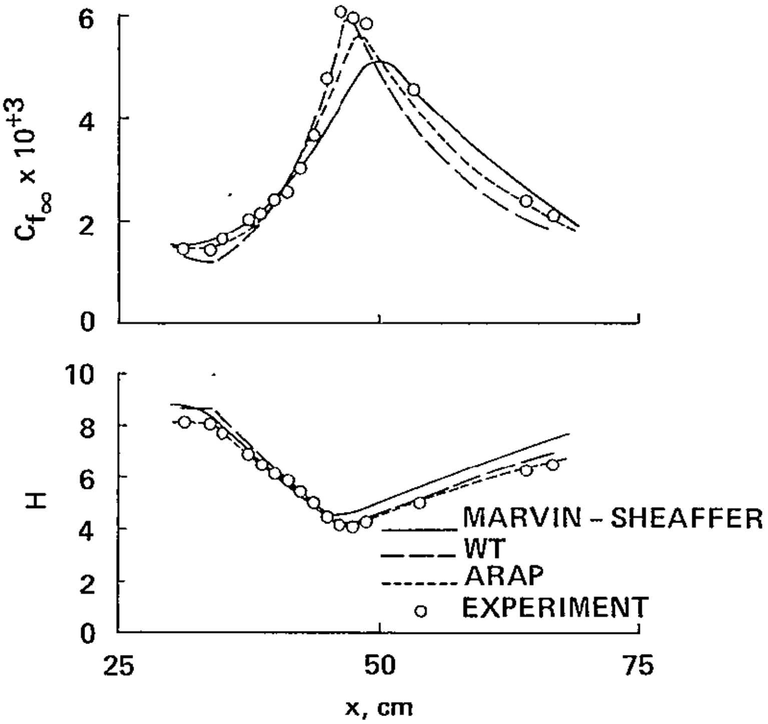


Figure 5. Comparison of Computations with Data of Lewis et al.

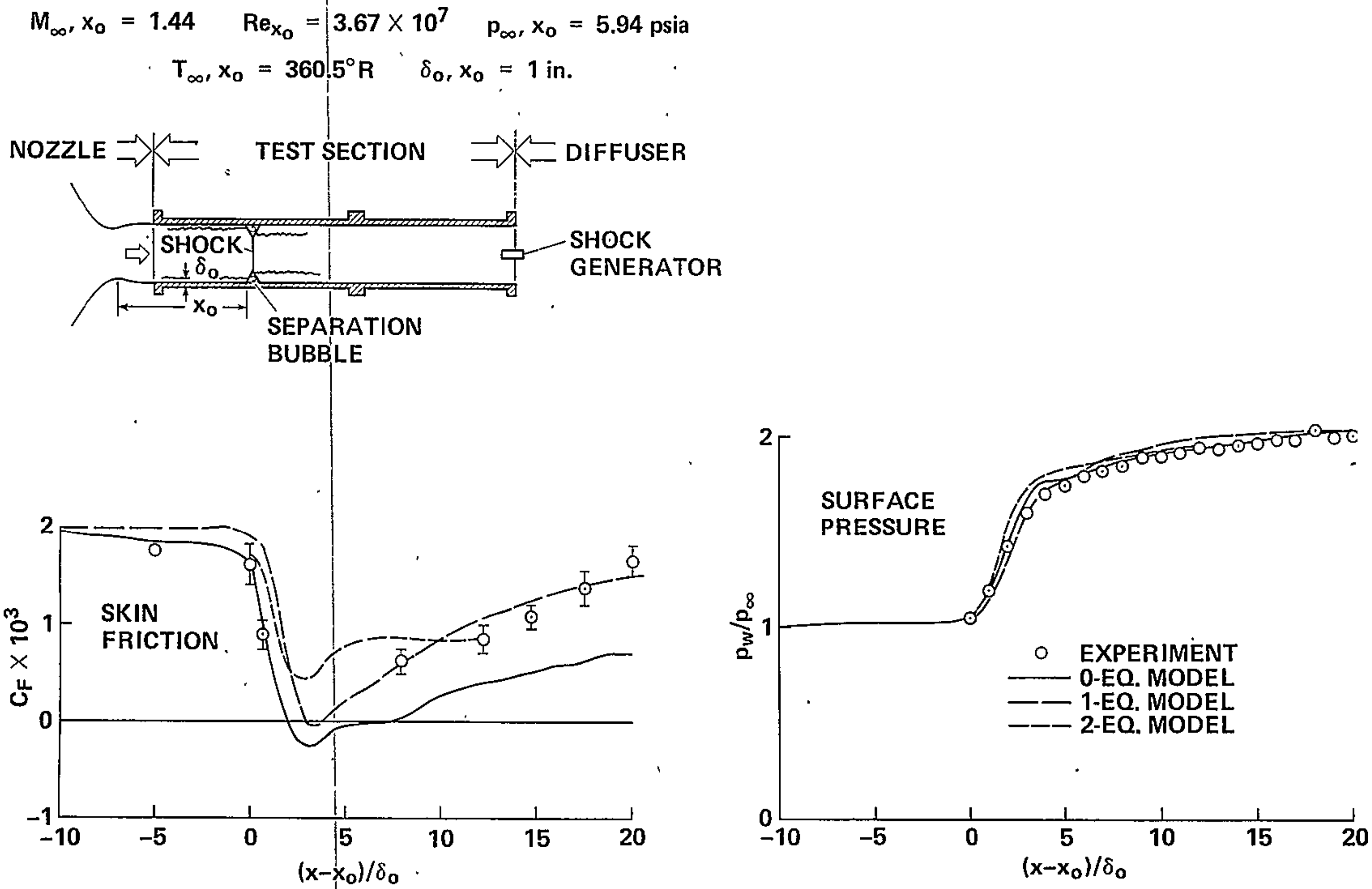


Figure 6. Transonic-Normal Shock-Wave-Induced Separation Experiment

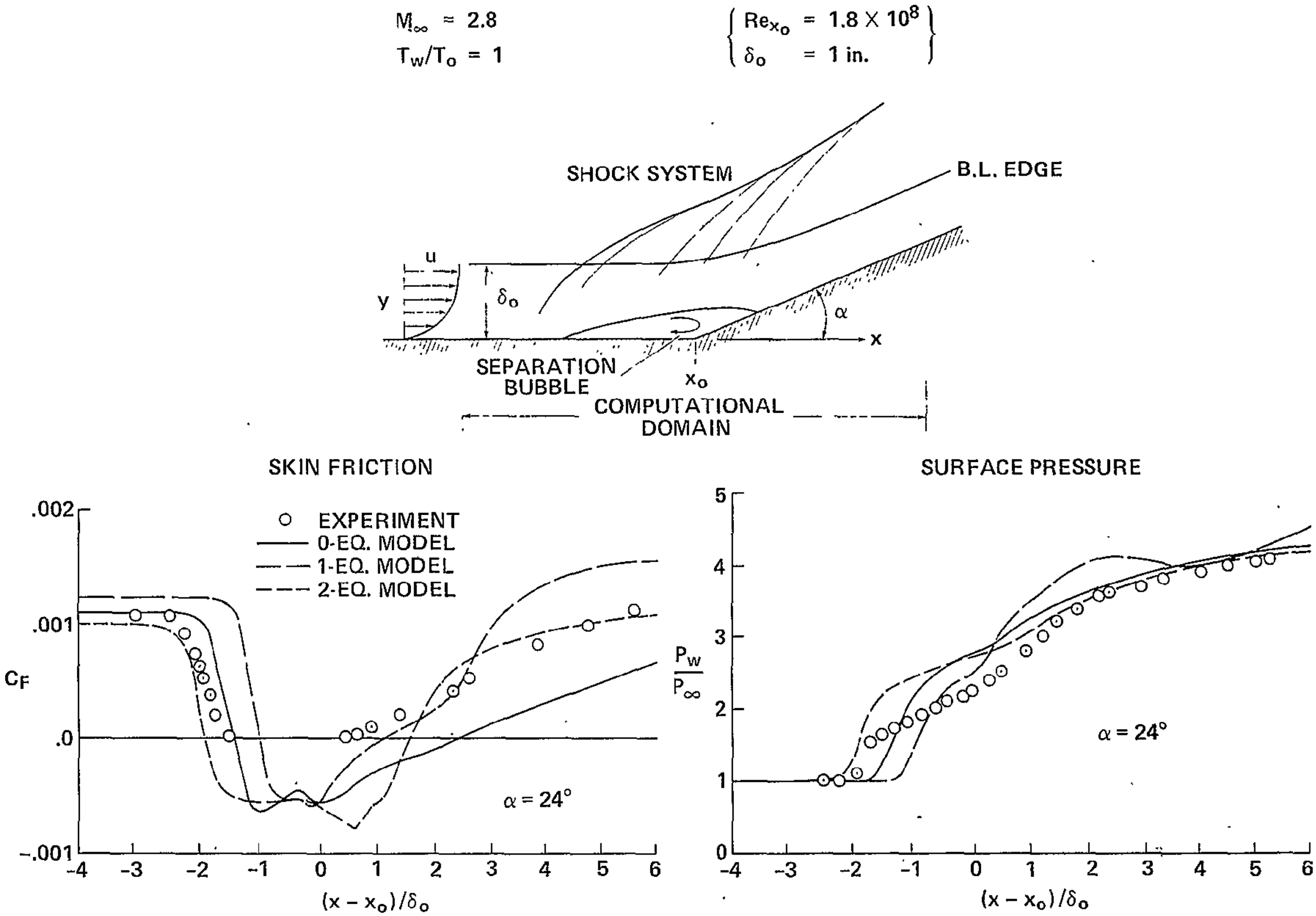


Figure 7. Supersonic-Compression Corner-Shock-Wave-Induced Separation Experiment

TURBULENCE MODELS FROM THE POINT OF VIEW OF AN INDUSTRIAL USER

S. F. BIRCH

BOEING MILITARY AIRPLANE DEVELOPMENT
SEATTLE, WASHINGTON 98124

INTRODUCTION

From the point of view of the potential user of numerical fluid mechanics, the overall objective is the development of useful design tools. In the aircraft industry, this means methods capable of handling fully three-dimensional mixed subsonic and supersonic flows.

Since there appears to be little prospect of the development of methods for the solution of the full, time-dependent Navier-Stokes equations in the near future, we will continue to need turbulence models to approximate the Reynolds stress terms that appear in the time-averaged Navier-Stokes equations. It is important to emphasize, however, that even if methods were available for solving the full equations, this would not necessarily be the optimum choice in all cases. As the cost of numerical computations decreases, the trend toward the use of more complex methods is likely to continue, but there will always be a need for a range of methods, depending on the accuracy and detail required from the calculation.

It is also important to appreciate that if useful design tools are to become available in a timely manner, it will require the coordinated efforts of specialists in a variety of research areas, and turbulence modeling is only one of the areas. The emphasis here is on the word "coordinated." Specifically, this means that not only must the turbulence model be valid for the flows considered, it must also be compatible with the solution algorithm being used, and with the storage capacity of the available computers.

Since much of the expected increase in computer speed and storage capacity over, say, the next 10 years is probably going to be used primarily in the solution of more geometrically complex problems, interest in relatively simple turbulence is likely to continue. It is probably inevitable that increased generality will require increased complexity but, at least for the industrial user, simplicity will probably continue to be a desirable goal.

PROGRESS AND PROBLEMS

One of the most obvious conclusions one reaches in reviewing progress of our understanding of turbulent flow over the last 10 years or so is that improved understanding is not achieved either easily or quickly. Much of the recent improvement in our prediction ability has been due more to the availability of large computers, which has allowed us to implement ideas proposed earlier, than to any breakthrough in our understanding of turbulence itself. Virtually all of the turbulence models now in use are based on work started in the mid-forties or early fifties. Certainly, there have been some recent improvements and refinements, but the major advance has been in our ability to solve sets of coupled, nonlinear, partial differential equations.

In an excellent review paper on turbulent shear flows published in 1966(1), Kline identified many of the important problem areas in both free shear flows and in wall boundary layers. It is discouraging to find that most of the problem areas identified by Kline are still with us. Take for example the near field or developing region of free shear flows. As Kline points out, this region of free shear flows is important for at least two reasons. First, it is important in itself, since in many industrial applications most or all of the events of interest take place within the developing region. Secondly, it is important even if we are primarily interested in the far field or the fully developed region of the flow. Say we wish to predict the velocity decay in the far field of a simple axisymmetric jet. There are a number of turbulence models available that will accurately predict the mixing rate in the far field of an axisymmetric jet, but since we must start our calculation at the nozzle exit, the overall accuracy of our prediction in the far field will be limited by our inability to accurately predict the mixing rate in the initial developing region of the jet. In spite of some improvement in our understanding of the near field, our ability to predict it has remained substantially unchanged over the last 10 years.

This is due, at least in part, to the lack of detailed experimental data, and this brings us to a second major problem. Our ability to predict turbulent flows is at present increasing much faster than we are acquiring the experimental data necessary to evaluate the predictions. This problem is particularly acute for complex three-dimensional flows, especially at full scale. More and more today we are finding that our numerical prediction capability cannot be fully utilized because we do not have sufficient experimental data to establish the reliability of the predictions. This is already a serious problem and may well become chronic in the near future.

In spite of the above problems, numerical methods have had a significant impact on the design process over the last 10 years. Finite difference solutions for two-dimensional wall boundary layers are now almost standard procedure in the aircraft industry. Transition and separation are still problem areas, but the overall reliability of the predictions is generally good. This was dramatically illustrated recently when Boeing selected an inlet design for the 727-300 aircraft without any experimental tests. Had development of the airplane continued, the inlet would undoubtedly have been tested before the airplane went into production. Nevertheless, this does illustrate the extent to which numerical methods have replaced parametric experimental testing.

Unfortunately, many flows of practical importance are inherently three-dimensional, and the ability to predict such flows has become possible only recently. Some examples of the type of three-dimensonal viscous flows that are now being analyzed are shown in Figures 1 and 2. The first is an experimental and numerical study of the flow downstream of a 12-lobe mixer, inside the tailpipe of a turbofan engine. The calculations were started at the mixer exit plane and were continued downstream to the nozzle exit. A comparison between numerical predictions and experimental data, for a model-scale simulation of the full-scale flow, is shown in Figure 1, together with the full-scale data. In view of the fact that these predictions were run "blind," without detailed experimental data at the starting plane, the agreement between the predicted and measured data is very encouraging. This work is described in more detail in reference 2.

The work illustrated in Figure 2 was undertaken because of discrepancies between numerical predictions and experimental data. Initial attempts to predict the flow within the tailpipe of the same engine, with the mixer removed, were not in good agreement with the available experimental data. Since the flow was nominally axisymmetric, only one or two data traverses had been taken at each axial station. However, the discrepancies between the predicted and measured data were larger than could be explained based on the approximations involved in the analysis, and this led to a more detailed experimental study of the flows. Apparently, the flow leaving the turbine retained sufficient swirl to set up recirculation cells in the cross plane, when it interacted with engine struts located downstream of the turbine exit. This led to a strongly three-dimensional flow within the engine tailpipe. Using experimental mean velocity profiles, measured at a station about one foot downstream of the turbine exit, the numerical calculations were repeated, and these are the predictions shown in Figure 2 -- clearly a big improvement. Although the types of three-dimensional flows that can be analyzed at present are still somewhat limited, and the results are not always highly accurate, the reliability of the predictions, at least for some selected flows, does appear to be good enough for the results to be useful as an aid in the design process.

Although any assessment of progress in the development of turbulence models will reflect, to some extent, the author's interests and personal opinions, there are, I believe, two developments over the last 10 years that deserve special mention. One is the development of model equations for turbulence length scales, or for length scale containing quantities. The second is the proposal by Bradshaw(3,4) for a classification system for complex turbulent flows.

When the Navier-Stokes equations are time-averaged to give the Reynolds equations, information is lost. A consequence of this is that we are left with an open set of equations in which there are always more unknowns than there are equations. This is the familiar turbulence closure problem. The equations for the mean velocity components contain second-order correlations known as the Reynolds stresses. Equations can be derived for these correlations, but they will be found to contain additional correlations, and so on. The objective of developing a turbulence model is to try to replace the information lost in the averaging process, and so to close the set of equations. Now since most of the information lost in the time-averaging process is phase information, information about the turbulence length scales, it should be no surprise to find that the range of application of a turbulence model is critically dependent on how the turbulence length scales are specified. If one is interested only in a limited range of flow, then a simple means of specifying the length scale is often adequate. For example, Prandtl's mixing length formula will give good results for many wall boundary layer flows. But if one requires a turbulence model valid for a wide range of flows, then a length scale equation, or its equivalent, is required.

The development of model equations for turbulence length scales, however, presents formidable problems. Exact equations for length scale containing quantities can be derived, but because of their complexity these equations are only of limited use in the development of model equations. In spite of the problems involved, a number of such equations have been developed and some have been tested for a fairly wide range of flows. None of these turbulence models are valid for all flows, but the best of them do give predictions that are accurate enough for many engineering applications, for a surprisingly wide range of flows.

The need for a classification system for turbulent flows, and in particular its relation to turbulence models, is perhaps less obvious. It is generally agreed that current turbulence models cannot be reliably used to predict flows that differ from those used to validate the model. But how different is different? The variety of flows at present amenable to numerical analysis is so large that the specific flow of interest to the potential user of a calculation method will almost certainly differ in some way from the flows that have been used to validate the model. After all, if experimental data were available for the flow of interest, there would be no need to predict it. The important question is, are the differences significant? It is not possible to answer this question without some implicit or explicit classification of turbulent flows. A classification system of some sort is also implicit in any discussion of experimental data, where the results of one experiment are compared and contrasted with the results from other experiments.

Turbulent flows have traditionally been classified based on flow geometry, as for example, jets, wakes, or wall boundary layers. If one is concerned primarily with the simple classical flows, then this system may appear to be entirely adequate. But for the complex three-dimensional flows one encounters in most practical applications, a classification scheme based on flow geometry is almost useless. To give just one example, in two dimensions a jet may be either planar or axisymmetric, or perhaps radial. In three dimensions, the variations possible are almost endless; in the aircraft industry, for noise applications alone, thousands of different nozzles have been tested over the last 20 years. To regard each flow as a class by itself is obviously impractical, yet the differences from flow to flow may be significant.

Bradshaw's proposal to classify complex turbulent flows by flow phenomena rather than by flow geometry has a number of advantages. The most obvious of these is that it greatly reduces the number of flow classes. Secondly, a classification system based on flow phenomena appears to be more useful, at least in the context of turbulence models, since the models themselves are basically phenomenological.

TURBULENCE MODELS IN THE EIGHTIES

What changes do we expect to see in turbulence models over the next 10 or 15 years? First, I think we must accept that there is not likely to be a major breakthrough that will revolutionize turbulence modeling. It could happen, but we should not count on it. As larger computers become available, we will see more work on subgrid scale models and attempts to obtain solutions to the full time-dependent Navier-Stokes equations for some selected low Reynolds number flows. I would expect to see this work starting to have some impact on the development of turbulence models, but these methods will probably not be used directly for the solution of practical problems. The turbulence models used in practical calculations will not differ greatly from the models now in use. They will be more general and probably more complex, but still recognizable extensions of models now in use. However, given sufficient computer resources, relatively modest improvements in turbulence models will allow us to compute many flows of practical importance. Ten years from now, I would expect to see three-dimensional viscous flow predictions in general use, at least at the preliminary design stage, and perhaps for some detailed design problems where the validity of the models has been demonstrated.

The turbulence models in use at present use a single turbulence length scale. This implies a universal turbulence energy spectrum, and this can obviously only be true for a very limited range of flows. For many flows, such turbulence models may predict results of acceptable accuracy. There are, however, many situations where this assumption is not only clearly invalid, but where it appears to lead to predictions that are not even qualitatively in agreement with experimental measurements. Transition and laminarization are obvious examples of flow situations where the shape of the turbulence energy spectrum changes dramatically. There are, however, many other flow situations where similar but perhaps less dramatic effects must be expected. Strong additional rates of strain, or sudden changes in the boundary conditions on a shear layer, for example, near a separation or reattachment point, may also lead to significant changes in the shape of the turbulence energy spectrum. To account for these changes, we will probably need additional length scale equations. A number of groups are already working on such models, and hopefully they will be available for use by the mid-eighties.

REFERENCES

(1) Kline, S. J., "Some Remarks on Turbulent Shear Flows," Proc. Intn. Mech. Engrs., Vol. 180, Pt3F, 1965-1966.

(2) Birch, S. F., Paynter, G. C., Spalding, D. B., and Tatchell, D. G., "An Experimental and Numerical Study of the 3-D Mixing Flows of a Turbofan Engine Exhaust System" AIAA 15th Aerospace Sciences Meeting, Los Angeles, 1977 Paper No. 77-204.

(3) Bradshaw, P., "Variation on a Theme of Prandtl," AGARD Conference Proceedings No. 93, Turbulent Shear Flows, 1971.

(4) Bradshaw, P., "Complex Turbulent Flows," Trans. ASME, J. Fluids Eng., 97,146, 1975.

MEASURED AND PREDICTED RADIAL DISTRIBUTIONS OF TOTAL TEMPERATURE AT NOZZLE EXIT IN PLANE ALIGNED WITH PRIMARY LOBE

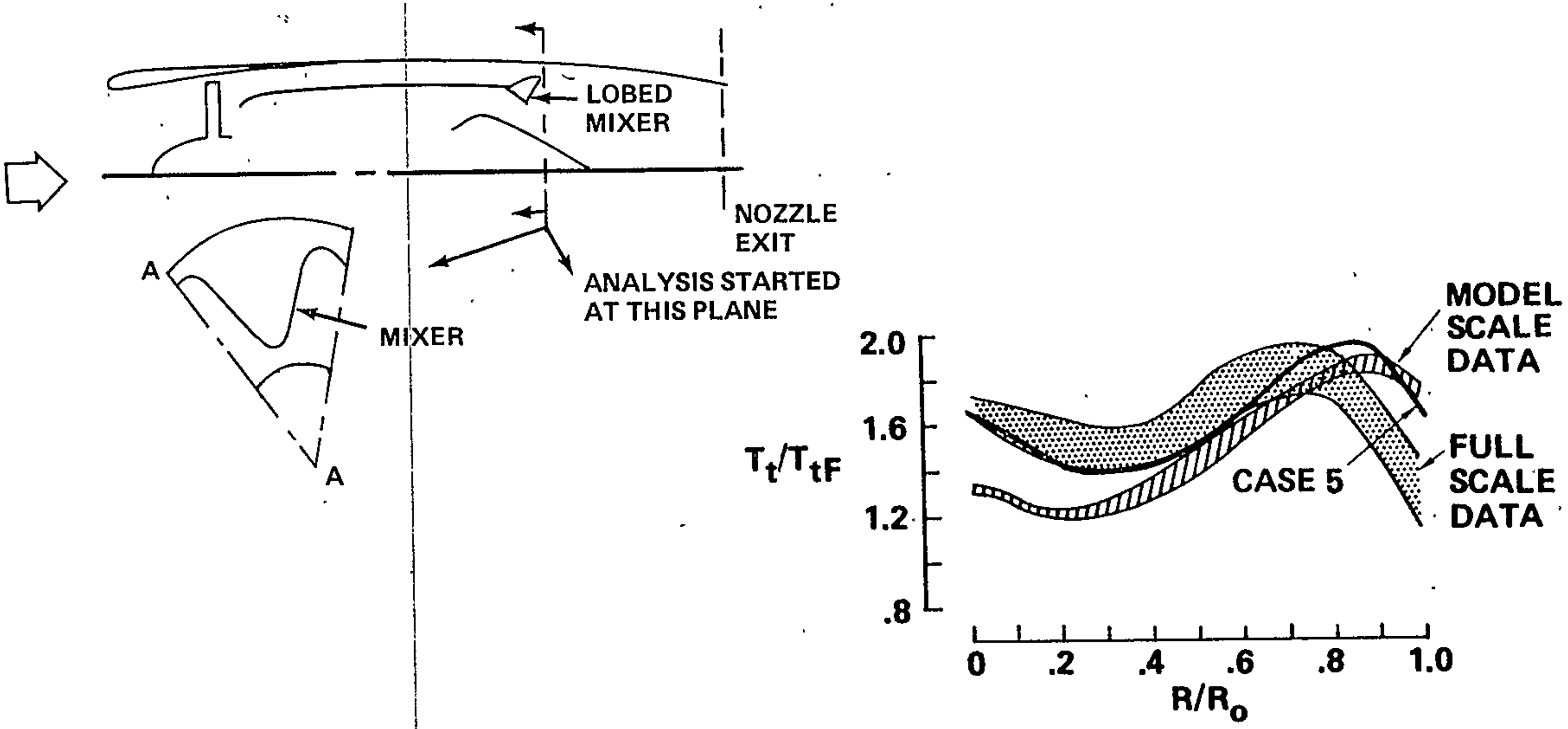
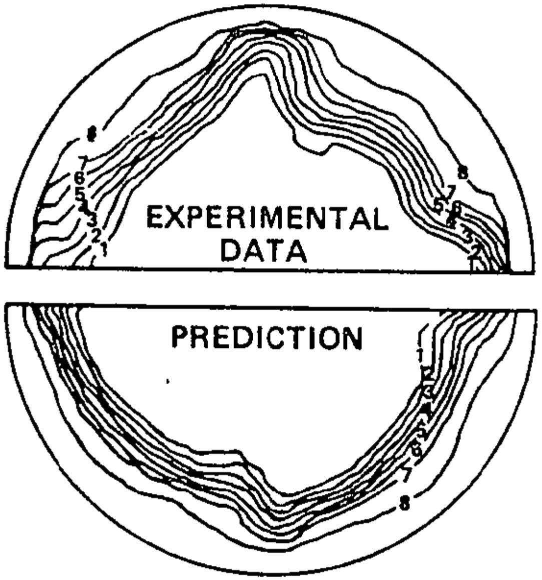


Figure 1. Example — 3-D Analysis for Lobed Mixers

PREDICTED AND MEASURED VELOCITY CONTOURS AT THE EXIT PLANE OF A JT8D-17 ENGINE



		V/VIP
1	—	.924
2	—	.893
3	—	.862
4	—	.8316
5	—	.800
6	—	.770
7	—	.740
8	—	.680

CONCLUSION:

INTERACTION BETWEEN ENGINE SWIRL AND TURBINE SUPPORT STRUT SETS UP AN ASYMMETRIC NOZZLE EXIT FLOW

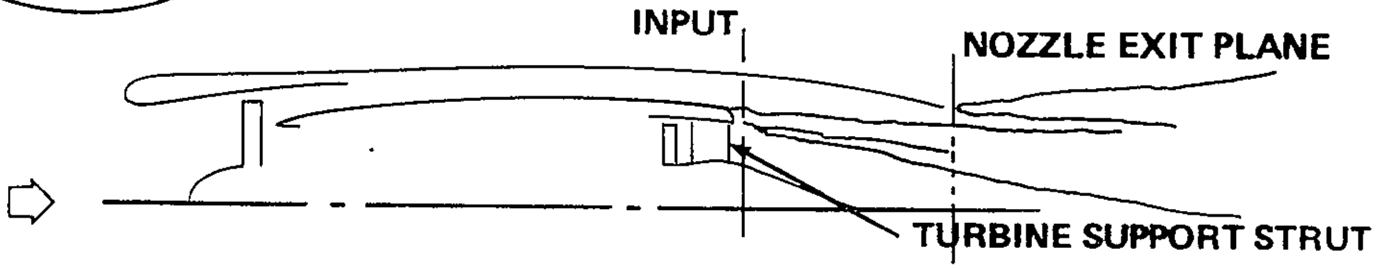


Figure 2. Example of 3-D Mixing Analysis for Confluent Fan Engine Nozzle Flows

D21

N78-19799

A DUAL ASSAULT UPON TURBULENCE

F. R. Payne
University of Texas at Arlington

I. Introduction

The fundamental problem of turbulence modelling (Rubesin, 1975) is the wide range of length and time scales of motions contributing to the turbulence "syndrome" whose symptoms Stewart (1972) denotes as 1) disorder (hence statistical averaging is necessary), 2) efficient mixing (which implies molecular processes are not dominant) and 3) vorticity continuously distributed in three dimensions (which precludes the simplification of two-dimensionality). The usual, Reynolds' decomposition of the instantaneous velocity and pressure into a "mean" and deviation from the mean, i.e., "turbulence" via homo/heterodyning in the (non-linear) Navier-Stokes equations yields a new quantity, - $\overline{u_i u_j}$, the "extra," Reynolds' stress tensor. Some sort of "closure" hypothesis, e.g.-quasi-normal, "eddy viscosity," transport equation for $\overline{u_i u_j}$, must be made to enable even "supercomputers," to "solve" the turbulence equations. All known calculation methods incorporate some sort of turbulence "model" to reduce the infinite hierarchy of equations, under Reynolds' averaging, to a finite set.

All such models suffer from a certain ad hoc - nature. Townsend (1956, 1976) developed a dual-structure model wherein the turbulence field is, somewhat arbitrarily, decomposed into "large eddies" which presumably are dominant contributors to the Reynolds' stress and "small eddies" which "feed" on the large eddies as these, in turn "feed" upon the average flow to gain their energy. Townsend's concepts have been developed by Lumley and others into a dual approach, one extractive and the other predictive as outlined below.

II. PODT-SAS*Extraction from Experiment

Lumley (1967) gave the first rational definition of "large eddy" and produced a scheme for isolating these from experimental, two-point velocity co-variances in the form of an integral eigen-value problem:

$$\iiint R_{ij}(\underline{x},\underline{x}')\phi_j(\underline{x}')d\underline{x}' = \lambda\phi_i(\underline{x}) \qquad \text{-(1)}$$

where R_{ij} is the average of the two-point Reynolds stress:

$$R_{ij}(\underline{x},\underline{x}') = \overline{u_i(\underline{x})u_j(\underline{x}')}$$

and showed that this Proper Orthogonal Decomposition Theorem (PODT) is optimal in the sense that R_{ij} can be expanded in a series:

$$R_{ij} = \sum_{n=1}^{\infty} \lambda^{(n)}\phi_i{}^{(n)}(\underline{x})\phi_j{}^{(n)}(\underline{x}') \qquad \text{-(2)}$$

where truncation of the series (2) at any finite term recovers the maximum of R_{ij}

Payne (1966) performed the Lumley decomposition on Grant's (1958) data in the far wake of a circular cylinder and Lemmerman (1976) extracted the large eddies from extensive flat-plate boundary-layer data (Grant 1958, Tritton 1967). Unfortunately, both empirical data sets were rather sparse so that considerable ingenuity was required in both cases to augment the given data bases. A third geometry, i.e.- round jet, is currently under experiment (Reed, 1977); this is the first experiment specifically designed with PODT-SAS* in mind.

Payoff of PODT-SAS extracted large eddies should be at least two-fold; 1) determination of scales of motion which strongly interact with the mean flow and 2) generation of a "Lumley Decomposition" of the Reynolds' stress:

$$\overline{-u_iu_j} = B_{ij} + 1/3(B_{kk}-q^2)\delta_{ij} + \frac{\nu_{se}}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_j}\right) \qquad \text{-(3)}$$

*SAS = Structural Analysis System (Payne 1966, Lemmerman 1976, Payne 1977) which is an obvious extension (and hopefully improvement) over the usual "eddy viscosity" i.e. -

$$-\overline{u_i u_j} = \frac{\nu_e}{2} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - q^2/3\ \delta_{ij} \qquad -(4)$$

"obvious" because eq(3) has incorporated empirical "large eddy" information and, hence, the ν_{se}, "small eddy viscosity" models <u>only</u> that portion of the turbulence, <u>not</u> the entire turbulent field as does ν_e in eq(4). Hence, one has a hope, partially verified by preliminary calculations of B_{ij}, the "big eddy" correlation of Lemmerman (1976), that ν_{se} will be a simple function of, perhaps, y alone.

III. OLP* PREDICTIONS

Lumley (1966) postulated a variational principle which yields a quasi-linear differential eigen-value problem for the unstable modes of a <u>turbulent</u> velocity profile:

$$S_{ij}u_j = \phi,_i + \frac{\partial}{\partial x_j}\left(\nu_T(u_{i,j} + u_{j,i})\right) \qquad -(5)$$

where u_j is the perturbation velocity, S_{ij} is the mean rate-of-strain, ϕ is a Lagrange multiplier and ν_T is "eddy viscosity."

It should be recalled that usual laminar flow stability analyses assume small perturbations which linearize the equations of motion; this luxury is not possible in turbulence because the inherent "driver" of turbulence is $\overline{u_i u_j}$, the Reynolds' stress. Although there is no precise mathematical comparison of the eigen-solutions of OLP to those of PODT-SAS, there are physical reasons why one expects at least qualitative agreement: 1) predictions of linear theory agree well with most details of transition due, presumably, to extremely rapid growth rates of (linearily) unstable modes and 2) presumably the Reynolds' stress levels are

*OLP = Orr (1907), Lumley (1966), and Payne (1968) method of flow stability analysis.

maintained by a non-linear instability mechanism which permits the large eddies to extract energy efficiently from the base flow.

In any case, Payne (1968) via OLP predicted unstable modes which compared favorably, in wave number space, to PODT-SAS extractions for the 2-D wave. Unfortunately, due to the inherent phase ambiguity of complex eigen-vectors across $\underline{k}$-space, comparison in laboratory coordinates was not possible. A brief outline of Payne's (1968) OLP calculations follows:

Assumptions of planar homogeneity permit a 2-D Fourier transform of eq.(5) which becomes, after cross-differentiation to eliminate ϕ:

$$\left.\begin{aligned} L_1\ (\psi_1) &= M\ (\psi_2) \\ L_2\ (\psi_2) &= M\ (\psi_1) \end{aligned}\right\} \qquad -(6)$$

where $L_1 = k^2\nabla^2$, $L_2 = (k_3{}^2 - D^2)\ \nabla^2$, $M = ik_1\ \nabla^2 D + \frac{1}{2}\ k_3 R_T U'$

are linear operators and $D = \frac{d}{dy}$, $\nabla^2 = D^2 - k^2$

Further cross-differentiation of (6) yields

$$\left.\begin{aligned} \nabla^4\psi_1 &= L_o\ (U'\psi_1) + L_{12}(U'\psi_2) \\ \nabla^4\psi_2 &= L_o\ (U'\psi_2) + L_{21}(U'\psi_1) \end{aligned}\right\} \qquad (7)$$

where L_o, L_{12}, are linear operators, eq. (7) was converted, via Green's functions to coupled, integral and thence to matrix equations:

$$\psi_i(\underline{k},y) = \frac{1}{2}\ R_T\ k_{ij}\ \psi_j \qquad (8)$$

A matrix eigen-value problem which was solved via an iteration scheme (Lumley, 1970).

IV. Comparison of PODT-SAS Extraction with OLP Predictive Results (See Payne 1968, 1977).

As mentioned in Section III, these comparisons were restricted to k-space because the OLP predictions (as of then) were unable to be transformed back to laboratory coordinates. One should also note the somewhat different interpretations of eigen-solutions of the two methods:

	λ, eigenvalue	ψ_1, eigenvector
OLP (Prediction)	Stability Parameter	Unstable modes (of turbulent profile)
PODT-SAS (Extraction)	Mean Square Energy	Strong, "Large" eddies (of turbulence)

Hence, criterion for inter k-grid relative amplitudes (for inverse F.T.) is lacking in the case of OLP, whereas the weighting factor for PODT-SAS is simply λ, the mean square energy. Herein lies a major piece of work with, possibly, "vector" processors; namely, one may be able, with new computing machinery available in the 1980-85 time frame, to redo the PODT-SAS and OLP analysis without the homogeneity assumptions. This means that all calculations will occur in laboratory space and all Fourier transformations, the major time (CPU) consumer, avoided. Direct, quantitative comparison of PODT-SAS large eddies extracted from experimental data can then be made with the OLP predictions of the most unstable modes of the turbulent velocity profile.

V. Summary

a. PODT-SAS extractions have been successful in extracting the "Large Eddy" structure in two flow prototypes, the 2-D wake (Payne 1966, Payne and Lumley 1967) and the flat-plate boundary-layer (Lemmerman 1976, Lemmerman and Payne 1977) and a third, the round jet, is in progress (Reed 1977).

b. OLP predictions have been accomplished in one flow prototype, the 2-D wake (Payne 1968) and are in progress for a second, the flat-plate boundary-layer (Payne 1977).

c. Impact of PODT-SAS extractions appears to be at least two-fold: 1) grid generation for "sub-grid" modelling of the smaller scales of turbulence in the dynamical equations and 2) possible generation of prototype families of fundamental modes for various flow geometries since the large scales are presumably independent, or at most weakly dependent, of Reynolds' number.

d. Impact of OLP may be primarily corroborative and, possibly, extrapolative to new geometries wherein a dearth of empirical data exists.

Cited References

Grant, H. L. (1958), Journal Fluid Mechanics, p. 149.

Lemmerman, L. A. (1976) Ph.D. Dissertation, University of Texas at Arlington.

Lemmerman, L. A. and Payne, F. R. (1977), "Extracted Large Eddy Structure of a Turbulent Boundary Layer", AIAA Paper 77-717, 10th Fluid & Plasmadynamics Conference, Albuquerque.

Lumley, J. L. (1966), "Large Disturbances to the Steady Motion of a Liquid", Memo/Ordnance Res. Lab., Penn State, 22 August.

Lumley, J. L. (1967), "The Structure of Inhomogeneous Turbulent Flows", Paper presented in 1966 at Moscow and printed in Doklady Akad, Nauk, SSSR, Moscow.

Lumley, J. L. (1970), Stochastic Tools in Turbulence, Academic Press, New York and London.

Orr, W., (1907), "The Stability ... of Steady Motions of a Perfect Fluid and a Viscous Fluid", Proc. Royal Irish Acad., Sec. A. Vol. XXVII, p. 69.

Payne, F. R. (1966), Ph.D. Thesis, Penn State Univ. and rep. to U.S.N./ONR under Nonr 656(33).

Payne, F. R. and Lumley, J. L. (1967), Phys. Fluids, SII, p. S194.

Payne, F. R. (1968), Predicted Large Eddy Structure of a Turbulent Wake, rep. to U.S.N./ONR under Nonr 656(33).

Payne, F. R. (1977), "Comparison of PODT-SAS Extractive with OLP-Predictive Eigen-Structures in a Turbulent Wake," SIAM Fall 1977 meeting.

Reed, X. B., Jr., et al (1977), Proc. Symposium on Turbulent Shear Flows, Penn State, April 18-20, p. 2.23-2.32.

Rubesin, M. W. (1975), "Subgrid or Reynolds Stress Modeling for Three Dimensional Turbulent Computations", NASA SP-347.

Stewart, R. W. (1972), "Turbulence," Illustrated Exp. in Fluid Mech., p. 82-88, MIT Press, Cambridge.

Townsend, A. A. (1956), The Structure of Turbulent Shear Flow, Cambridge University Press.

Townsend, A. A. (1976), Ibid, 2nd edition.

Tritton, D. J. (1967), Journal Fluid Mechanics, p. 439.

SESSION 7

Panel on GRID GENERATION

Joe F. Thompson, Chairman

D22

REMARKS

ON

BOUNDARY-FITTED COORDINATE SYSTEM GENERATION

Joe F. Thompson
Department of Aerophysics and Aerospace Engineering
Mississippi State University
Mississippi State, Mississippi 39762

Computational fluid dynamics must, of course, be able to treat flows about bodies of any shape. Furthermore, it must be easy to change the shape of the body under consideration, so that design studies can be performed economically via input devices and a single code without reprogramming. In addition, the simulation must include complex bodies composed of multiple parts, e.g., wings with flaps, and must provide for dynamic changes in shape. It is also important that the device providing treatment of arbitrary shapes be such that it can be incorporated into new codes as they are developed in a straightforward manner.

Now it may be that numerical simulations of fluid mechanics may someday be developed which do not utilize any type of mesh system. However, at present computational fluid dynamics is based on the numerical solution of partial differential equations, and some mesh system is an inherent part of such codes, whether the solution is of the finite difference or finite element type. This will continue to be the case in the foreseeable future.

The essential part of numerical solutions of partial differential equations is the representation of gradients and integrals by, respectively, differences between points and summations over points. In order for such numerical representations to be accurate, it is necessary that these points be more closely spaced in regions of large gradients. The need for accurate representation is particularly acute near body surfaces, since the boundary conditions are generally the most influential part of a partial differential equation solution. This is especially true of viscous solutions at high Reynolds number, where very large gradients occur in the boundary layer.

If the boundaries do not pass through points of an ordered mesh, then interpolation among neighboring points must be used to represent the boundary conditions. This is possible, of course, but introduces error and irregularity in the most sensitive region of the solution. The irregularity of spacing that then occurs near the boundary makes it very difficult to achieve a close enough spacing of points near the boundary without resorting to either an excessively large number of points or to a patched-together grid system with consequent complexity of code.

Although solutions can be formulated with a random point distribution, efficient codes require some organization of the mesh structure. This can be accomplished by having the points aligned on some mesh of intersecting lines, one of which lines coincides with the body surface.

It is both more accurate and more convenient to have a line of mesh points lying on the boundary. This allows the points to be distributed along the boundary as desired, and also allows the boundary conditions to be represented logically using the boundary points and adjacent points. With regular lines of points surrounding the boundary, concentration of points near the boundary can be achieved economically without complicating the code.

What is needed, then, is a general curvilinear coordinate system that can fit arbitrary shapes in the same way that cylindrical coordinates fit circles. The defining characteristic of such a system is that some coordinate line be coincident with the body contour, i.e., that one of the curvilinear coordinates be constant on the body contour. (For instance, in cylindrical coordinates, a circular body has the radial coordinate constant on its contour.) This coincidence of a coordinate line with the body contour must occur automatically, regardless of the body shape, and must be maintained even if the body undergoes deformation.

With such a grid having a coordinate line coincident with the body surface, boundary conditions can be represented accurately, and the point distribution is efficiently organized.

This type of general "boundary-fitted" curvilinear coordinate system [1] can be generated by defining the curvilinear coordinates to be solutions of an elliptic partial differential system in the physical plane. The boundary conditions of this elliptic system are the specification of one coordinate to be constant on each boundary surface, and the specification of a monotonic variation of the other over the surface. If these partial differential equations are transformed by interchanging the dependent and independent variables, so that the Cartesian coordinates become the dependent variables, then the Cartesian coordinates of the grid points can be generated by numerically solving the transformed partial differential equations in the transformed plane, which is by nature rectangular regardless of the shape of the boundaries in the physical plane.

Similarly, any partial differential system of interest may be transformed to the curvilinear coordinate system, so that the solution can be done numerically in the rectangular plane. Since time derivatives can also be transformed to be taken with the curvilinear coordinates, rather than the Cartesian coordinates, held constant, the computational mesh in the transformed plane is fixed even though the physical boundaries may be deforming.

All computation, both to generate the mesh system and to solve the partial differential equations of interest, can thus be done on a fixed square mesh in the transformed plane regardless of the shape and number of bodies (boundaries) in the physical plane, movement thereof, or the mesh spacing in the physical plane. The transformed equations are naturally more complicated than those in Cartesian coordinates but all boundary conditions now occur on straight boundaries. A system with simple equations but complicated boundary conditions has thus been exchanged for a system with complicated equations but simple boundary conditions - generally an advantageous trade.

This general procedure of coordinate generation contains conformal mapping as a special case but, unlike this more restricted case, the general procedure is extendible in principle to three dimensions and allows coordinate lines to be concentrated as desired. This control of the coordinate system can be accomplished by varying terms in the partial differential equations for the coordinates, through input to the code.

General curvilinear meshes fitted to all boundaries of a region containing any number of arbitrary-shaped bodies can thus be automatically generated by a code requiring only the input of the desired distribution of points on the boundaries. The spacing of the coordinate lines in the field can be controlled through input to the code. Many different coordinate configurations can be generated without changing the code, as has been shown in published examples [1-4,6]. Several examples are included in Figures 1-3. In these figures, only a portion of the coordinate system is shown in the interest of space.

This general procedure of coordinate generation is considered preferable to the alternatives of (1) a random point distribution, because the point distribution is more easily controlled and has more regularity leading to more efficient codes, (2) conformal mapping, because control of the line spacing and extension to three dimensions are desirable, and (3) analytic transformations, because these must be devised for each new boundary configuration. This general mesh can be used in either finite difference or finite element solutions of any system of partial differential equations of interest.

The most important area of current research is in the control of the curvilinear coordinate lines in the field. In the original development this control was exercised through inputting amplitudes and decay factors for exponential terms that caused attraction of coordinate lines to other lines and/or points. This requires some experience, of course, to implement effectively. Recently, procedures have been developed whereby a specified number of coordinate lines can be located within a boundary layer at a specified Reynolds number. These procedures have been used with some success at Reynolds number of 10^6. [5,6]. (See Fig. 2).

Some discretion is necessary, however, in the concentration of coordinate lines, since there are truncation error terms proportional to the rate of change of the coordinate spacing and to the deviation from orthogonality. [4]. This truncation error can introduce artificial diffusion which may even be negative. This is an area in need of further study to devise procedures for control of the truncation error or to devise difference representations that reduce it.

Another procedure currently under study is the coupling of the elliptic system for the coordinates with the differential equations of motion so that the flow solution itself causes coordinate lines to concentrate in regions of large gradients as they develop. This procedure has had some success in causing lines to concentrate in the region of a bow shock (Fig. 3). Related to this is coupling through a deforming boundary, and some free surface solutions have been developed using this feature (Fig. 4). Another obvious application is in the automatic concentration within a developing boundary layer.

This coupling of the coordinate system with the flow solution is a particularly attractive area for further effort, with the ultimate goal of making the mesh system automatically sense areas where concentration of points is needed, moving the mesh accordingly and also monitoring and controlling its own truncation error. Current efforts are also being directed toward three-dimensional coordinate systems (see Fig. 5).

In summary, a general coordinate mesh generation procedure must be incorporated in computational fluid dynamics codes. This should ultimately be in an interactive mode with the flow solution, so that the coordinate mesh adjusts itself as the flow develops. The boundary-fitted coordinate system generated by solving elliptic systems seems to hold the most promise.

REFERENCES

1. Thompson, J. F., Thames, F. C., Mastin, C. W., "TOMCAT - A Code for Numerical Generation of Boundary-Fitted Curvilinear Coordinate Systems on Fields Containing any Number of Arbitrary Two-Dimensional Bodies," Journal of Computational Physics, 24, 274 (1977).
2. Thames, F. C., Thompson, J. F., et. al., "Numerical Solutions for Viscous and Potential Flow about Arbitrary Two-Dimensional Bodies using Body-Fitted Coordinate Systems," Journal of Computational Physics, 24, 245 (1977).
3. Thompson, J. F., Thames, F. C., Shanks, S. P., Reddy, R. N., Mastin, C. W., "Solutions of the Navier-Stokes Equations in Various Flow Regimes on Fields Containing any Number of Arbitrary Bodies Using Boundary-Fitted Coordinate Systems," Proceedings of Vth International Conference on Numerical Methods in Fluid Dynamics, Enschede, the Netherlands, Lecture Notes in Physics, 59, Springer-Verlag, 1976.
4. Thompson, J. F., Thames, F. C., and Mastin, C. W., "Boundary-Fitted Coordinate Systems for Solution of Partial Differential Equations on Fields Containing any Number of Arbitrary Two-Dimensional Bodies," NACA CR-2729, 1977.
5. Reddy, R. N., and Thompson, J. F., "Numerical Solution of Incompressible Navier-Stokes Equations in the Integro-Differential Formulation Using Boundary-Fitted Coordinate Systems," Proceedings of AIAA 3rd Computational Fluid Dynamics Conference, Albuquerque, NM, 1977.
6. Bearden, John H., "A High Reynolds Number Numerical Solution of the Navier-Stokes Equations in Stream Function Vorticity Form," MS Thesis, Mississippi State University, August 1977.
7. Long, W. Serrill, "Two-Body Coordinate System Generation Using Body-Fitted Coordinate System and Complex Variable Transformation," MS Thesis, Mississippi State University, August 1977.
8. Shanks, S. P. and Thompson, J. F., "Numerical Solution of the Navier-Stokes Equations for 2D Hydrofoils in or Below a Free Surface," 2nd International Conference on Numerical Ship Hydrodynamics, Berkeley, CA, 1977.

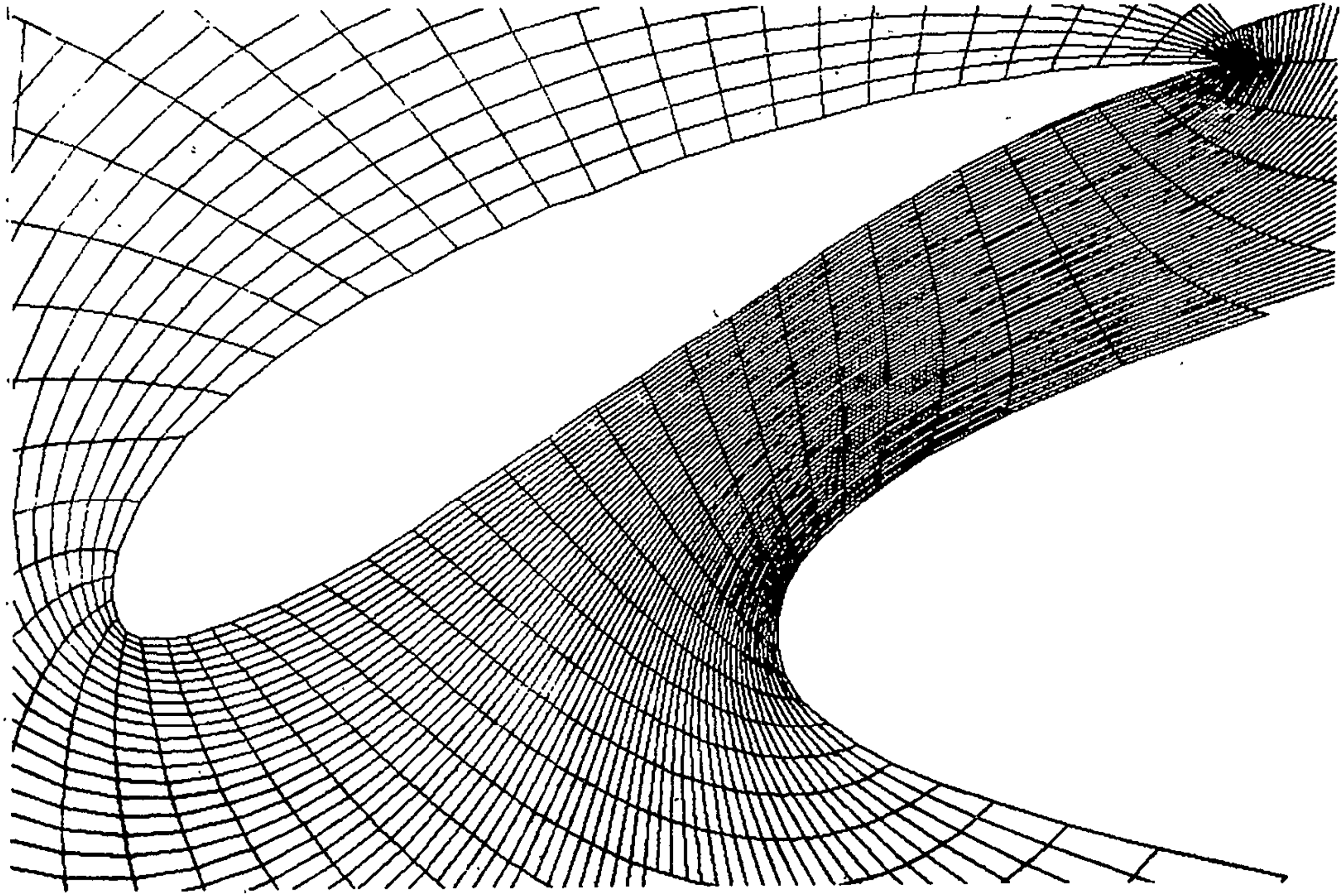


Figure 1. **Expanded View of Physical Plane Plot of Wing-Slat Coordinate** System [7]

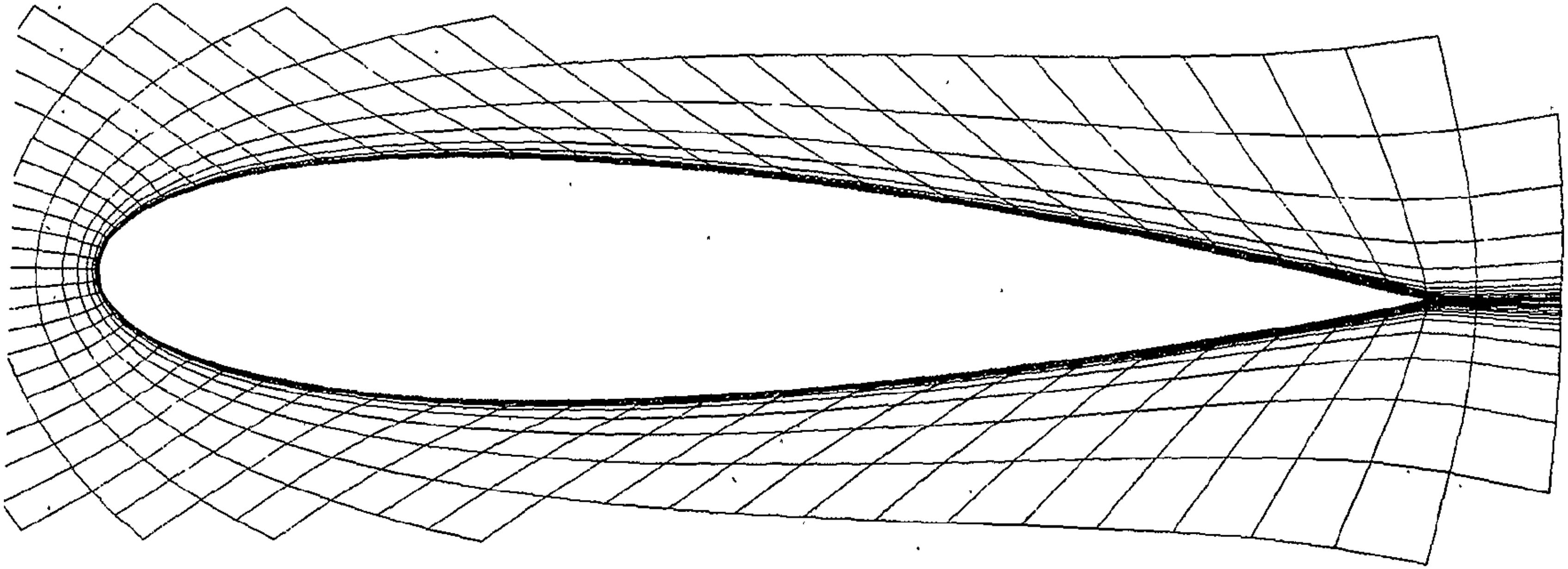


Figure 2 Detail of Coordinate System Near Airfoil

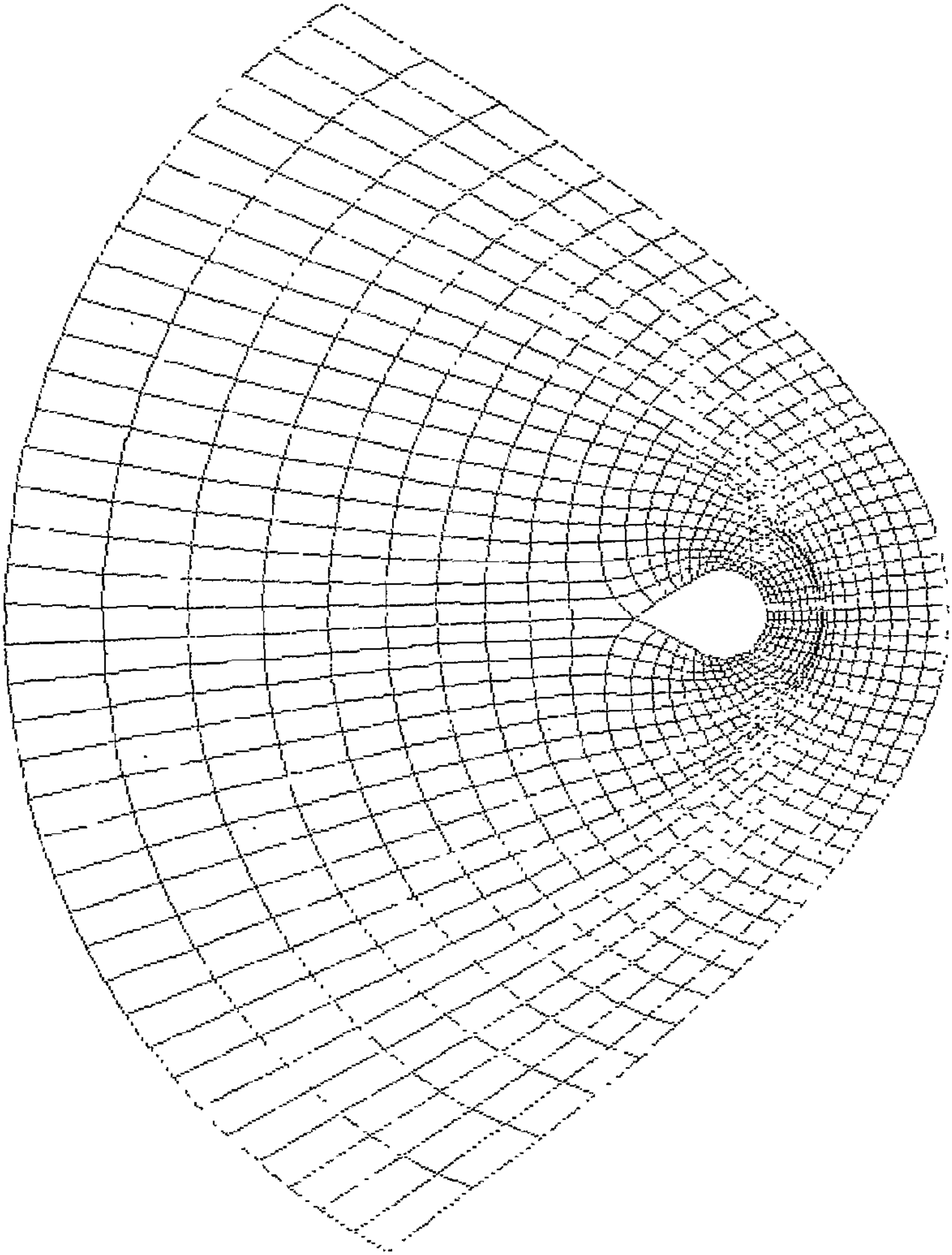


Figure 3. Coordinate Lines Contracting Dynamically into Bow Shock

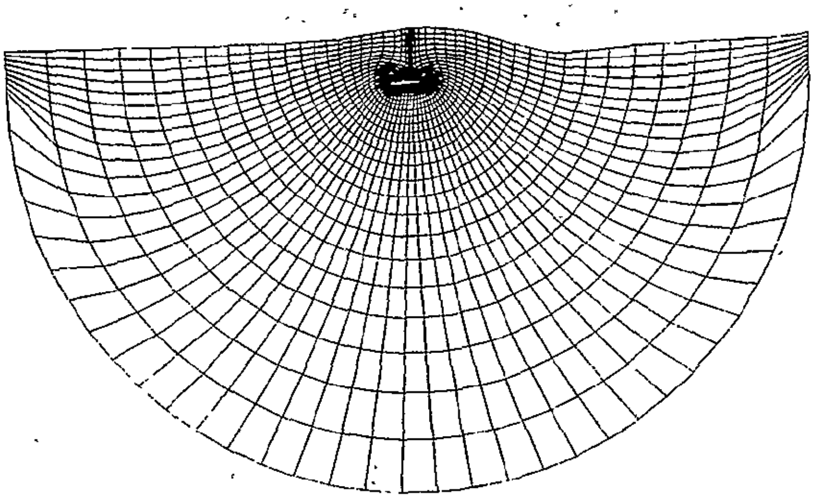


Figure 4. Coordinate System Dynamically Following Free Surface [8]

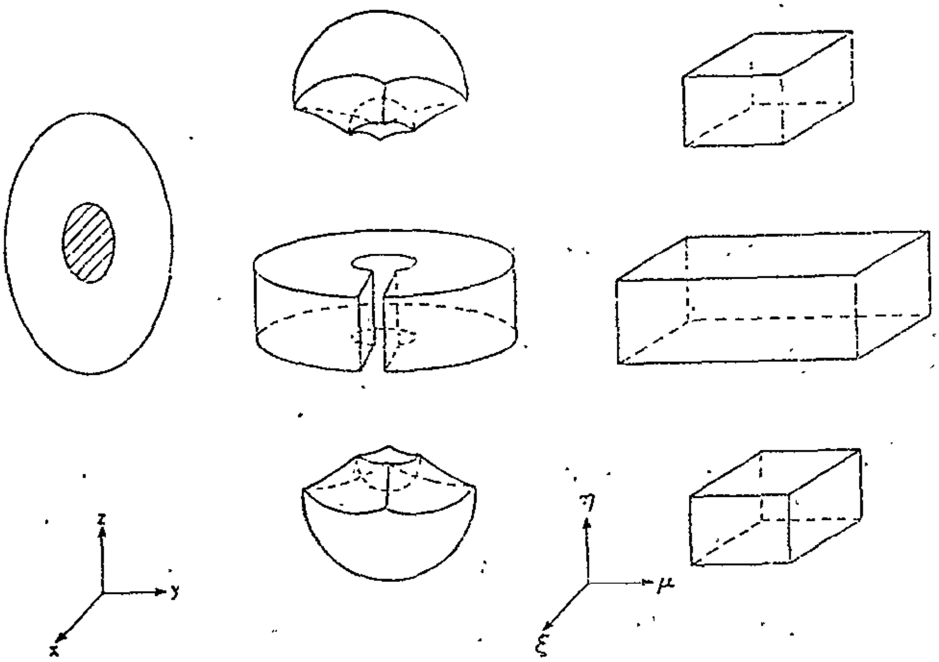


Figure 5. Partition and Transformation of the Region about a Three-Dimensional Body

D23

FINITE ELEMENT CONCEPTS IN COMPUTATIONAL AERODYNAMICS

A. J. Baker
The University of Tennessee
Knoxville, Tennessee

SUMMARY

Finite element theory is employed to establish an implicit numerical solution algorithm for the time-averaged unsteady Navier-Stokes equations. Both the multi-dimensional and a time-split form of the algorithm are considered, the latter of particular interest for problem specification on a regular mesh. A Newton matrix iteration procedure is outlined for solving the resultant non-linear algebraic equation systems. Multi-dimensional discretization procedures are discussed with emphasis on automated generation of specifiable non-uniform solution grids and accounting of curved surfaces. The time-split algorithm is evaluated with regards to accuracy and convergence properties for hyperbolic equations on rectangular coordinates. An overall assessment of the viability of the finite element concept for computational aerodynamics is made.

INTRODUCTION

The finite element theory for support of numerical solution algorithms in computational fluid mechanics emerged in the late 1960's. Up to this time, considerable effort had been expended on the "search for variational principles" (cf., ref. 1), since finite elements were considered constrained to differential descriptions possessing an equivalent extremal statement. The Method of Weighted Residuals (MWR) was rediscovered (cf., ref. 2), and with proper interpretation of an assembly operator, MWR could be directly employed to establish a finite element algorithm for any (non-linear) differential equation. Early numerical results for the boundary layer (ref. 3) and two-dimensional Navier-Stokes (ref. 4,5) equations confirmed the viability of the concept in fluid mechanics. Since 1971, a virtual flood of finite element solutions in many branches of fluid mechanics has inundated the technical literature. Yet, the true value of the method as a preferable alternative to finite differences remains unanswered, due both to the significant advances made in finite difference methodology and the "status incommunicatus" between respective researchers.

A significant difficulty associated with finite difference procedures in elliptic fluid flow descriptions has been getting off the "unit square", and in particular the establishment of equal-order accurate boundary condition constraints on domain closure segments not aligned parallel with a global coordinate surface. In distinction, the finite element concept manifests utter disregard for the global coordinate system, and can directly enforce gradient boundary condition constraints anywhere within a consistent order of accuracy. However, recent developments in regularizing coordinate transformations, on two-dimensional space at least (cf., ref. 6-7), have given rebirth to recursive and tri-diagonal finite difference procedures for non-regular shaped domains. However, maintaining a consistent order of accuracy in the differenced transformed differential equation, grid resolution near a wall in turbulent flow, and extension to three-dimensional space remain to pose difficulties requiring resolution. Conversely, these are not a problem in a finite element based algorithm, but the resultant matrix structure, while banded, will be much larger and hence require significantly more core if not also computer CPU for execution.

Numerical solution of the hyperbolic inviscid Euler equations has commanded great attention in finite difference methodology, and almost none using finite element concepts. MacCormack's time-splitting algorithm (ref. 8) has become an industry standard of proven accuracy. Recently, Beam and Warming (ref. 9) proposed an implicit non-iterative, finite difference time-splitting algorithm. In an allied field (cf., ref. 10), the implicit algorithm resulting from elementary finite element theory applied to an inviscid linear hyperbolic transport equation was predicted superior to equal complexity finite difference forms. Computational results using multi-dimensional (i.e., non-tri-diagonal) finite elements (ref. 11) confirmed the superior behavior predicted by the lower dimensional theory. Recently, under NASA Grant NSG-1391, the concept of a time-split implicit finite element algorithm, for non-linear hyperbolic and/or elliptic partial differential equations, has been established. Numerical results indicate the time-split algorithm superior to both the various finite difference, and the multi-dimensional finite element forms, with regards to storage, CPU and solution accuracy. Of considerable potential value, the time-split algorithm appears directly extendible to three-dimensions and higher order accuracy. Hence, finite element concept might prove to be competitive for solution of the hyperbolic equation systems of interest in certain branches of aerodynamics.

This paper presents an overview of the key aspects of finite element solution methodology for computational fluid mechanics, and their potential impact on future computer system design. The primary focus for a general multi-dimensional specification is grid formation and economical tabulation of element connection and boundary data. Introductory concepts on a time-split form for a multi-dimensional problem specification are also presented.

PROBLEM SPECIFICATION

The prime objective is solution of various forms of the time-averaged Navier-Stokes equations, including the differential equations of a second order (at least) closure model for turbulence. The continuity and momentum equations illustrate the essential character of the system; in tensor divergence form, with summation on repeated Latin subscripts

$$L(\bar{\rho}) = \frac{\partial\bar{\rho}}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j) = 0 \qquad (1)$$

$$L(\bar{\rho}\tilde{u}_i) = \frac{\partial(\bar{\rho}\tilde{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}\left[\bar{\rho}\tilde{u}_j\tilde{u}_i + \frac{\partial\bar{p}}{\partial x_i} - (\bar{\sigma}_{ij} - \overline{\rho u_i' u_j'})\right] = 0 \qquad (2)$$

In eq(1)-(2), $\bar{\rho}$ is the time-averaged density, $\tilde{u}_j$ is the mass weighted time-averaged velocity (cf., ref. 12), $\bar{p}$ is the time-averaged pressure, $-\overline{\rho u_i' u_j'}$ is the Reynolds stress tensor, and $\bar{\sigma}_{ij}$ is the time-averaged Stokes stress tensor

$$\bar{\sigma}_{ij} \equiv \frac{\bar{\mu}}{Re}\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij}\right] \tag{3}$$

Equation (2) is hyperbolic for inviscid flows, and elliptic for laminar viscous flows. An elliptic character can also be imbedded into the inviscid form by modeling the Reynolds stress in terms of the mean-flow strain-rate tensor and an effective diffusion coefficient. For example, using the turbulence kinetic energy-dissipation model, the elementary form of the constitutive equation involves a scalar kinematic coefficient as

$$-\overline{\rho u_i' u_j'} \equiv \bar{\rho}\nu^t\left[\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right] \tag{4}$$

where, for example (ref. 13)

$$\nu^t \equiv C_\nu k^2 \varepsilon^{-1} \tag{5}$$

and C_ν is a correlation coefficient. Combining eq(3)-(5) and defining an effective diffusion coefficient

$$\mu^e \equiv \frac{\bar{\mu}}{Re} + \bar{\rho}\nu^t \tag{6}$$

renders eq(2) elliptic for all cases. Equation (2) also becomes elliptic in the absence of definitions of the type eq(4) if the wall layer is resolved.

The solution to eq(1)-(6) lies on the bounded open domain $\Omega \equiv R^n \times t \in x_i \times [t_0,t)$ where $1 \le i \le n$. Boundary conditions on $\partial\Omega \equiv \partial R \times t$, where q is identified as a generalized dependent variable, are of the form

$$\ell(q) \equiv a_1 q + a_2 \frac{\partial q}{\partial x_i}\hat{n}_i - a_3 = 0 \tag{7}$$

An initial condition is also specified on $\Omega_0 \equiv R^n \times t_0$. A finite element solution algorithm assumes elements of the dependent variable vector, $\{q\}$

$$\{q\} = \{\bar{\rho}, u_i, k, \varepsilon, \ldots\} \tag{8}$$

interpolated on non-overlapping sub-domains R_e of R^n, where $\cup R_e = R^n$, and separable on Ω_e as

$$q_e^*(x_i,t) \equiv \{N_k(x_i)\}^T\{0(t)\}_e \tag{9}$$

The elements of the "shape functions" $\{N_k(x_i)\}$ are typically k^{th} degree polynomials written on x_i and constitute a cardinal basis. The elements of $\{Q(t)\}_e$ are the time-dependent values of q_e^* at coordinates of $R_e \cup \partial R_e$ called nodes.

The finite element solution algorithm is established by insertion of eq(9) into eq(1)-(7), and setting to zero the integral of the inner product with $\{N_k\}$ as

$$\underset{e}{S}\left[\int_{R_e}\left\{N_k(x_i)\right\} L\left[q_e^*(x_i,t)d\tau \;-\; \lambda\int_{\partial R_e \cap \partial R^n \neq 0}\left\{N_k(x_i)\right\}\ell\left[q_e^*(x_i,t)d\sigma\right.\right.\right] \equiv \{0\} \tag{10}$$

Lambda is a convenient multiplier, $\mathop{S}_{e}$ is the elemental assembly operator, and eq(10) is an ordinary differential equation system for the time-evolution of the totality of elemental expansion coefficients $\sum_e \{Q(t)\}_e = \{Q\}$. The general form is

$$\mathop{S}_{e}\left[[C_k]_e\{Q\}'_e + [K_k]_e\{Q\}_e + \{y\}_e\right] \equiv \{0\} \tag{11}$$

The superscript prime denotes the ordinary derivative column matrix, and $[C_k]_e$ and $[K_k]_e$ are full square matrices, of rank equal to the order of $\{Q\}_e$, hence related to the degree k of the polynomial sets in $\{N_k(x_i)\}$ and the dimension n of R^n. They are assumed generally dependent upon q, and all parameters in the defining differential equation are assumed interpolated on R_e using eq(9).

Equation (11) is distinct from any finite difference equivalent in that the initial-value behavior is coupled for all $k \geq 1$. This is of no computational consequence, and of considerable value regarding solution accuracy, provided an implicit integration algorithm is employed. Any single or multi-step finite difference integration algorithm is applicable; for example, consider

$$\{Q\}_{k+1} = \{Q\}_k + h\left[\theta\{Q\}'_{k+1} + (1-\theta)\{Q\}'_k\right] \tag{12}$$

where k is the time step index, h is the integration step-size and $0 \leq \theta \leq 1$ controls implicitness (note $\theta = \frac{1}{2}$ is the trapezoidal rule). A matrix iteration algorithm, based upon a modified Newton procedure which exhibits quadratic convergence for eq(10) linear, is

$$\left[J\left(\{F\}_{k-\ell}\right)\right]\{\delta Q\}^{p+1}_{k+1} = -\left\{F\left(\{Q\}^{p+1}_{k+1}\right)\right\} \tag{13}$$

where $\{\delta Q\}$ is the iteration matrix

$$\{Q\}^{p+1}_{k+1} \equiv \{Q\}^{p}_{k+1} + w\{\delta Q\}^{p+1}_{k+1} \tag{14}$$

and w is a relaxation factor. In eq(13), $\{F\}$ is the homogeneous form of eq(11), using eq(12), which yields

$$\begin{aligned}\left\{F\left(\{Q\}^{p+1}_{k+1}\right)\right\} \equiv \mathop{S}_{e}\Bigg[&\left[C\left(\{Q\}^{p}_{k+\frac{1}{2}}\right)\right]_e\left(\{Q\}^{p}_{k+1} - \{Q\}_k\right)\\ &+ h\theta\left[K\{Q\}^{p}_{k+1}\right]_e\{Q\}^{p}_{k+1} + h(1-\theta)\left[K\left(\{Q\}_k\right)\right]_e\{Q\}_k\\ &- h\left(\theta\{y\}_{k+1} + (1-\theta)\{y\}_k\right)\Bigg]\end{aligned} \tag{15}$$

The subscript $k+\frac{1}{2}$ implies evaluation at the mid-point of h. Similarly, [J] is the Jacobian of eq(15), i.e.,

$$[J] \equiv \frac{\partial\{F\}}{\partial\{Q\}} = \mathop{S}_{e}\left[[C]_e + h\theta[K]_e + \left(\frac{\partial[C]_e}{\partial\{Q\}_e} + h\theta\frac{\partial[K]_e}{\partial\{Q\}_e}\right)\{Q\}_e\right] \tag{16}$$

For any evaluation of the Jacobian, the vanishing of $\{\delta Q\}$ as solution to eq(13) implies convergence. In practice, all elements of $\{\delta Q\}$ do not vanish simultaneously, and a convergence test is employed. The initial estimate, i.e., $\{Q\}^1_{k+1}$ for any iteration is obtained as the solution of eq(13) for $\theta = 0$. Should any guess fall outside the convergence interval for $\{\delta Q\}$, reduction of step-size h and reformation of eq(15)-(16) is required.

AUTOMATED DISCRETIZATION AND REFINEMENT

Aerodynamic flowfield determinations can require solution of three-dimensional elliptic boundary value problems on domains bounded by non-coordinate surfaces. An elementary example is solution for the irrotational perturbation potential distribution for steady flow about an aerodynamic shape. The perturbation function definition is

$$u_j \equiv U_\infty \left[\hat{\varepsilon}_j - \frac{\partial \phi}{\partial x_j} \right] \tag{17}$$

where $\hat{\varepsilon}_j$ is the freestream flow unit vector. Substituting eq(17) into eq(1)-(2) and dividing by the freestream sound speed yields

$$L(\phi) \equiv \left[\delta_{ij} - M_\infty^2 \hat{\varepsilon}_i \hat{\varepsilon}_j + (\gamma - 1) M_\infty^2 \delta_{ij} \left(\hat{\varepsilon}_k \frac{\partial \phi}{\partial x_k} - \frac{1}{2} \frac{\partial \phi}{\partial x_k} \frac{\partial \phi}{\partial x_k} \right) \right.$$
$$\left. + M_\infty^2 \left(\hat{\varepsilon}_i \frac{\partial \phi}{\partial x_j} + \hat{\varepsilon}_j \frac{\partial \phi}{\partial x_i} - \frac{\partial \phi}{\partial x_i} \frac{\partial \phi}{\partial x_j} \right) \right] \frac{\partial^2 \phi}{\partial x_i \partial x_j} = 0 \tag{18}$$

The appropriate freestream boundary condition valid for all cases is

$$\frac{\partial \phi}{\partial x_i} \hat{n}_i = 0 \tag{19}$$

where $\hat{n}_i$ is the outward-pointing unit normal vector. At the airfoil, since $u_j \hat{n}_j$ vanishes identically, the boundary condition is

$$\frac{\partial \phi}{\partial x_j} \hat{n}_j = \hat{\varepsilon}_j \hat{n}_j \tag{20}$$

Hence, angle of attack and local airfoil contour (unit normal) are applied directly as boundary conditions and eq(19)-(20) are special cases of eq(7). Equation (18) is valid for all Mach numbers, the slender body assumption can simplify terms, and it can be cast as the linear Laplacian for Mach numbers below transonic.

The key problem specification facet is the requirement for gradient constraint boundary conditions everywhere on ∂R. The order of the resultant matrix equation system is rendered equal to the rank by setting $\phi = \text{constant}$ at only one location. The computational requirement is to non-uniformly distribute nodes on R^n in an efficient manner, and to minimize matrix rank for a given accuracy. One approach, documented by Thompson (ref. 6), is to establish a curvilinear coordinate transformation that places boundary conditions on coordinate surfaces of the transformed system. On R^2 this requires solution of two Poisson equations for the coordinate transformation. The Laplacian on ϕ must also be recast, into a usually non-self-adjoint form on the transformed coordinates, and then solved. Hence, the price of boundary condition regularity is solution of three boundary value problems.

Finite elements provide an alternative approach by allowing boundary condition specification on arbitrarily oriented surfaces, and the single solution of the self-adjoint operator on physical space. Independent coordinate transformations, utilizing isoparametric finite elements, are individually employed on macro-subregions of the solution domain to non-uniformly distribute computational node points (cf., ref. 14). This approach is equally variable on both two- and three-dimensional spaces. The concept is illustrated for R^2 in Fig. 1, for potential flow about an isolated airfoil. This macro-discretization contains 8 quadrilaterals and 4 triangles, and the resultant computer generated computational grid of 648 triangles is illustrated in Fig. 2. The computational grid could equally well employ general quadrilaterals. In Fig. 1, the dots represent locations of input coordinate pairs that describe subdomain boundary curvature. The non-symmetrical disposition of the non-vertex pairs provides the capability for uniform element size gradation in the computational grid. Variation in refinement and non-uniformity is elementary input to the sub-domain specification (cf., ref. 14).

The grid refinement algorithm loops over the subdomains and generates both gridpoint and finite element connection data. For both the quadrilateral and triangular subdomain shapes, a coordinate transformation which maps the physical plane onto a regular local natural coordinate system η_i is

$$x_i = \{N_k(\eta_j)\}^T\{X_i\} \qquad 1 \leq i \leq n \tag{23}$$

The shape functions in eq(23) are polynomials of degree k and formally identical to the dependent variable specification eq(9). The elements of $\{X_i\}$ are the x_i coordinates of the nodes to be generated by the refinement algorithm. Equation (23) can be established on R^n for all n, and shape functions are readily derived for a variety of geometric shapes and polynomial degrees. Since the derivation procedures are well documented, see Zienkiewicz (ref. 15), a two-dimensional quadrilateral with bi-quadratic interpolation facilitates the discussion. The basic transformation is illustrated in Fig. 3. The elements of the bi-quadratic shape function are

$$\{N_2(\eta,\varepsilon)\} = \begin{Bmatrix} \frac{1}{4}(1-\varepsilon)(1-\eta)(-\varepsilon-\eta-1) \\ \frac{1}{4}(1+\varepsilon)(1-\varepsilon)(\varepsilon-\eta-1) \\ \frac{1}{4}(1+\varepsilon)(1+\eta)(\varepsilon+\eta-1) \\ \frac{1}{4}(1-\varepsilon)(1+\eta)(-\varepsilon+\eta-1) \\ \frac{1}{2}(1-\varepsilon^2)(1-\eta) \\ \frac{1}{2}(1+\varepsilon)(1-\eta^2) \\ \frac{1}{2}(1-\varepsilon^2)(1+\eta) \\ \frac{1}{2}(1-\varepsilon)(1-\eta^2) \end{Bmatrix} \tag{24}$$

The η_i origin is at the centroid of R_e, and nodal numbering is given in Fig. 3b. Accuracy of the generated boundary nodes depends upon the shape function ability to interpolate the physical geometry; a quadratic is exactly interpolated. Note in Fig. 3 that the non-vertex nodes are located exactly midside in η_i, but not necessarily midside in x_i. Side node relocation in the physical plane provides for generation of a smoothly varying distribution of generated data over the subdomain, as illustrated in Fig. 2. The coordinate transformation is of the Serendipity family, however, which can yield a non-planar transformation for excessive skewing of the side nodes. Experience indicates these nodes must be placed within the approximate center two-thirds of a closure segment to avoid the problem.

Each of the subdomains is treated independently, but data generated along adjacent subdomain boundaries must be identical. This requires that subdomain boundaries share the common gridpoints, and is accomplished by the linear independence of the η_i coordinates and the fact that eq(24) is a cardinal basis. Triangular-shaped subdomains can be interspersed with quadrilaterals and will yield similar results. Three-dimensional subdomains possess surface interfaces, and similar constraints are applicable. This generality eliminates any requirement for specifying subdomains in a particular order, and input node numbering may begin at any vertex. Output node numbering is rearranged to minimize bandwidth of the resultant algebraic equation system.

It is improbable that the entire spectrum of geometric shape and refinement requirements can be contained within a few subdomain types. The geometric flexibility is open ended, however, and special subdomain functions may be developed and inserted directly into the system. An example is the telescoping quadrilateral with quadratic interpolation illustrated in Fig. 4. This special function provides the means for decreasing the number of generated (triangular) elements in progressing through the subdomain. Fig. 5 illustrates use of the telescoping domain for formation of a fine discretization in the immediate vicinity of a wing tip.

The interpolation function eq(24) is also used to distribute initial and gridpoint data over each subdomain. Removal of the duplicate sets of data along common boundaries is accomplished through construction of a subdomain connection table from the specified data. Table 1 illustrates the connection table for two subdomains of Fig. 1. The storage requirements are twice the sum of the number of subdomain sides.

Table 1
Two-Subdomain Connection Table for Two-Dimensional Airfoil.

Subdomain	Side	Subdomain	Side
1	1	10	3
1	2	0	0
1	3	2	1
1	4	0	0
2	1	1	3
2	2	0	0
2	3	3	1
2	4	0	0

The first two columns are not stored, since they are sequential, and only eight words of storage are required to eliminate boundary duplication. The algorithm loops on subdomains, generating a sequence of local (dummy) node numbers, and then interrogates the connection table. Upon finding a common entry, which is of lower number, it substitutes the local boundary node numbers for that subdomain with the coincident set. Simultaneously, the duplicate generated variable set at the boundary is eliminated. The connection table is formed for all generated elements in the subdomain, and sequential global node numbering is substituted in a simple loop at the end. The connection table can also locate external boundaries since they appear as zeros. Note that only the subdomain data was manipulated, to sequence the generated data, thus maintaining high efficiency. Also, storage requirements are minimized by operating on not more than two subdomains simultaneously. Hence, data for a large problem could be generated interactively on a minicomputer or small time sharing system utilizing disk or tape storage. Combined with a graphics package and a video (CRT) terminal, interactive data deck debug could enhance human efficiency as well.

ECONOMY, ACCURACY AND CONVERGENCE

Efficient matrix solution algorithms have been developed in finite element applications in the structural mechanics community. An LU decomposition of the Jacobian in eq(13) facilitates solution for multiple right-hand sides using back substitution. Since the gradient boundary condition is non-iterative within the finite element algorithm, the solution for several angles of attack could be economically achieved in sequence. Numerical results for incompressible potential flow are documented in reference 14. Solution accuracy is good for surface pressure distribution, but no definitive cycle time comparison is available.

A most interesting aspect regarding accuracy is that the finite element equivalent of a gradient boundary condition constraint is insensitive to the order of accuracy of the interpolation functions $\{N_k(x_i)\}$ spanning R^n. Studies on convergence with discretization refinement have confirmed excellent results for a sample parabolic problem. Shown in Fig. 6 is convergence measured in the gradient of the dependent variable at the surface subjected to a non-homogeneous gradient constraint for $1 < k \leq 3$. The linear element $(k = 1)$ convergence rate of 2.4 is higher than the solution convergence on R^n. The convergence rate and absolute error both improve using higher order accurate $\{N_k\}$.

The economy of multi-dimensional finite element techniques on solution domains with regular grids may be improved by conversion to a time-splitting algorithm. In this instance, the Jacobian in eq(13) is factored into the product

$$[J] \Rightarrow [J_1] \otimes [J_2] \otimes [J_3] \tag{25}$$

as are the components of $\{F\}$. For $1 \leq k \leq 3$, the corresponding matrix structure is 3-, 5- and 7-diagonal respectively. For eq(1) with constant velocity, the linear element $(k = 1)$ algorithm is 4th order accurate in space, and 2nd order in time and neutrally stable for $\theta = \frac{1}{2}$. Convection of a concentration cone on R^2 has been computed using both a bilinear two-dimensional element spanning a regular quadrilateral and the linear element time-split algorithm. Comparison results were generated using a Crank-Nicolson finite difference algorithm. For the test problem, the initial distribution was selected to be interpolated on the least number of grid points allowable to include a point of inflection. The initial-distribution, and the time-split finite element results after 150 time steps are shown in Fig. 7a-b. The Crank-Nicolson results after only 100 time steps, Fig. 7c, are clearly inferior with regards to both numerical diffusion and dispersion. The dispersion error for the two-dimensional finite element solution was just measurably larger than for the time-split solution, and both finite element solutions preserved the peak level within 1% after 150 time steps. Computer CPU and storage for the time-split finite element and Crank-Nicolson programs were essentially identical. The two-dimensional finite element solution required approximately four times the core and computer CPU for the same problem. All results were generated within the identical computer program. Both finite element solutions conserve mass and rms-mass to five significant digits on the coarse grid. Solution accuracy, as measured in a distortion norm, converges quadratically with both discretization refinement and reduction of Courant number.

CONCLUSIONS

The distinct features of finite element numerical solution theory applied to computational aerodynamics have been briefly reviewed. Computer implementation of automated discretization and refinement capitalizes on isoparametric methodology. The capability exists to generate smoothly non-uniform computational grids on domains with curved closure segments arbitrarily oriented with respect to a global coordinate description. Favorable convergence trends are measured on such surfaces for boundary value problem specifications with gradient constraints. Both multi-dimensional and time-split forms of the finite element algorithm have produced accurate results on a coarse grid for a sample hyperbolic equation. The latter form may be economically competitive with finite difference procedures on a regular rectangular mesh. The combination of features, for both initial- and/or boundary value problem specifications in aerodynamics, warrants continued detailed evaluation of the finite element approach in computational aerodynamics.

ACKNOWLEDGMENTS

Portions of this work were supported by NASA Langley Research Center under contract NAS1-14307, administered through the Subsonic-Transonic Aerodynamics Division, and grants NSG-1261 and NSG-1391, administered through the Analysis and Computation Division. The author gratefully acknowledges the contributions of Mr. M. O. Soliman, University of Tennessee and Mr. P. D. Manhardt, Computational Mechanics Consultants.

REFERENCES

(1) Finlayson, B. A. and Scriven, L. E., Int. J. Ht. Ms. Trans., V. 10, pp. 799-821, 1967.

(2) Finlayson, B. A. and Scriven, L. E., App. Mech. Rev., V. 19, pp. 735-748, 1966.

(3) Baker, A. J., AIAA Paper No. 72-108, 1972.

(4) Baker, A. J., Int. J. Num. Mtd. Engr., V. 6, pp. 89-101, 1973.

(5) Baker, A. J., NASA CR-2391, 1974.

(6) Thompson, J. F., Thames, F. C., and Mastin, W. C., J. Comp. Phys., V. 15, pp. 299-319, 1974.

(7) Thames, F. C., Thompson, J. F., Mastin, W. C., NASA SP-347, pp. 469-530, 1975.

(8) MacCormack, R. W., AIAA Paper No. 69-354, 1969.

(9) Beam, R. M. and Warming, R. F., J. Comp. Phys. to appear 1977.

(10) Long, P. E., and Shaffer, W. A., NOAA Tech. Memo. NWS TDL-56, 1975.

(11) Gresho, P. M., Lee, R. L., and Sani, R. L., Proc. 2nd Int. Sym. F.E. Mtd., Italy, June 1976.

(12) Cebeci, T. C. and Smith, A. M. O., Academic Press, New York, 1974.

(13) Launder, B. E., Reese, G. J. and Rodi, W., J. Fl. Mech., V. 68, pp. 537-566, 1975.

(14) Baker, A. J. and Manhardt, P. D., Final Report, NASA Contract NAS1-14307, 1977.

(15) Zienkiewicz, O. C., McGraw Hill, London, 1971.

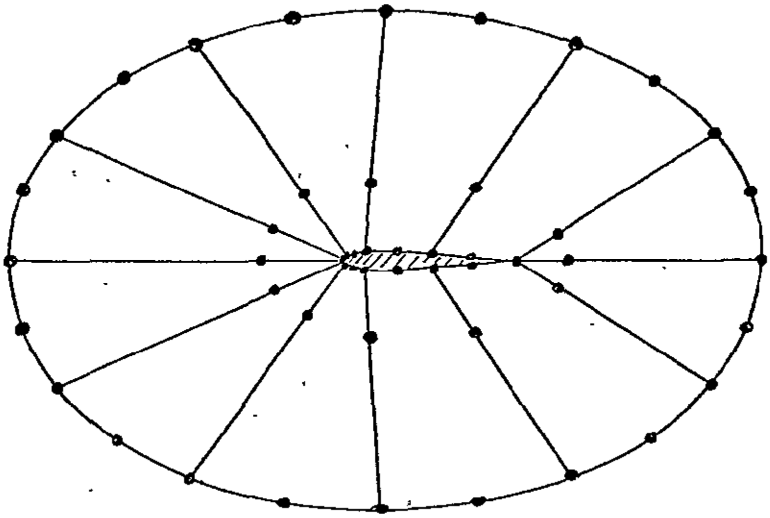


Figure 1. Macro-Element Discretization for Isolated Two-Dimensional Airfoil

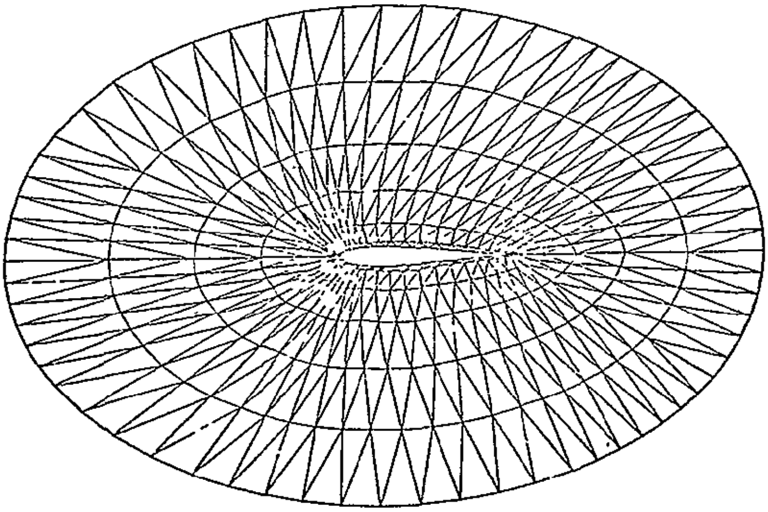
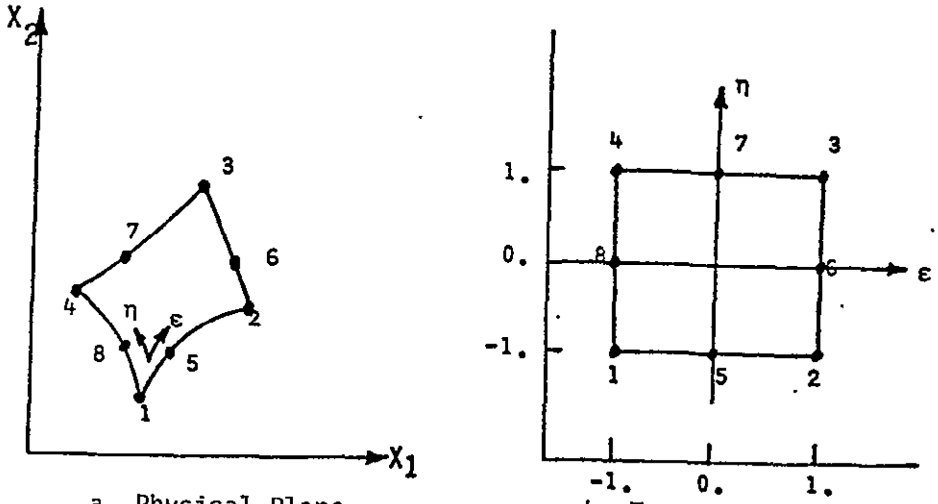


Figure 2. Finite Element Discretization for Isolated Two-Dimensional Airfoil



a. Physical Plane

b. Transformed Plane

Figure 3. Mapping a General Quadrilateral Onto Natural Coordinate Description

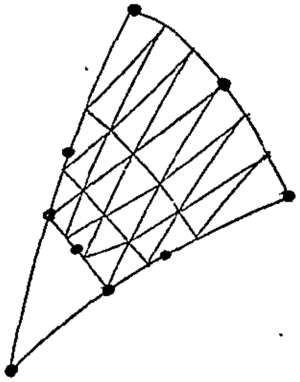


Figure 4. Telescoping Subdomain

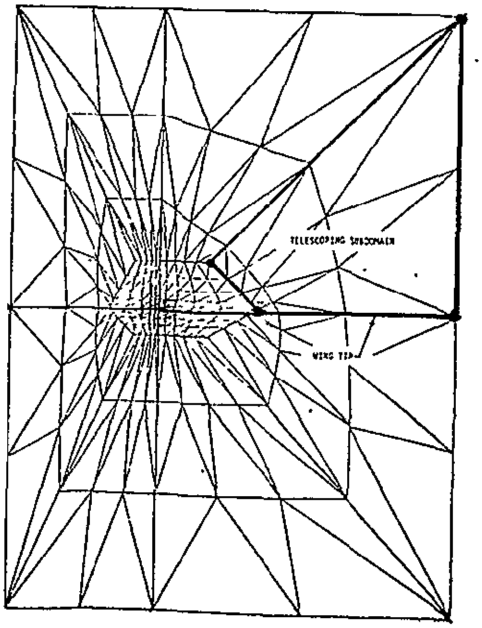


Figure 5. Wing Tip Discretization Using Telescoping Subdomain

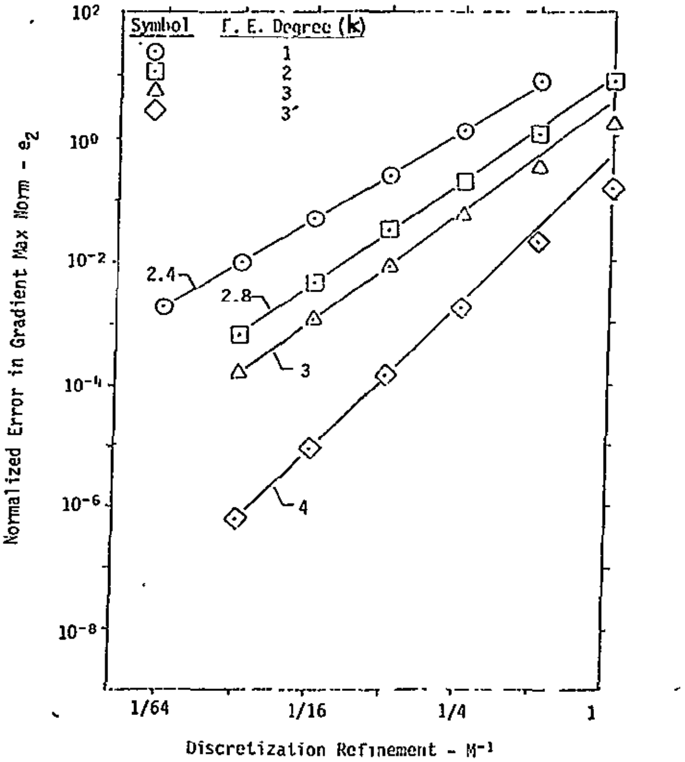
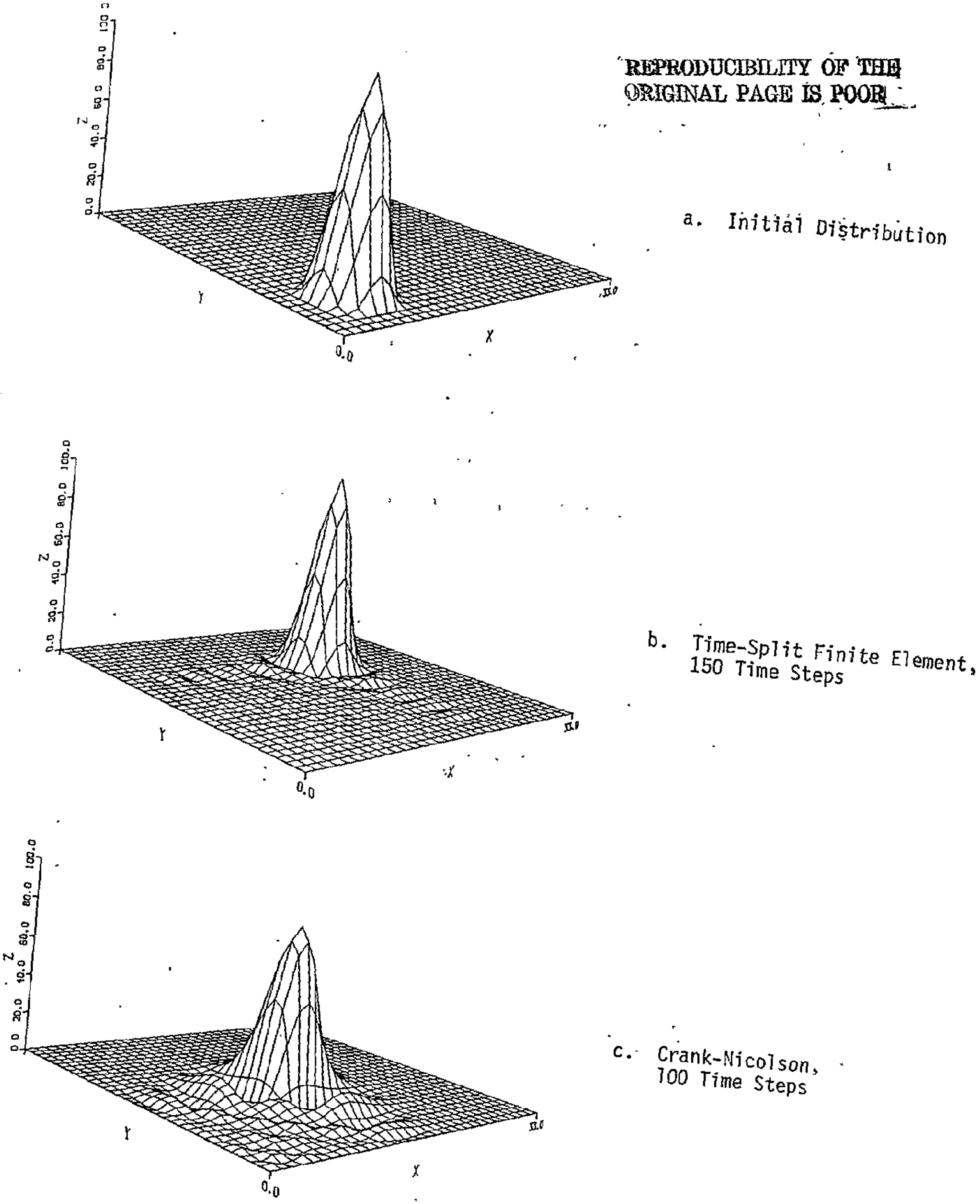


Figure 6. Solution Error for Parabolic Equation with Convection Boundary Condition



a. Initial Distribution

b. Time-Split Finite Element, 150 Time Steps

c. Crank-Nicolson, 100 Time Steps

Figure 7. Computed Density Distributions Using Time-Split Finite Element and Finite Difference Algorithms, C = 0.1

SOME MESH GENERATION REQUIREMENTS AND METHODS

Lawrence J. Dickson
Boeing Aerospace Co.
Seattle, Washington

I. The Generation Mapping.

Discretized solution algorithms, which find solutions of field equations in a (two or) three dimensional field, generally use meshes which are fitted to the field boundary to allow convenient formulation of boundary conditions there. For the purposes of the present discussion, a mesh is defined to be the image of a rectangular grid in computational space $\{(i,j,k)\}$ under a mesh mapping which maps computational space into physical space $\{(x,y,z)\}$. See Fig. 1.

It is not necessary that all of computational space be mapped onto the region of interest in physical space. Parts of it can be excised, as in Fig. 1, to give a better fit to the boundary. Many different excisions can be made to fit a single boundary; the choice depends on the mesh arrangement desired in the field. Up to now, very simple

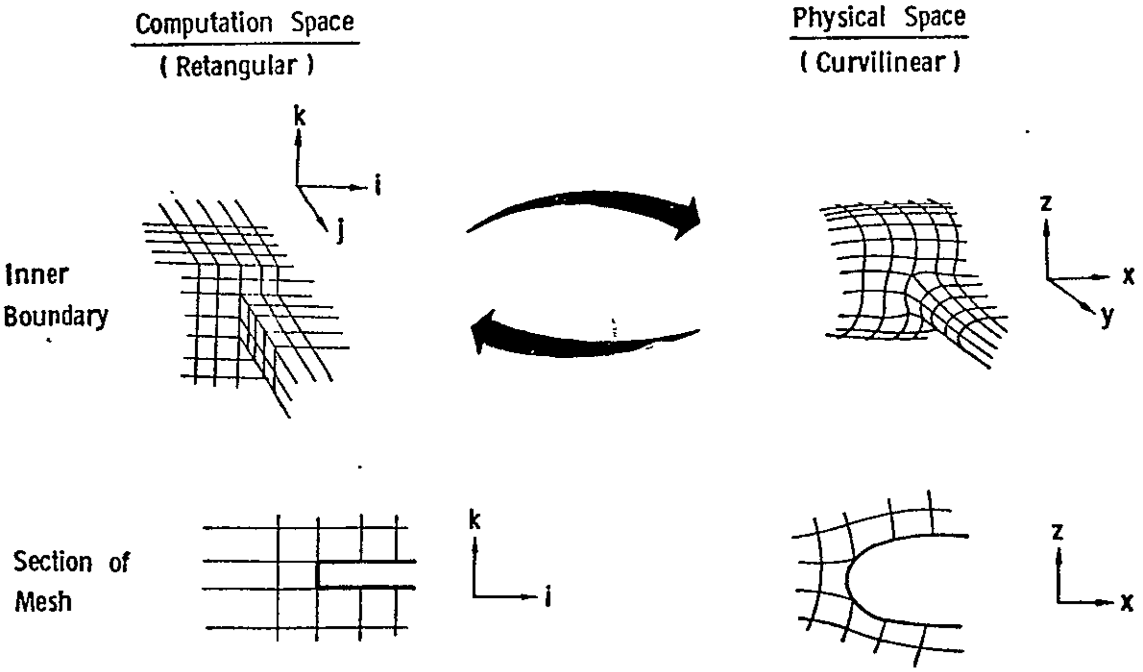


Fig. 1. The Mesh Mapping

excisions, such as half-spaces, have usually been used. They can be molded to surprisingly complex shapes, as in Art Rizzi's finite volume mesh for the space shuttle.

Four conditions are required of a "good" mesh mapping:

(1) Well-conditioning: It is always necessary to avoid "folds", or zero Jacobians.

(2) Smoothness: Most computational algorithms require a mesh that is smooth in the field, even if the boundary conditions are unsmooth.

(3) Desirable boundary behavior: The mesh must be such as to minimize discretization errors. For instance, it should be dense in regions where the solution function changes rapidly, and highly skewed meshes are usuallv undesirable.

(4) Generality: It should be possible to fit a grid around topologically complex boundary shapes or sets of shapes.

Fig. 2 exhibits good and bad meshes according to these criteria.

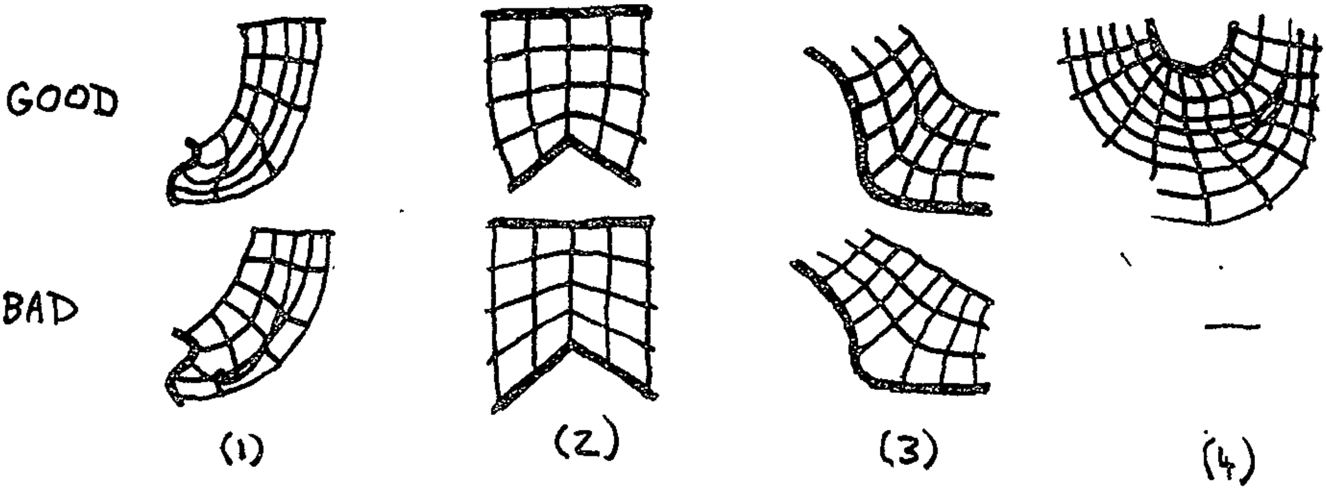


Fig. 2. Requirements for Mesh Mappings.

II. Candidate Methods

A. Blending functions. The simplest method is to impose the boundary conditions up to the order required on each rectangular network in computational space, and satisfy them using interpolating polynomials (Gordon's transfinite interpolation). For instance, if $\vec{x}$ and $\nabla\vec{x}$ are imposed at boundaries ($\vec{x}=(x,y,z)$), tricubics in (i,j,k) might be used.

This method gives very simple expressions for the mesh mapping and (by matching interior boundary conditions) can give smooth, general results, but in large or complex regions is in danger of being ill-conditioned.

B. Laplace solvers - univariate, (Rubbert et. al.). ,One computational variable is made to satisfy Laplace's equation in the physical variables, and the other variables are defined by making them constant on "streamlines" (Fig. 3,4). This robustly enforces well-conditioning and gives a smooth mesh, and can be adapted to topologically complex cases, but sometimes results in poor mesh densities in critical regions, and forces one to allow the "circumferential" computational variables to find their own values on one boundary.

C. Laplace solvers - trivariate (Thompson et. al.). All the computational variables are made to satisfy Laplace's equation in the physical variables (Fig. 3). This gives a smooth result which can be made to satisfy "Dirichlet" type conditions on all boundaries, yielding more boundary control than the

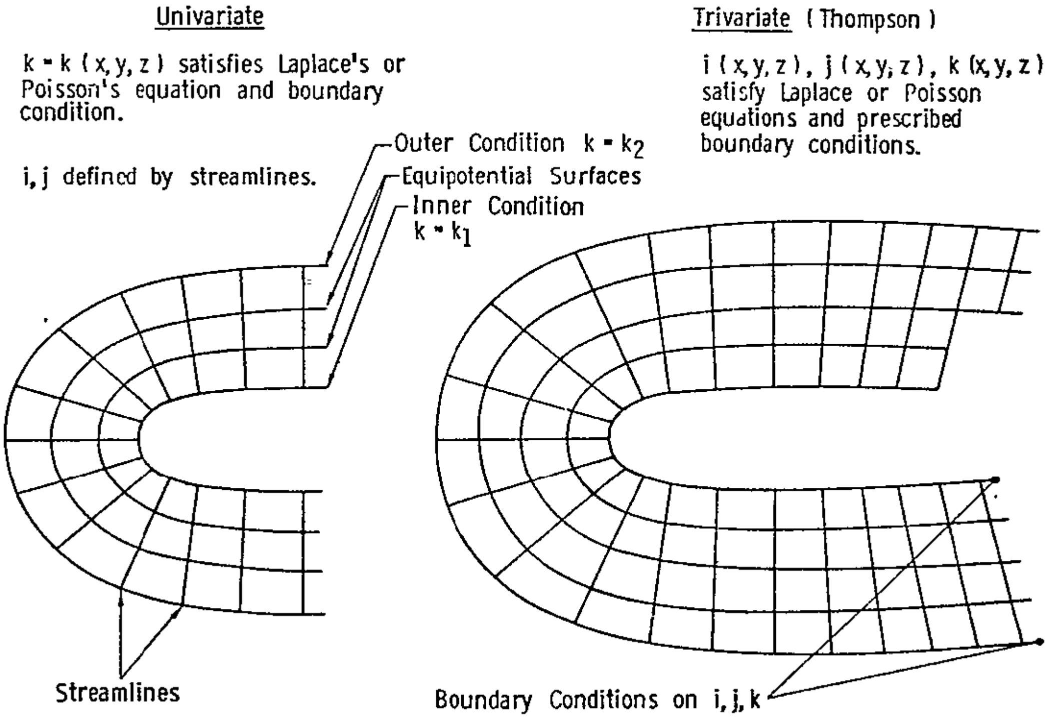

Univariate
k = k (x, y, z) satisfies Laplace's or Poisson's equation and boundary condition.
i, j defined by streamlines.
Outer Condition k = k2
Equipotential Surfaces
Inner Condition k = k1
Streamlines
Trivariate (Thompson)
i (x, y, z), j (x, y, z), k (x, y, z) satisfy Laplace or Poisson equations and prescribed boundary conditions.
Boundary Conditions on i, j, k

Fig. 3. Laplace Solving Methods.

univariate method. It still may fail to give desirable mesh densities (as at concave corners), and is in danger of ill-conditioning or at least skewness, if one gets too far away from the univariate solution.

D. Poisson or biharmonic equations. Introducing Poisson terms ("source clouds") in the above methods gives some control over mesh density in "bad" regions. A subset of this approach is to satisfy the biharmonic equation, $\nabla^4\vec{\xi} = 0$. That allows one to impose normal derivative boundary conditions ($\partial\vec{\xi}/\partial N = \vec{\mu}$). A normal derivative condition on the tangential computational variables controls skewness of the mesh; one on the normal computational variable controls the mesh density.

Again, boundary conditions that are too "wild" may lead to ill-conditioning of the mesh. Also, for the last three methods, an inverse Laplace, Poisson, or biharmonic equation must be solved to give the physical variables in terms of the computational. This can be done by finite difference, finite element, or singularity-panel methods; any of these methods can be expensive.

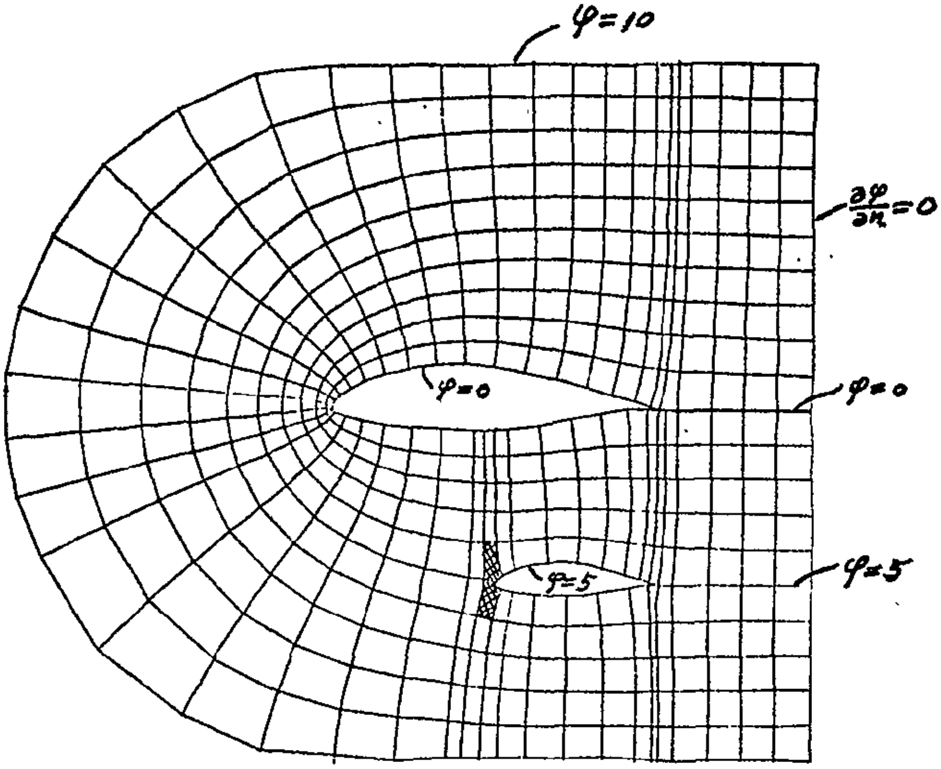


Fig. 4. Univariate 2D Grid.

The author suggests the approach of using B., C., and D. in sequence, setting the boundary conditions on each equal to those that hold on the solution to the previous method, except where alterations are necessary to achieve some desirable values. Such economy should give a good mesh at the boundary while preserving well-conditioning in the interior in most cases. Otherwise, patching with A. may be called for.

III. Conclusions.

Table 1 shows the author's estimate of the good and bad points about each method. Criteria not mentioned are "indifferently" satisfied.

The author believes that only a combination of most or all of the above grid generation methods will give assurance of good results for the arbitrary problem.

METHOD	GOOD	BAD
A. Blending Functions	2,3,4	1
Laplace's Equation —		
B. Univariate	1,2	3
C. Trivariate	2,4	
D. Biharmonic Equation or Poisson Equation	2,3,4	

(1) Well Conditioning

(2) Smoothness

(3) Desired Boundary Behavior

(4) Generality

Table 1. Advantages and Disadvantages.

SESSION 8

F. R. Bailey, Chairman

D25

SEL #113

N78-19803

INTERIM REPORT OF A STUDY OF A MULTIPIPE CRAY-1 FOR FLUID MECHANICS SIMULATION

D. A. Calahan
P. G. Buning
D. A. Orbits
W. G. Ames

7/1/77 - 9/30/77

Sponsored by Mathematical and Information
Sciences Directorate
Air Force Office of Scientific Research
under Grant 75-2812

Systems Engineering Laboratory
University of Michigan
Ann Arbor, Michigan 48109

ABSTRACT

This report documents the initial phase of a study of the performance of the CRAY-1 and its architectural extensions on 2-D and 3-D codes for the solution of the Navier-Stokes equations describing aerodynamic fluid flow. In this phase, a standard 2-D code has been benchmarked on the CRAY-1 and a preliminary version of a simulator of the CRAY-1 has been programmed.

ACKNOWLEDGEMENTS

We wish to acknowledge the computer access and general assistance provided by Cray Research, Inc., the technical support of the Air Force Flight Dynamics Laboratory, as well as the cooperation of Dr. Robert MacCormack of NASA/Ames in providing us with his 2-D explicit/implicit aerodynamic simulation code.

I. Introduction

In a recent study of the performance of high speed scalar and vector processors [1], it was found that the CRAY-1, programmed in assembly language, significantly outperformed other processors programmed in Fortran on simple matrix benchmarks. Although it was not clear what performance degradation would result from executing Fortran programs on the CRAY-1, it was felt from a cursory study of their architecture that assembly language coding on the other processors was not likely to speed up their performance significantly on vectorized algorithms.

In a series of scalar benchmarks [2], LASL demonstrated the CRAY-1 to be several times faster than the CDC 7600.

Based on these preliminary results, it appeared that the CRAY-1 performance should also be evaluated on fluid mechanics benchmarks currently being used to determine an architecture for high speed aerodynamic simulation in the early 1980's. It was clear, however, that even the 100 megaflop execution rate of the CRAY-1 was an order of magnitude too low for projected computational requirements. Therefore, it was proposed as part of this study to consider architecture/algorithm tradeoffs associated with a multipipe CRAY-1. Detailed modeling of the CRAY-1 was considered feasible due to the explicitness of its architecture, despite its variety of vector and scalar resources.

This report covers the initial three-month period of the nine-month study. Two major research topics are documented:

1. The conversion to the CRAY-1 of a state-of-the-art 2-D fluid mechanics code, a timing study of its performance, and projected speedups achievable by assembly language coding.
2. The initial development of a simulator of the present CRAY-1, for detailed algorithm performance evaluation.

II. Physical Problem and Method of Solution

The method employed by MacCormack [4] is intended to solve viscous-inviscid flow interaction at high Reynolds numbers. This involves equation splitting the Navier-Stokes equations and using a combination of explicit and implicit solution techniques.

The particular flow problem being solved with this code is supersonic flow over a flat plate with shock impingement. The grid is of constant increment in the x-direction and is exponentially stretched in the y-direction. A division of the grid is made, separating the region where viscous terms are small compared to inviscid terms, from the boundary layer region (see fig. 1).

The Navier-Stokes equations, in conservation form, can be written as:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0$$

Operators L_x and L_y are developed to advance the solution in time by accounting only for the effect of the x-derivative or y-derivative, respectively. The L_y operator, then, solves the equation:

$$\frac{\partial U}{\partial t} + \frac{\partial G}{\partial y} = 0$$

in predictor/corrector fashion, so that:

$$U_{i,j}^{(P)} = U_{i,j} - \frac{\Delta t}{\Delta y} (G_{i,j} - G_{i,j-1})$$

the predicted value, and the corrected value is:

$$U_{i,j}^{(C)} = \frac{1}{2} \left\{ U_{i,j} + U_{i,j}^{(P)} - \frac{\Delta t}{\Delta y} [G_{i,j+1}^{(P)} - G_{i,j}^{(P)}] \right\}$$

L_x is defined similarly.

This is an explicit scheme, relying only on previous values of U and G to solve for each new value of $U_{i,j}$. It adequately solves the flow field in the coarse mesh when the operators are employed symmetrically, as:

$$U_{i,j}^{n+1} = L_x(\frac{\Delta t}{2}) \ L_y(\Delta t) \ L_x(\frac{\Delta t}{2}) \ U_{i,j}^n$$

where we are calculating the flow at time t = (n + 1)Δt. MacCormack, in [5] used this method for the entire flow field.

For the boundary layer region, however, stability restrictions on the time step make this method inefficient. Two new operators are developed, replacing L_y with L_{y_H} and L_{y_P}. Here the differential equation in the y-direction is split again, and the L_{y_H} operator solves:

$$\frac{\partial U}{\partial t} + \frac{\partial G_H}{\partial y} = 0$$

where G_H are the inviscid (hyperbolic) terms of G and G_P (following) are the viscous, parabolic terms. L_{y_P} solves:

$$\frac{\partial U}{\partial t} + \frac{\partial G_P}{\partial y} = 0.$$

L_{y_H} uses a space averaged form of the method of characteristics to calculate convective velocity and pressure terms before calling the L_y operator for solution. L_{y_H} is thus explicit.

Solution of the parabolic terms is done implicitly, with tridiagonal matrices being generated for the u and v momentum equations. These are tridiagonal, i.e.,

$$U^{n+1}_{i,j} = U^{n+1}_{i,j} \left(U^{n+1}_{i,j-1}, U^{n+1}_{i,j+1}\right)$$

because only derivatives in the y-direction are used (due to equation splitting). One tridiagonal system is generated per variable per grid column. The energy equation in e, the total energy per unit volume, is not in model parabolic form and is split into the solution of three variables u^2, v^2, and ε (specific internal energy), where:

$$\frac{e}{\rho} = \varepsilon + [(u^2 + v^2)/2].$$

The coefficient matrices for u^2 and v^2 are the same as those of u and v, respectively, leading to the factorization of one system and forward and back substitution of three systems for the additional three variables.

In the fine mesh, it is found that the L_x operator is sufficient to evaluate terms in the x-direction, leading to the symmetric operator sequence of:

$$U_{i,j}^{n+1} = [L_{yH}(\frac{\Delta t}{2m}) \; L_{yP}(\frac{\Delta t}{2m}) \; L_x(\frac{\Delta t}{m}) \; L_{yP}(\frac{\Delta t}{2m}) \; L_{yH}(\frac{\Delta t}{2m})]^m \; U_{i,j}^n$$

The time step here is smaller than that of the coarse mesh, so the operator sequence is executed m times, where m is a small integer, usually 2.

The program flow chart (fig. 2) reflects each of these features and shows the computation involved per time step.

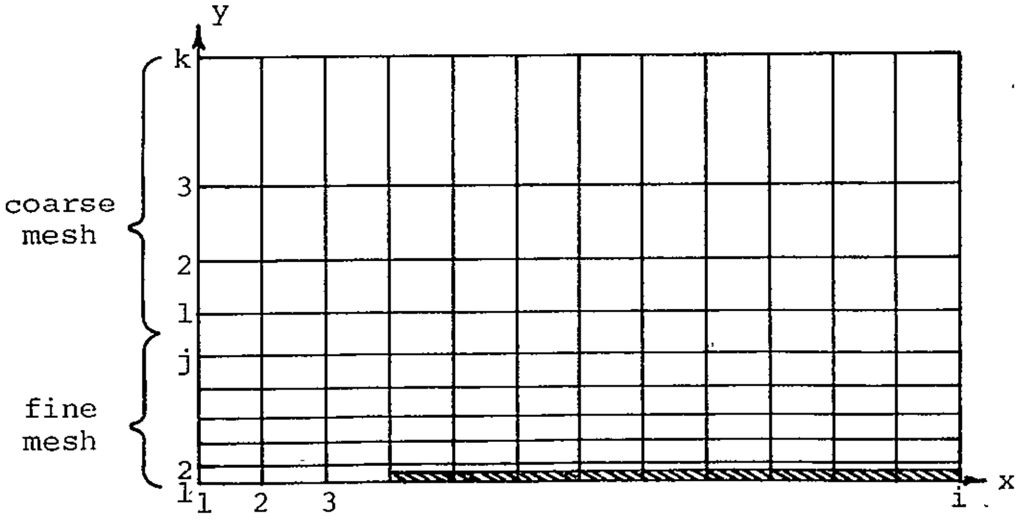
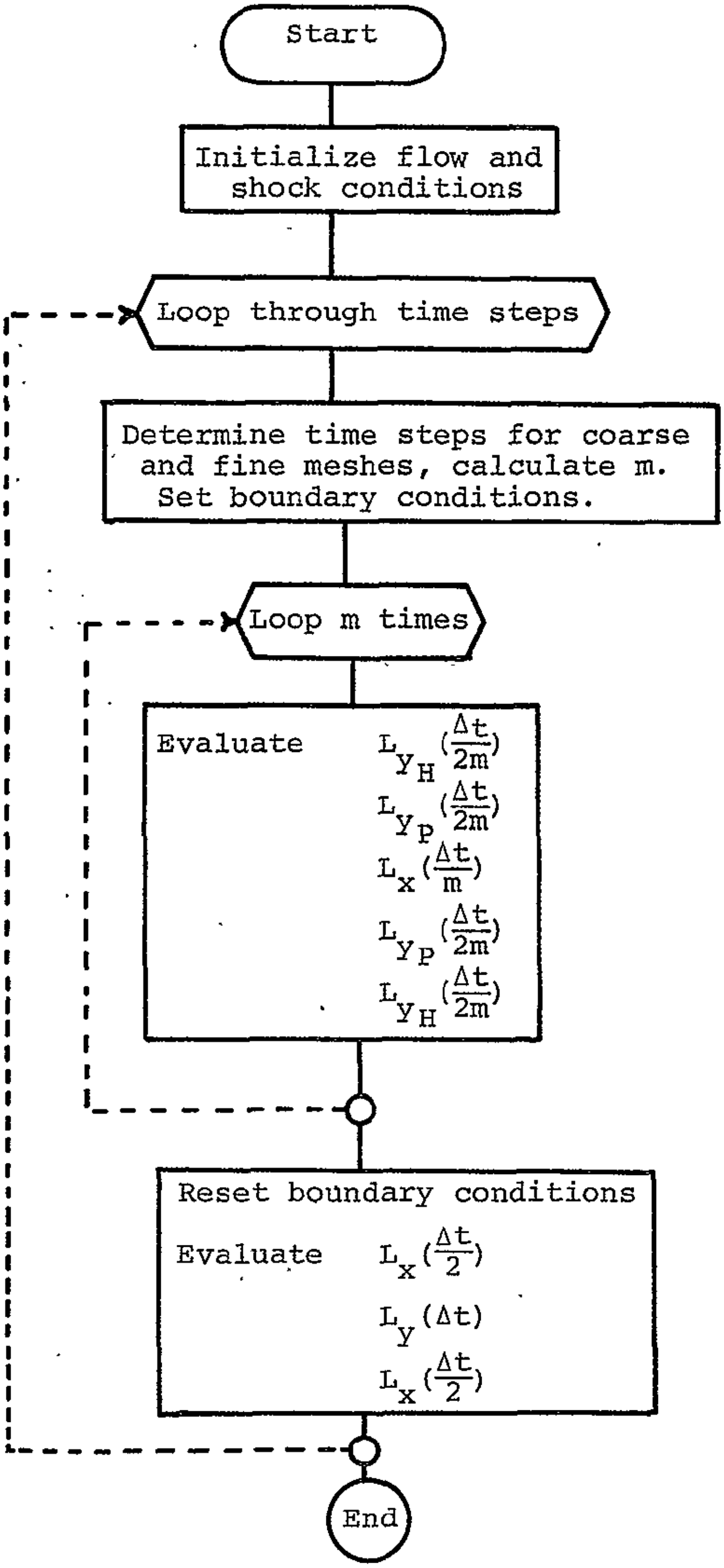


Figure 1. Computational mesh

Figure 2. PROGRAM FLOW CHART


Start
Initialize flow and shock conditions
Loop through time steps
Determine time steps for coarse and fine meshes, calculate m. Set boundary conditions.
Loop m times
Evaluate $L_{y_H}(\frac{\Delta t}{2m})$
$L_{y_P}(\frac{\Delta t}{2m})$
$L_x(\frac{\Delta t}{m})$
$L_{y_P}(\frac{\Delta t}{2m})$
$L_{y_H}(\frac{\Delta t}{2m})$
Reset boundary conditions
Evaluate $L_x(\frac{\Delta t}{2})$
$L_y(\Delta t)$
$L_x(\frac{\Delta t}{2})$
End

III. Conversion of MacCormack's 2-D code to the CRAY-1

Table 1 is a summary of results obtained from runs on the CRAY-1 computer, in comparison to the Air Force Flight Dynamics Lab CDC 6600. The original code, as received from NASA Ames was run on the CDC 6600 using level 2 optimization. The same code was then run on the CRAY, with results presented in column two. The CRAY FORTRAN compiler automatically recognizes vectorizable loops and generates vector machine instructions accordingly. Thus a few loops in the program were vectorized. All variables were 64 bits (CRAY single precision).

In the main computational sections of code, million floating point operations per second (MFLOPS) rates of between 2 and 5 were achieved. This speedup of a factor of 10 over the CDC 6600 basically represents the scalar performance of the CRAY. By reorganizing the code, most of the floating point computations could be included in vectorizable loops, leading to a further speedup of a factor of 4 in the explicit solution codes representing the L_x and L_y operators. With assembly language coding, all floating point operations could be vectorized, leading to rates of possibly 70 MFLOPS. The routines based on the method of characteristics calculation of convective velocity and pressure terms for the L_{y_H} operator present major problems to vectorization, since the process relies heavily on logical decisions to arrive at numeric results. This section can be vectorized, then, only with extensive assembly language coding or a change in the algorithm.

Table 1. COMPARATIVE TIMING AND MFLOP RATES (32 x 32 physical grid)

Execution times represent one time step in solution

	CDC 6600 Optimization level 2 Original Code	CRAY-1 CRAY FORTRAN Original code	CRAY-1 CRAY FORTRAN Vectorized code	Projected CRAY-1 Assembly Lang. Maximum theoretical speeds 32 x 32 grid	 66 x 130 grid
Explicit solution, x-direction	0.6287 sec 0.5675 MFLOPS	0.08649 sec (7.3x) 4.125 MFLOPS	0.02347 sec (27x) 15.20 MFLOPS	127x 72 MFLOPS	180x 102 MFLOPS
Explicit solution, y-direction	0.6176 sec 0.3303 MFLOPS	0.07920 sec (7.8x) 2.576 MFLOPS	0.01818 sec (34x) 11.22 MFLOPS	269x 89 MFLOPS	309x 102 MFLOPS
Method of characteristics (Hyperbolic)-calculation of convective velocity and pressure terms.	1.602 sec *	0.1159 sec (13.8x) *	Not vectorized***	Not vectorized	Not vectorized
Implicit (Parabolic) solution: Equation formulation	1.695 sec 0.4671 MFLOPS	0.1728 sec (9.8x) 4.582 MFLOPS	0.02500 sec (68x) 31.67 MFLOPS	199x† 93 MFLOPS	227x† 106 MFLOPS
Tridiagonal solution	0.7570 sec 0.2172 MFLOPS	0.05957 sec (12.7x) 2.760 MFLOPS	0.003530 sec **(214x) 38.28 MFLOPS	207x 45 MFLOPS	240x 52 MFLOPS
Total execution time, one time step.	6.222 sec	0.6321 sec (9.8x)	0.3180 sec (19.6x)		

†Kernel estimate only

*No accurate count of floating point operations available at this time.

**Tridiagonal code written in CRAY Assembly Language by Mr. Tom Jordan, Los Alamos Scientific Laboratory.

***Not extendible to 3-D.

The L_{y_p} operator, solving the parabolic equations, is an implicit scheme, involving the solution of tridiagonal matrices. This routine received the most attention in the vectorization benchmarks, and a short description and tabular breakdown of vector operations are included here (see Table 2).

Formulation of tridiagonal matrices in five variables, u, v, u^2, v^2, and ε is written in Fortran, and all numeric calculations have been arranged into vectorizable loops. This allows similar numeric operations to be performed in rapid succession since the structure of the numeric expression need not be examined for each array element. The tridiagonal solver, written by Mr. Tom Jordan of Los Alamos Scientific Laboratory, is coded in assembly language and is set up to solve up to 64 tridiagonal systems simultaneously. The current grid solved by the implicit routine is 32 points in the x-direction and 16 in the y-direction. In the table, then, let $i = 32$, $j = 16$. Fortran sections of this code are achieving 37.1 MFLOPS, close to the average tridiagonal assembly code rate of 38.3. Also note that only 12 percent of the execution time is spent in the tridiagonal routines, the equation formulation (in Fortran) accounting for the other 88 percent of the implicit solution.

The method of characteristics and many auxiliary routines (boundary value and time step calculations) have not been vectorized. The resulting final speedup of the program is only 19.6, so that a complete solution to a 256-step problem would take approximately 80 seconds.

Table 2. BREAKDOWN OF VECTOR OPERATIONS IN IMPLICIT SOLUTION

	in words:
Main formulation loop - set arrays for grid points. Average vector length = (i - 2). i = 32, j = 16	vector loads = 69ij - 67i - 13j + 134 vector stores = 32ij - 30i - 64j + 60 vector FLOPS = 150ij - 148i - 300j + 296 1.822 x 10^{-3} CPU secs 37.1 MFLOPS
Set boundary conditions, final matrix coefficients and right hand sides. Average vector length = (i - 2). i = 32, j = 16	vector loads = 38ij + 12i - 76j - 33 vector stores = 17ij + 6i - 34j - 18 vector FLOPS = 62ij + 9i - 124j - 27 1.362 x 10^{-3} CPU secs 32.5 MFLOPS
Tridiagonal factorization, n systems Vector length = n	vector loads = n(ej - 2) vector stores = n(2j - 1) vector FLOPS = n(4j - 3)
n = 60 = 2(i-2)	.09585 x 10^{-3} CPU secs 38.2 MFLOPS
n = 30 = (i - 2)	.06055 x 10^{-3} CPU secs 30.2 MFLOPS
Tridiagonal forward and back substitution Vector length = n	vector loads = n(5j - 2) vector stores = n(2j - 1) vector FLOPS = n(5j - 4)
n = 60 = 2(i - 2)	.1086 x 10^{-3} CPU secs 42.0 MFLOPS
n = 30 = (i - 2)	.06766 x 10^{-3} CPU secs 33.7 MFLOPS
Total implicit routine, scalar and vector operations	total loads = 282ij - 112 i - 564j + 871 total stores = 130ij - 26i - 280j + 76 total FLOPS = 498ij - 238 i - 996 j + 654 7.469 x 10^{-3} CPU secs 31.1 MFLOPS

IV. Projected Performance Using Assembly Language Coding

A. Introduction

Because the CRAY-1 can achieve sizable speedups by assembly coding (which was in general not possible because of limited machine access), we project in the following sections the times of most critical phases of the solution process. These projections are sometimes based on attributes of a modified CRAY-1 (e.g., more vector registers), but otherwise represent what we believe are achievable performances based on our detailed knowledge of machine architecture.

Two parts of the code were not studied: (1) the characteristic method phase, which will be abandoned in 3-D simulations, and (2) boundary-condition-related calculations, which, as shown in [3], can become a significant component of a vector solution process. We declined to study the vectorization of this phase because of machine access limitations and because the version of MacCormack's code we used required simplified (unrealistic) boundaries.

B. Explicit Solution

1. x-Direction

In the MacCormack code, explicit solution in the x-direction occurs separately for the upper and lower parts of the grid. If the dimensions of either section are i points in the x-direction and k points in the y-direction, we can state the number of vector operations in the main section of code in terms of the problem size:

$$\begin{array}{rl} \#\text{ vector loads} = 2(i - 2)25 & \\ \text{stores} = 2(i - 2)12 & \\ \left.\begin{array}{r} \text{FLOPS} = 2(i - 2)71 \\ \text{multiplies} = 2(i - 2)49 \end{array}\right\} & \text{not including SQRT calls} \quad (1) \end{array}$$

For this x-direction solution, the vector length is $(k - 1)$. Note that for both cases in this code $i = 32$, $k = 16$.

We now assume that there is no limit on the number of available vector registers (this means we probably need about 16), and that all operations are overlapped with vector multiplies. This means we have $2(i - 2)49$ non-concurrent vector operations.* We further assume an average startup of 9 clock periods for each of these operations. We now have the number of floating point operations and required clock periods:

$$\#\text{ floating point operations} = 2(i - 2)71(k - 1)$$

$$\#\text{ clock periods} = 2(i - 2)49(k - 1 + 9)$$

The ratio of floating point operations to clock periods, divided by the time per clock period will give an expression for the MFLOP rate theoretically possible, assuming no overhead for the $2(i - 2)$ times through the outer loop, and no overhead for the possibility of vectors being longer than 64:

*Based on a vector length of 64 or less.

$$\text{MFLOPS} = \left[\frac{2(i-2)71}{2(i-2)49}\right]\left(\frac{k-1}{k-1+9}\right)\left(\frac{1}{12.5 \times 10^{-9}}\right)\left(\frac{\text{MFLOP}}{10^6 \ \text{FLOP}}\right)$$

$$= 80 \left(\frac{142}{98}\right)\left(\frac{k-1}{k-1+9}\right)$$

$$= 72.4, \text{ for } k = 16$$

$$\rightarrow 102 \text{ as } k - 1 \rightarrow 64$$

2. y-Direction

Explicit solution in the y-direction takes place almost identically, though vectorization is along rows of grid points rather than columns. Equations (1) are rewritten, then, as:

$$\begin{aligned} \text{\# vector loads} &= 2(k-1)25 \\ \text{stores} &= 2(k-1)12 \\ \text{FLOPS} &= 2(k-1)71 \\ \text{multiplies} &= 2(k-1)49 \end{aligned} \left.\vphantom{\begin{matrix} a \\ b \end{matrix}}\right\} \text{ not including SQRT calls}$$

Similarly, the vector length will now be (i - 2), and under the same assumptions as made previously, we derive the MFLOPS expression as:

$$\text{MFLOPS} = 80 \left(\frac{142}{98}\right)\left(\frac{i-2}{i-2+9}\right)$$

$$= 89.2, \text{ for } i = 32$$

$$\rightarrow 102 \text{ as } i - 2 \rightarrow 64$$

Note that without overhead, the MFLOP rate here is independent of the number of rows of grid points, as the x-direction explicit rate was independent of the number of columns.

C. Implicit

1. Main Formulation Loop (Restructured)

Grid points near the plate are solved in the y-direction by the explicit method for hyperbolic terms, added convective velocity and pressure terms being calculated by space averaged method of characteristics. Use of this method, though, will be discontinued. An implicit scheme is used for parabolic terms and includes the formulation and solution of tridiagonal systems. The main loop in formulation of the matrix coefficients and right hand sides yields the following breakdown of vector operations:

$$\begin{array}{rl} \text{\# vector loads} = 2(j - 1)30 & \\ \text{stores} = 2(j - 1)22 & \\ \left.\begin{array}{r}\text{FLOPS} = 2(j - 1)116 \\ \text{multiplies} = 2(j - 1)77\end{array}\right\} & \text{not including SQRT calls}\end{array}$$

As in the explicit code, we see that all memory references can be overlapped with multiplies. Vectorization here is along all points in a row, yielding a vector length of (i - 2). For this part of the grid, there are i points along the plate as before, and j points vertically. Here j = 16.

Again we assume no restriction on the number of vector registers and an average startup of 9 clocks,and arrive at the following expressions for floating point operations and number of clock periods:

$$\# \text{ floating point operations} = 2(j - 1)116(i - 2)$$

$$\# \text{ clock periods} = 2(j - 1)77(i - 2 + 9)$$

The number of MFLOPS achievable is then expressed as:

$$\text{MFLOPS} = 80 \left(\frac{116}{77}\right)\left(\frac{i - 2}{i - 2 + 9}\right) .$$

$$= 92.7 \text{ for } i = 32$$

$$\rightarrow 106 \text{ as } i - 2 \rightarrow 64$$

2. Tridiagonal

Solution of the tridiagonal systems is currently performed by CRAY Assembly Language routines, and run at 30 to 40 MFLOPS. We shall compare these with theoretically achievable rates with no overhead. In the LU decomposition routine TRIDEC, there are the following number of non-concurrent vector operations:

$$\#n - c \text{ vector operations} = 4 + 6(j - 1)^*$$

If we are solving n simultaneous tridiagonal systems, the length of all vector operations is n. In the current implicit scheme, TRIDEC is called once with 2(i – 2) systems, and once with (i – 2) systems. This is done only to keep the total number of systems less than or equal to 64 as required by the assembly language routines. Obviously this strategy would have to change for different problem sizes, and this does not

*This has now been reduced to 4 + 5(j – 1).

appear to be a problem. For now we will use n as the vector length to derive general expressions. The floating point and clock period counts are as follows:

$$\# \text{ floating point operations} = [j + 3(j - 1)]n$$

$$\# \text{ clock periods} = [4 + 6(j - 1)](n + 9)$$

The MFLOP rate is then:

$$\begin{aligned} \text{MFLOPS} &= 80 \left(\frac{4j - 3}{6j - 2}\right)\left(\frac{n}{n + 9}\right) \\ &= 45.1 \text{ for } j = 16,\ n = 60 \\ &= 39.9 \text{ for } j = 16,\ n = 30 \\ &\rightarrow 46.8 \text{ as } j \rightarrow \infty,\ n = 64 \end{aligned}$$

For the forward and back substitution routine TRISLV we have:

$$\# \text{ floating point operations} = [j + 4(j - 1)]n$$

$$\# \text{ clock periods} = [4 + 7(j - 1)](n + 9)$$

With this scheme, TRISLV is called twice with 2(i - 2) systems and once with (i - 2) systems. The number of MFLOPS achievable is:

$$\begin{aligned} \text{MFLOPS} &= 80 \left(\frac{5j - 4}{7j - 3}\right)\left(\frac{n}{n + 9}\right) \\ &= 48.5 \text{ for } j = 16,\ n = 60 \\ &= 42.9 \text{ for } j = 16,\ n = 30 \\ &\rightarrow 50.1 \text{ as } j \rightarrow \infty,\ n = 64 \end{aligned}$$

These theoretical rates are graphed in fig. 3.

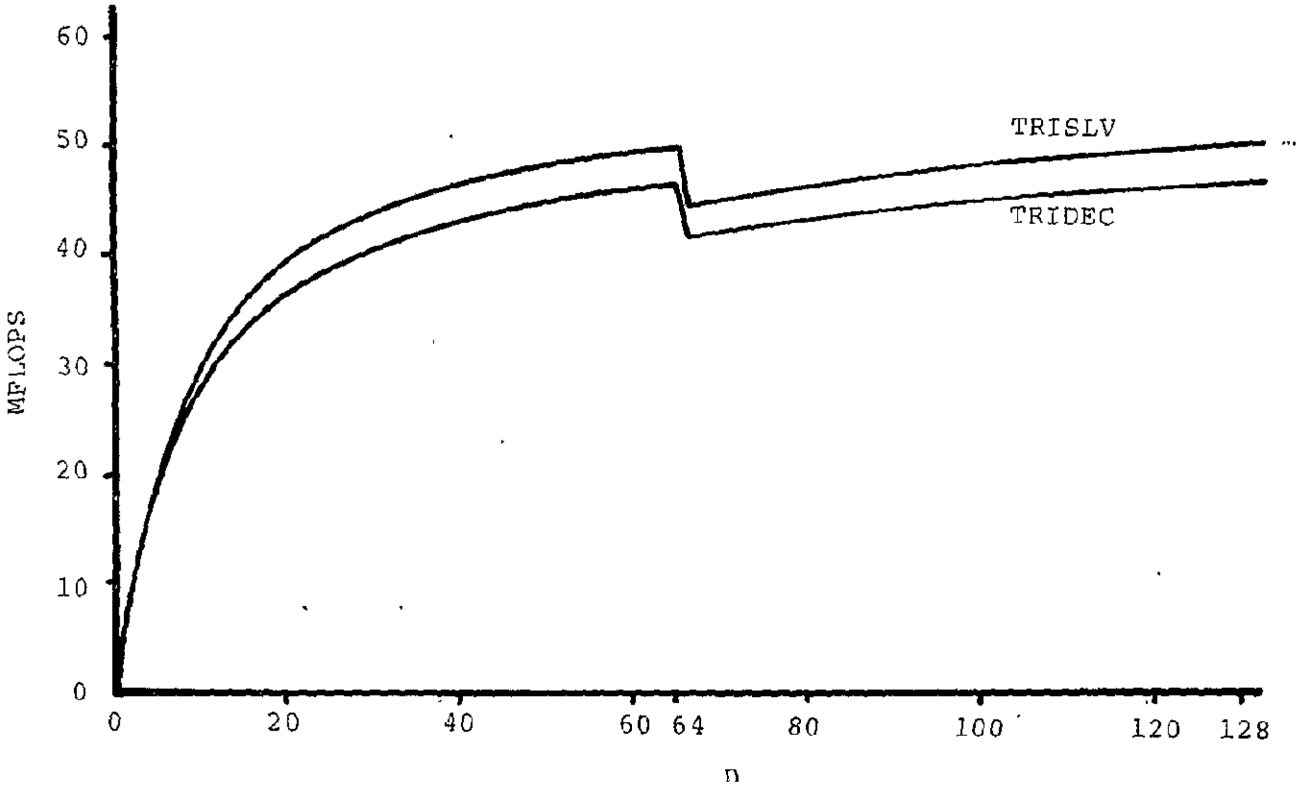


Figure 3. MFLOPS vs. Number of Simultaneous Systems. Theoretical rates, j → ∞.

D. Conclusions

Assembly language coding, then, can achieve rates of up to 100 MFLOPS on computational sections other than the tridiagonal, and possibly 50 MFLOPS on them. Assumptions made in arriving at these figures are as follows:

1. Problems will be large enough to guarantee vector lengths of 64.
2. We are modelling computational kernels only, ignoring loop setup, addressing calculations, and branching, some of which will be overlapped with vector computations.
3. There is no limit to the number of available vector registers.
4. There is an average startup for sets of chained or overlapped vector operations of 9 clock periods.

Some of these assumptions will have a definite degrading factor, reducing MFLOP rates by 20 to 30 percent, decreasing with increased problem size. Accuracy of these modelling concepts, then, must await actual coding of these routines.

V. CRAY-1 Simulator

A. Introduction

As a tool to analyze the behavior of kernels on the CRAY-1 computer, a timing simulator for the CRAY-1 was constructed. Initially a clock period level report is generated which indicates the state of the machine as it issues instructions. Various resources are flagged as busy for time appropriate to the issuing instruction. When an instruction can't issue due to a resource conflict, the conflict is highlighted in the report.

B. Method of Simulation

The simulator is written in IBM Fortran-IV and was developed at the University of Michigan on an AMDAHL 470 V/6. The timing for the simulator is event driven from a timing queue. Consequently, a trace of machine activity is possible at the clock period level. The state of resources can be observed and conflicts for resources analyzed.

The results of instruction execution are computed immediately when the instruction issues. As a consequence, data flow through the CRAY-1 and bit level arithmetic is not simulated. Instead all results are computed using the host machine's data formats. For the AMDAHL 470 V/6 this means:

1) Floating point arithmetic is done in IBM double precision.

2) Address arithmetic is done in 32 bits rather than in CRAY-1 24 bits.

Features of the CRAY-1 not simulated are:

1) Channel I/O

2) Interrupts

3) Exchange sequences

C. Current Progress

The simulator was designed to allow interactive debugging of the test programs. To this end an extensive repertoire of interactive commands are available to allow user control over the simulated program. These commands include stepping through the program, displaying or modifying registers or memory and setting break points in the simulated machine.

Currently about 2/3 of the instruction set is implemented consisting of the most common instructions encountered. Of these not all have been exercised heavily.

The only form of output is a detailed clock period record of the machine state. A portion of this appears on page 319. This report is divided into the following significant fields:

1) ST. - The machine state. (IS= issue, blank=hold issue)

2) TAG - A tag is assigned to each instruction to allow tracing its activity later in time.

3) Instr. - CRAY-1 instruction being issued.

4) P - The address of the instruction being issued.

5) CP - The machine clock period

6) +*/&>+ - The CRAY-1 Vector functional units floating point add, multiply, recip., logical, shift, integer add.

7) V. Reg. - The Vector registers

8) A. Reg. - The Address registers

9) S. Reg. - The Scalar registers

10) BCG - Parcel buffer change flag (used on a branch)

11) BSF - Block sequence flag (used on vector memory references)

12) BTX - B and T register block transfer flag

As the simulation proceeds, a tag is assigned to the issuing instruction. This tag is then used later in the report to indicate the instruction's use of a machine resource (Vector register, functional unit, etc.).

When a subsequent instruction tries to issue but finds one or more of its needed resources busy, the tag on the resource in conflict is underscored. In this way the instruction using the resource can be identified and the resource conflicts for the currently waiting instruction can be seen. This information can be useful for reordering instructions to minimize conflict.

```
     TRIP                                     U OF M   C R A Y - 1   S I M U L A T O R         12:37   SUN OCT 02/77    PAGE  9
    T                                     FPF      V. REG    S R R R   MEMORY BANKS   A  A. REG  S  S. REG   V AS          S011BFFBB
ST. A     INSTRUCTION      P-ADDR    CP   PPPVVV             C K K K                  R          R           M 00          T047CPLST
    G                                     +*/E>+  01234567   1 A B C 0123456789ABCDEF A 01234567 A 01234567    BB          H3X5GACPX

                                   000380| C     |  C CC    |                        | |       | |         | |           |
                                   000381| C     |  C CC    |                        | |       | |         | |           |
                                   000382| C     |  C CC    |                        | |       | |         | |           |
 IS E   ,A0,A3     V4     0000035A 000383| C     |    EC    |                        | |       | |         | |           |        E
 IS F   A4     A4+A5      0000035B 000384| C     |    EC    |                        | |    F  | |         | |           |        E
                                   000385| C     |    EC    |                        |F|    F  | |         | |           |        E
                                   000386| C     |    EC    |                        | |       | |         | |           |        E
                                   000387|       |    EC    |                        | |       | |         | |           |        E
                                   000388|       |    EC    |                        | |       | |         | |           |        E
                                   000389|       |    EC    |                        | |       | |         | |           |        E
                                   000390|       |    EC    |                        | |       | |         | |           |        E
                                   000391|       |    EC    |                        | |       | |         | |           |        E
 IS G   V0     V6*PV5     0000035C 000392| G     |G   EGG   |                        | |       | |         | |           |        E
 IS H   S0     #>77       0000035D 000393| G     |G   EGG   |                        | |       |H|         | | H         |        E
 IS I   A0     A1-A0      0000036A 000394| G     |G   EGG   |                        | |I      | |         | |IH         |        E
 IS J   A1     A1-A0      0000036B 000395| G     |G   EGG   |                        |I|IJ     | |         | |IH         |        E
                                   000396| G     |G   EGG   |                        |J| J     | |         | |I          |        E
                                   000397| G     |G   EGG   |                        | |       | |         | |I          |        E
                                   000398| G     |G   EGG   |                        | |       | |         | |           |        E
                                   000399| G     |G   EGG   |                        | |       | |         | |           |        E
                                   000400| G     |G   EGG   |                        | |       | |         | |           |        E
 IS K   V2     V7-PV0     0000036C 000401|KG     |K K EGGK  |                        | |       | |         | |           |        E
 IS L   JAN     0000030B  0000036D 000402|KG     |K K EGGK  |                        | |       | |         | |           |    L   E
                                   000403|KG     |K K EGGK  |                        | |       | |         | |           |    L   E
                                   000404|KG     |K K EGGK  |                        | |       | |         | |           |    L   E
 IS     <BLANK>                    000405|KG     |K K EGGK  |                        | |       | |         | |           |        E
 IS     <BLANK>                    000406|KG     |K K EGGK  |                        | |       | |         | |           |        E
 IS M   A0     A0+A6      0000030B 000407|KG     |K K EGGK  |                        | |M      | |         | |M          |        E
                                   000408|KG     |K K EGGK  |                        |M|M      | |         | |M          |        E
 IS N   V1     /HV2       0000030C 000409|KGN    |KNN EGGK  |                        | |       | |         | |M          |        E
                                   000410|KGN    |KNN EGGK  |                        | |       | |         | |M          |        E
                                   000411|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000412|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000413|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000414|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000415|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000416|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000417|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000418|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000419|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000420|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000421|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000422|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000423|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000424|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000425|KGN    |K*N EGGK  |                        | |       | |         | |           |        E
                                   000426|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000427|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000428|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000429|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000430|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
                                   000431|KGN    |KNN EGGK  |                        | |       | |         | |           |        E
```

D. Testing

At present we have timed the CRAY-1 on 30 small test code segments some of which have been run on the simulator. The timing agreement has been exact for the segments tested.

We have also run a tridiagonal equation solver on both the simulator and the CRAY-1. The following table shows the results of this timing with the times in clock periods.

Number of Equations	CRAY-1 Timing	Simulator Timing	Timing Error
4	1831	1844	.71%
10	4561	4591	.66%
20	9111	9172	.67%

In each case twenty systems were solved in parallel.

We consider this a fairly small error in light of the timing complexity of the CRAY-1.

We also modified the tridiagonal solver to further optimize it and achieved a 15 percent performance improvement. This has not been validated on the CRAY-1.

E. Future Plans

Currently the only reporting available from the simulator is a clock period report. We plan to extend the reporting to provide a more digestible summary of activity. This would include:

1) Percent functional unit utilization
2) Operation counts
3) FLOP rates
4) Percent memory utilization (scalar and vector)
5) Instruction hold issue conflict analysis

We also hope to extend the simulator to support a modified architecture.

To make the simulator more useful for large codes we plan to allow using it as subroutine from the large code. This would allow timing of certain segments closely while using the host machine to execute the bulk of the code.

We hope to have a cross assembler to allow the programming of larger codes. We currently assemble by hand which is effective only for small codes (less than 100 instructions).

F. Conclusion

Our current progress has demonstrated the feasibility of building a simulator to make reliable measurements of algorithm performance. Architectural extensions to the simulator could produce meaningful information regarding projected performance of algorithms on the modified architecture.

VI. Conclusions

A. A Multipipe CRAY-1

Programming the 2-D code on the CRAY-1 has exposed a number of issues which would concern both a re-architecture of the machine for fluid mechanics simulation and the use of such a machine from a higher level language. Since it is unlikely that a multipipe CRAY-1 will be built for only this application, these issues can be expected to influence a new design, but certainly not determine its major architectural features.

B. Algorithm/Architecture Issues

Vector length

Although a vector processor such as the CDC STAR 100 favors as long vectors as possible, there may be advantage for the CRAY-1 to segment the problem so as to operate with 64-length vectors which can reside in cache [6]. Our present version of the code vectorizes in only one direction, in contrast to [3]; this favors irregular boundary conditions in the direction of vectorization.

An n-pipe extension of the CRAY-1 would similarly favor 64n-length vectors, so that for n chosen large to achieve a gigaflop, it would be questionable whether at least partial vectorization in a second dimension would be advantageous.

Cache size

In vectorizing the original 2-D code of MacCormack, we maintained the separation of the equation formulation and solution steps, returning the equations to main memory from cache after formulation, and retrieving them for solution. This was necessitated by the small vector register cache in the CRAY-1. A larger (cache size)/(no. of processors) ratio in an n-pipe version would allow local equation formulation and solution within cache, reducing the main memory traffic.

Computational Imbalance

The principal reason for not projecting during equation formulation a megaflop rate closer to the 140 maximum (Table 1) is the preponderance of one type of arithmetic operation, so that not all arithmetic units can be busied. (Perhaps it is surprising that neither vector length nor cache size appears to be the limiting factor.) Since this is a global characteristic, it is doubtful that rearrangement of the computation would yield a higher execution rate.

Gather/Scatter Operations

We anticipate the necessity of using either short vector or gather/scatter operations in handling irregular boundaries. The CRAY-1 does not gather/scatter to main memory, but does allow masked operations between vector registers. If the available operations cannot efficiently handle the boundary condition problem, and if this segment of the code seriously impacts the total solution time, then one would have to consider installation of gather/scatter instructions to main memory in a multipipe CRAY-1 intended to solve 2-D and 3-D problems.

C. Software Issues

The 2:1 to 5:1 speedups achievable by use of assembly coding in the CRAY-1 are representative of results we have observed in other applications.* The lower ratio applies to largely scalar codes or codes irretrievably bound by main memory traffic (e.g., extensive indirect addressing); the larger ratio is representative of many linear algebra and other codes that can be highly vectorized and tuned to the CRAY-1. It is our feeling that a speedup of 2:1 to 3:1 can be virtually guaranteed for 2-D and 3-D codes.

From these observations, we conclude that to achieve high execution rates from a higher level language, either (1) the present Fortran compiler must perform a higher level of optimization, (2) vector extensions or a macro capability must be allowed from Fortran, or (3) a new vector-oriented language must be written. The alternative is a scientific library written in assembler; such a library might have to be written above the usual dyadic/triadic level to properly manage the cache memory.

*We assume that the Fortran code is vectorized, but no other special Fortran programming techniques are used to force the compiler to produce more efficient code.

References

[1] Calahan, D. A., W. N. Joy, and D. A. Orbits, "Preliminary Report on Results of Matrix Benchmarks on Vector Processors," Report SEL #94, Systems Engineering Laboratory, The University of Michigan, May 1976.

[2] Keller, T. W., "CRAY-1 Evaluation, Final Report," LASL Report LA-6456-MS, December 1976.

[3] Weilmuenster, K. J., and L. M. Howser, "Solution of a Large Hydrodynamics Problem Using the STAR 100 Computer," NASA Report TMX-73904, Langley Research Center, Hampton, Virginia, 1976.

[4] MacCormack, R. W., "An Efficient Numerical Method for Solving the Time-Dependent Compressible Navier-Stokes Equations at High Reynolds Number," NASA Report TMX-73,129, Ames Research Center, Moffett Field, California, July 1976.

[5] MacCormack, R. W., and B. S. Baldwin, "A Numerical Method for Solving the Navier-Stokes Equations with Application to Shock-Boundary Layer Interaction," AIAA Paper 75-1, presented at the AIAA 13th Aerospace Sciences Meeting, Pasadena, California, January 20-22, 1975.

[6] Orbits, D. A., and D. A. Calahan,"Data Flow Considerations in Implementing a Full Matrix Solver with Backing Store on the CRAY-1," Report SEL #98, Systems Engineering Laboratory, University of Michigan, September, 1976.

D26

REVIEW OF THE AIR FORCE SUMMER STUDY PROGRAM ON THE INTEGRATION OF WIND TUNNELS AND COMPUTERS

Bernard W. Marschner
Professor, Computer Science Department
Colorado State University
Fort Collins, Colorado 80523

ACKNOWLEDGEMENT

The material presented here is abstracted or summarized from the two volume report

VOLUME I - EXECUTIVE SUMMARY

VOLUME II - DETAILS OF SUMMER DESIGN STUDY

which was performed under contract R02-400178 sponsored by the Air Force Office of Scientific Research. The study was conducted at the University of Tennessee Space Institute with considerable support by the Arnold Engineering Development Center at Tullahoma, Tennessee.

The list of participants is given in Appendix IA, and the AFOSR Steering Committee is in Appendix IB.

SUMMARY

The Summer Design Study Group at the University of Tennessee Space Institute studied the status of integration of computers with wind tunnels. The study was begun with a series of presentations made to the group by industry, government, and university workers in the field. The background of the individuals making the presentation covered a broad spectrum of viewpoints and experience from computer design, theoretical analysis, computational aerodynamics, wind tunnel technology, and flight vehicle design. Each of the speakers had in-depth discussions with the Design Group as a whole or with one or more of the three panels:

(1) Experimental Methods

(2) Computational Fluid Dynamics

(3) Computer Systems

An extensive literature survey and review was undertaken. The Design Study, as it progressed, focused primarily on the following aspects:

(1) exploration of the present state of computational fluid dynamics and its impact on the design cycle and computer requirements for future developments in this field;

(2) the increase in productivity and efficiency which experimental facilities can achieve by a close integration with computers;

(3) improvements in simulation quality of wind tunnels possible in conjunction with computer control;

(4) research experiments necessary to provide a better understanding of the physics of fluid flow and to assist in the modeling of these phenomena for computational methods, with primary emphasis on turbulent flows.

A Steering Committee, whose membership represented a spectrum of specialized talents from universities and governmental agencies, assisted the Technical Director in delineating the scope of the study.

OBJECTIVES

The following objectives guided the Design Study. These objectives were arrived at in guidance meetings between the Technical Director and the Steering Committee before the study began.

(1) To provide a design study experience on a realistic and pertinent engineering subject for the faculty participants.

(2) To ascertain the current status of experimental aerodynamic facilities and test methods and the current status of aerodynamic computational methodologies and computer systems.

(3) To prepare an estimate of future developments in experimental and computational aerodynamics consistent with projected design needs, with special emphasis on the impact of the next generation of experimental and computational facilities.

(4) To explore means of obtaining and improving aerodynamic data by developing concepts for integrated use of computers and wind tunnels.

(5) To prepare the faculty participants to make future contributions in the area of experimental and computational aerodynamics.

CONCLUSIONS AND RECOMMENDATIONS

Since the Summer Study Group investigated a broader subject than the scope covered by this conference, only those items concerned with computational fluid dynamics will be covered.

Not all of the recommendations are repeated here; rather, a number of the recommendations are combined and reorganized and presented in a more overall summary fashion. The reader is referred to Volume II, Details of Summer Design Study, for the supporting material for the various conclusions.

The general conclusions and recommendations are as follows:

(1) The pacing item for progress in computational fluid dynamics is an understanding of the physical fluid flow with turbulence. A continuing level of effort in fundamental studies of turbulence is necessary for progress in the derivation of physically reasonable and consistent turbulence models.

(2) The real-time availability of a modern large-scale computer during the conduct of wind tunnel tests on which the design computer program results could be used for comparison with test results would improve the design process by verifying numerical optimization and by allowing the examination of only critical areas. At a minimum, planning should be begun for remote terminals with graphics capabilities connected to the aircraft designer's computer for access from the tunnel control room.

(3) In the area of computational fluid dynamics, efforts should be made to give researchers in the field an easier access to some of the very large sequential machines presently installed in the United States. A freer access to the machines for computational work will improve the understanding of the mathematics, numerical methods, and fluid mechanics in this field by allowing more of the researchers access to suitable machines.

(4) Parallel to this effort in numerical experimentation, serious consideration and support should be given to the mathematical aspects of computational fluid dynamics. This work will pace the development of methods of solutions and greatly affect the subsequent choice of computer architectures.

(5) The efforts to conduct design studies on future machines which have special abilities for the solving of three-dimensional time-averaged Navier-Stokes (Reynolds) equations should be pursued. These design studies should include a significant amount of simulation activity and a rather complete development of the software; this is particularly true of the operating system. Proposed vectorized architectures should be simulated on existing host machines, and a large number of timing studies of various architectures should be made to assist in setting the critical design parameters of a large-scale computing system.

(6) Various investigators examining advanced architectural concepts such as the class of machines of the multiple instruction, multiple data (MIMD) type should be encouraged, as should individuals pursuing software developments for presently conceived parallel, or pipelining machines. In particular, considerable effort should be given to the area of developing vectorizing software in order to make this class of machine more user-oriented. Otherwise, computational fluid dynamicists will need the additional skills of computer scientists.

In particular, in the problem areas of computational aerodynamics on which the possible new generation of computers may be used, various additional observations were made.

In the computational solution of fluid dynamics problems:

(1) The discretized formulation should satisfy the integrated conservation laws for arbitrary combinations of discretized volumes throughout the field of computation to the desired order of accuracy (not merely the local truncation errors).

(2) An error analysis should accompany each computational solution with the sensitivity and influence of the arbitrary parameters inherent in the discretized formulation, documented, both in the interior and on the boundary. An absolute error bound of key results should be made, with breakdown of the sources of errors if at all possible, and at least the most important ones identified.

(3) Analysis of the discretized formulations and their solutions of meaningful models of Navier-Stokes equations should be encouraged to establish simple and narrow upper bounds of the various error sources. The most important one is the accumulated discretization error for coarse mesh computations when the mesh Reynolds number is large.

(4) Analysis of the discretized formulations of the Navier-Stokes equations with and without turbulent modeling transport equations under nontrivial boundary conditions should be encouraged, especially in connection with the techniques of rendering a poorly posed problem "well posed" for computational purposes.

(5) Development of algorithms and logic for the solution of initial boundary value problems of Navier-Stokes equations particularly suited to take advantage of parallel computers should be encouraged.

(6) Super computers for solving complex fluid dynamics problems should possess balanced speeds for scalar and vector processing rather than having orders of magnitude difference in the two modes of operation.

In the computer panel, some of the observations were:

(1) To foster the communication and cooperation essential to progress in computational and experimental aerodynamics, an annual conference sponsored by the aerodynamics societies in cooperation with interested government agencies be conducted on the theme "computers and wind tunnels." The thrust of this technical meeting should be the mutual interaction of computation, experiment, and computers as a unified topic.

(2) The development of a computational aerodynamic computer system should be orderly and systematic. Current scientific computers should be used to verify and improve computational procedures and should be used to simulate the performance of proposed advanced computer architecture prior to the implementation of a computer design.

(3) Computing systems should be made available to the entire aerodynamics community. Current scientific computers should be made available as soon as possible for the verification and simulation studies mentioned above. The advanced computers should also be widely accessible to foster further developments in computational aerodynamics.

(4) Government operation and ownership of the advanced computational aerodynamics computing facilities seems inevitable from a financial point of view. It is strongly recommended that these facilities remain free of domination by government agencies to preclude the exclusion of any sectors of the computational aerodynamics field.

(5) The development of software suitable both to the machine and to the programmer is as crucial as the machine design itself. A vector high level language and a vectorizing precompiler should be developed to suit the advanced computer and the problem.

(6) An annual workshop on the topic of computers and wind tunnels should be conducted by interested government agencies, such as AFOSR, in cooperation with the aerodynamics societies. The thrust of this technical meeting should be the mutual interactions of computation, experiment, and computers as a single topic.

APPENDIX IA

L. E. Broome
Mathematics Department
Moody College
Galveston, Texas 77553

Donald A. Chambless
Mathematics Department
Auburn University at Montgomery
Montgomery, Alabama 36117

Sin-I Cheng
Aerospace Engineering Department
Princeton University

Frank G. Collins
Aerospace Engineering
UTSI
Tullahoma, Tennessee 37388

James R. Cunningham
School of Engineering
UT-Chattanooga
Chattanooga, Tennessee 37401

Gregory M. Dick
Division of Engineering Technology
University of Pittsburgh at Johnstown
Johnstown, Pennsylvania 15904

Salvador R. Garcia
Maritime Systems Engineering
Moody College
Galveston, Texas 77553

William A. Hornfeck
Electrical Engineering Program
Gannon College
Erie, Pennsylvania 16501

James A. Jacocks
Senior Engineer
PWT/ARO, Inc.
Arnold AFS, Tennessee 37389

Michael H. Jones
Engineering Division
Motlow State Community College
Tullahoma, Tennessee 37388

Bernard W. Marschner
Computer Science Department
Colorado State University
Fort Collins, Colorado 80523

Vireshwar Sahai
Engineering Science Department
Tennessee Technical University
Cookeville, Tennessee 38501

Carlos Tirres
Engineering Division
Motlow State Community College
Tullahoma, Tennessee 37388

Robert L. Young
Associate Dean
University of Tennessee Space Institute
Tullahoma, Tennessee 37388

APPENDIX IB

Hans W. Liepmann, Chairman	Director, Graduate Aeronautical Laboratories California Institute of Technology Pasadena, California 91125
Gary T. Chapman	Aerodynamics Research Branch Code FAR NASA-Ames Research Center Moffett Field, California 94035
Wilbur Hankey	Air Force Flight Dynamics Laboratory Wright-Patterson AFB Ohio 45433
David McIntyre	Air Force Weapons Laboratory/AD Kirtland AFB New Mexico 87117
Richard Seebass	Department of Aerospace Engineering University of Arizona Tucson, Arizona 85721

SESSION 9

Panel on COMPUTER ARCHITECTURE AND TECHNOLOGY

Tien Chi Chen, Chairman

MULTIPROCESSING TRADEOFFS AND THE WIND-TUNNEL SIMULATION PROBLEM

Tien Chi Chen
IBM San Jose Research Laboratory
San Jose, California 95193

1. Computer architecture

Like an architect for housing structures, the computer architect has to honor many constraints, such as: the laws of nature; available technology; bounds on time, manpower, and budget; demands for performance, reliability, availability, and serviceability; and, last but not least, user habits and society mores.

Not all of the constraints are absolute; most are elastic, and can be the subject of tradeoff. The architect tries to reach the best compromise, to minimize costs and maximize economy for the manufacturer and the users. Computer architecture is an art rather than a science.

2. Multiprocessing tradeoffs

The machine should be capable of general processing, but should be geared to do the intended job particularly well. A knowledge of the expected application is essential in making the proper design choices.

For the NASA wind-tunnel simulation computer problem, the goal is a one-gigaflop machine to handle the three-dimensional hydrodynamical differential equation with about 100 mesh points along each direction, each mesh point being associated with about 40 floating-point words.

The gigaflop general purpose computer is not visible at the horizon, in terms of the current silicon technology (some people will say instead, "cooling technology"). To be deliverable in 1982, the machine design must rely on multiprocessing, to exploit the high degree of inherent parallelism in the job specification.

We shall discuss briefly the multiprocessing tradeoff problem.

3. Dimensions of multiprocessing

The computation involves many time-steps; during each time-step a complete sweep of the million-point mesh in each of the three dimensions is needed, consuming many floating-point operations on each mesh point.

One possible multiprocessing design philosophy is to cover the entire space domain with processing elements (PEs), in the form of volume multiprocessing. Assuming a mesh point to be indivisible to first order, the highest degree of volume multiprocessing is one million, with one PE per mesh point. Each PE can run at 1 kiloflop, to reach the aggregate rate of one gigaflop. The number of PEs can be reduced, say by subjecting a cube of 8 neighboring mesh points under the control of a PE with eightfold computing power.

While volume multiprocessing apparently is nature's way to produce physical phenomena, it must be used with care in computer design, lest most of the PE's will be idle. It does appear that each use of the implicit method locks up the entire volume, within which essentially one plane is being processed at a time. This algorithm appears to preclude volume multiprocessing.

Next to be considered is plane multiprocessing, assigning a plane of mesh points to an array of PEs. The degree of multiprocessing can match the number of mesh points in a plane, namely 10000, using 100-Kiloflop PEs.

A multiprocessinq system involving up to 10000 PE's appears feasible, though engineers tend to be uneasy over its reliability. Lesser degrees of plane multiprocessing can be obtained by assigning rectangles of, say. k mesh points to a PE of 100k Kiloflop computing power.

Plane multiprocessing is thoroughly consistent with the NASA algorithm for three dimensional computation: during any sweep, each plane can be treated as a vector of 10000 elements, and the space-split computation implies processing corresponding elements of three successive vectors, with no cross-talk whatever.

While the algorithm favors plane multiprocessing, still the computation has to cover an entire volume. Efficient plane multiprocessing requires solving the associated problem of systematic data movement.

Next in rank is line multiprocessing, mapping the work required for a line of mesh points to linear array of PEs. Here the degree of multiprocessing is up to 100, using PEs each running at 10 or more megaflops. Since the NASA algorithm is actually a plane-parallel one, line multiprocessing would imply extra data movement, within the planes.

The final reduction in rank leads to point processing, which involves moving data through a single point-PE. In the simplest form, the point-PE is not subdivided, and its use is just monoprocessing, which for a 1 gigaflop machine is probably infeasible using current technology. Subdivision of the point-PE will create an effect similar to line-and plane processing.

The above crude analysis shows that volume multiprocessing is not feasible, as is point processing. Plane and line multiprocessing, using up to 10000 units, are likely candidates for the wind-tunnel simulation facility.

We note in passing that, partly because of the extra data transport facilities provided, lower dimensional multiprocessing tends to be more flexible. There is no need to match several dimensional widths simultaneously to secure full employment of all PEs. For example, a line-multiprocessing system can emulate plane-parallel computation easily, but not vice versa.

4. Identical modules vs. specialization

After choosing the approximate degree of multiprocessing, there is still the choice of the kind of multiprocessing. The choice here is between identical modules and specialized units.

The identical module approach is exemplified by the ILLIAC IV. This approach works best if the workload can be symmetrically partitioned into subsets, one for each PE. The vector nature of the wind-tunnel simulation problem is ideal for this partition.

The use of identical modules is illustrated in Figure 1, where the job profile, represented by a closed area in the space-time graph, is swept by the processor array of multiplicity m. The sweeping is repeated, each time over a different part of the profile, until complete coverage is achieved. The performance of the system is

$$P = (\text{job profile area})/(\text{total time of sweep})$$

An important form of multiprocessing using specialized units is pipelining. In Figure 1, the job profile for vector processing calls for operations a,b,c,d on each of the elements. It may appear possible to design specialized processors, the k-th one for the k-th operation, to be used together.

The first attempt might lead to the graph in Figure 2; it is unworkable due to possible causality violation. For instance, Operation b may have to work on the results of Operation a for the same vector element, this, is clearly not possible if both are started at the same time. To preserve causality, the k-th layer from the bottom should be offset to the right by k time cycles, resulting in the jagged profile in Figure 3, which can be realized if the processing times are made equal, and if the processors are linked into a linear array, namely a pipeline.

The pipeline performance is again measured by the applying the equation above to the job profile in Figure 3. The triangular regions, representing overheads due to pipeline filling and draining, have diminishing timing cost if the number of vector elements, represented by the width of the jagged parallelogram, is large.

Pipeline systems have the merit that their efficient use requires no knowledge of the number of pipeline segments. However, meticulous design is required to ensure the proper relaying of data; moreover, the number of pipeline segments depends intimately on laws of nature and technology, and a 10000-segment pipeline is hard to conceive at this time. For the problem at hand, a measure of symmetric job partition is probably unavoidable. It is much more reasonable to consider a pipeline unit of s segments, and replicate it r times to yield a throughput proportional to rs.

4. Conclusion

We have discussed the architecture tradeoff issue, concentrating on an oversimplified version of the multiprocessing aspect. It appears that a degree of symmetric multiprocessing is unavoidable; the choice is either complete symmetric multiprocessing or a number of identical pipelines. The processing elements can be geared to do either plane-multiprocessing, or line-multiprocessing.

There are other important design choices, such as the number notation, word length, main memory size, cache memory size, and different means to implement data transport. Clearly multiprocessing is only one item in the computer architect's long list of tradeoff possibilities.

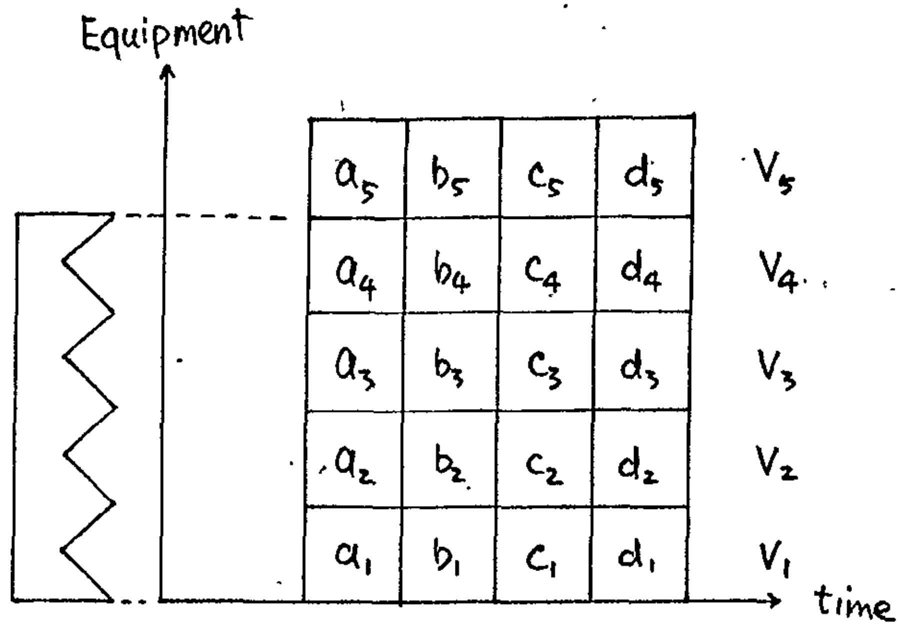

Equipment
a_5 b_5 c_5 d_5 V_5
a_4 b_4 c_4 d_4 V_4
a_3 b_3 c_3 d_3 V_3
a_2 b_2 c_2 d_2 V_2
a_1 b_1 c_1 d_1 V_1
time

Figure 1. Multiprocessing by identical units

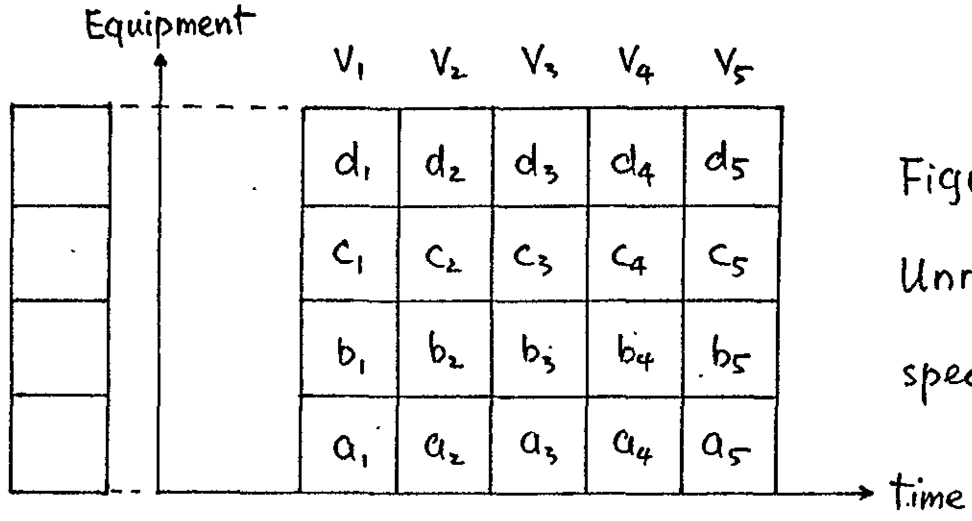

Equipment
V_1 V_2 V_3 V_4 V_5
d_1 d_2 d_3 d_4 d_5
c_1 c_2 c_3 c_4 c_5
b_1 b_2 b_3 b_4 b_5
a_1 a_2 a_3 a_4 a_5
time

Figure 2. Unrealistic use of specialized units

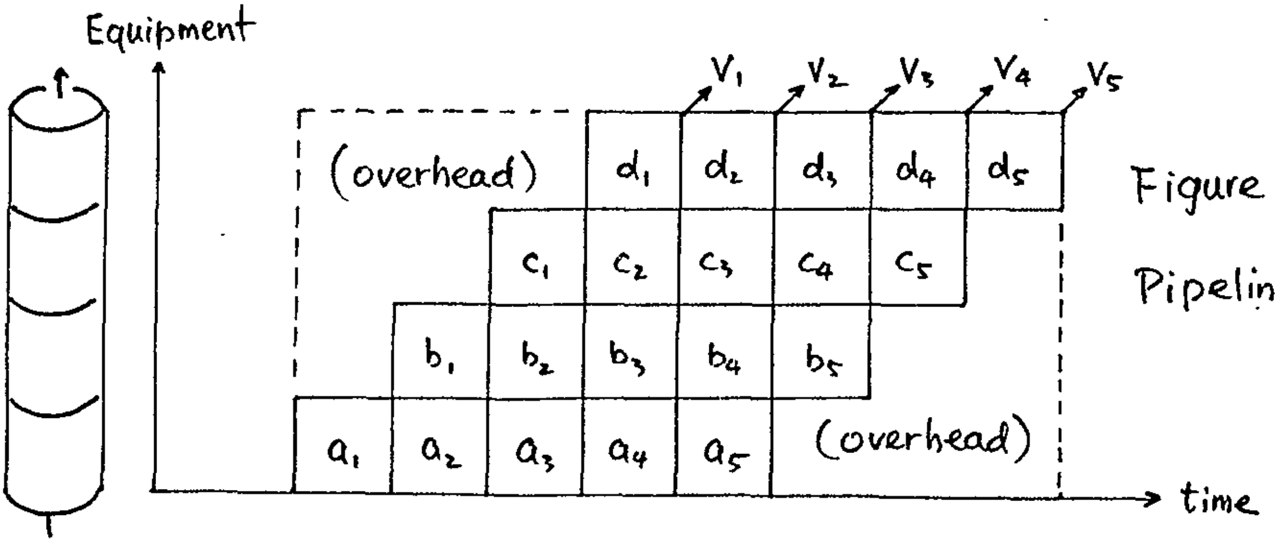

Equipment
V_1 V_2 V_3 V_4 V_5
(overhead)
d_1 d_2 d_3 d_4 d_5
c_1 c_2 c_3 c_4 c_5
b_1 b_2 b_3 b_4 b_5
a_1 a_2 a_3 a_4 a_5
(overhead)
time

Figure 3. Pipeline

D28

TECHNOLOGY ADVANCES AND MARKET FORCES:

THEIR IMPACT ON HIGH PERFORMANCE ARCHITECTURES

Dennis R. Best
Texas Instruments Incorporated
Dallas, Texas

ABSTRACT

Reasonable projections into future supercomputer architectures and technology requires an analysis of the computer industry market environment, the current capabilities and trends within the component industry, and the research activities on computer architecture in the industrial and academic communities.

The supercomputer market is not a major driving force in the development of computer equipment and components. Development resources are being used to solve the problems of the small systems user. Equipment development is concentrated on the peripheral and mass storage segments and component development is obtaining major advances in circuit density of conventional speed microprocessor and memory devices, but little progress on ultra high speed technologies.

The successful supercomputer of the future will attain its goals only by exploiting all levels of parallelism in problem descriptions on computer structures built of conventional logic for other end-user requirements. The partitioning of the problem onto the architecture must be automatic as ad hoc partitions are neither cost-effective nor sufficient. Both program control and data structures must be distributed across an architecture of many low cost microprocessor and memory devices with the key to success being the efficient handling of processor/memory intercommunication.

Management, programmer, architect, and user must cooperate to increase the efficiency of supercomputer development efforts. Care must be taken to match the funding, compiler, architecture and application with greater attention to testability, maintainability, reliability, and usability than supercomputer development programs of the past.

INTRODUCTION

We at Texas Instruments have survived a ten year experiment toward breaking the bonds of "300 years of basically sequential mathematics, 50 years of sequential algorithm development, and 15 years of sequential Fortran programming" in an attempt to approach the 100 million instructions per second computational barrier. With the scars of battle still painful, we now stand before you to project the tools and techniques available to address the requirement for a staggering computational speed of one billion operations per second!

In doing so we will attempt to follow the advice of that sage philosopher Satchel Paige - "Don't look back. Somethin' might be gaining on you" - and leave the analysis of prior battles to the session on Supercomputer Development Experience. However, this prior experience with Texas Instruments Advanced Scientific Computer (ASC) will, hopefully, tinge our visions of the future supercomputer with the realities of "making it work".

Reasonable projections into the future of supercomputer architecture and technology requires

1) an analysis of the current market environment within the computer industry,

2) an examination of the current activities, capabilities, and trends within the component industries, and

3) a discussion of the current activities within the industry and academic research communities on computer architecture.

Then, we can project the architectural features that meet the supercomputer user requirements of performance and, often under-emphasized, usability, maintainability, and reliability. We will then summarize the problems to be solved by management, architect, and programmer in order to provide a viable solution to our computational goals.

Before examining these areas, let me first detail my position on future computer architecture and technology:

> The supercomputer market is no longer a major driving force in the development of computer equipment and components.
>
> There are no indications of an imminent breakthrough in ultra high speed circuit or interconnect technology that will allow even an order of magnitude improvement in raw logic speed.
>
> Therefore, the supercomputer of the future will attain its goals only by exploiting all levels of parallelism inherent in the real world on a configuration of computer structures built of conventional logic for other end-user requirements.

THE MARKET FORCES

In the mid-sixties, Texas Instruments initiated the development of the ASC and a high speed ECL logic family to meet an internal requirement for large volume processing of seismic data. The external market for large scientific processors appeared insatiable - current machines were saturated and projected requirements were staggering. However, during the long development cycle of the ASC and other supercomputers the market shifted dramatically. Many large users of processing power discovered that they could meet their requirements by simply installing additional systems like the ones currently in use. That is, their requirement was one of total throughput, not one of minimum time for any single but massive program.

Also during this time frame, the lowering cost and increasing density of digital logic created an entirely new market force - the minicomputer. The low cost and easy to use features of the minicomputer, besides greatly expanding the markets for the computer industry, further chipped away some of the processing requirements previously relegated to the large centralized processor, with techniques now referred to as "distributed processing". Then in the mid-seventies came the microprocessor - further expanding the computer market base, almost to the personal cost threshold, and further reducing the supercomputer's market share. The net result is, in 1977, an installed operational base of supercomputers consisting of seven ASC's, four STAR's, an ILLIAC, a PEPE, a couple of STARAN's and the promise of CRAY's to come.

The net effect of this market shift on the large computer user has been a loss of leverage in the development of key technologies for product improvements. In the 1960's much of the semiconductor industry's independent research and development funds were concentrated on the requirements of the large computer manufacturer. Today, these funds are distributed across many product requirements - from the consumer, scientific, and programmable calculators, through the intelligent terminals and minicomputers to the main frame computers. The projections are for this market shift to continue and in fact accelerate. This market shift has already had a marked effect on computer manufacturers as indicated by the cost trends in Figure 1. The price of main frame computing power has continued to decline by 60% during the past ten years, but that of minicomputers (and now microcomputers) has declined even more sharply.

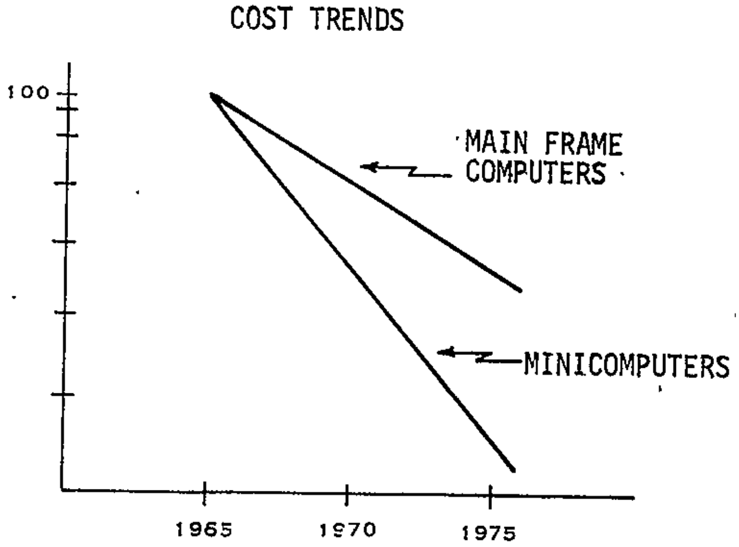


Figure 1

Figure 2 illustrates these market trends. In 1970, 69% of the dollars spent for computer equipment went for systems valued at more than $200K and this will reduce to approximately 24% by 1985.

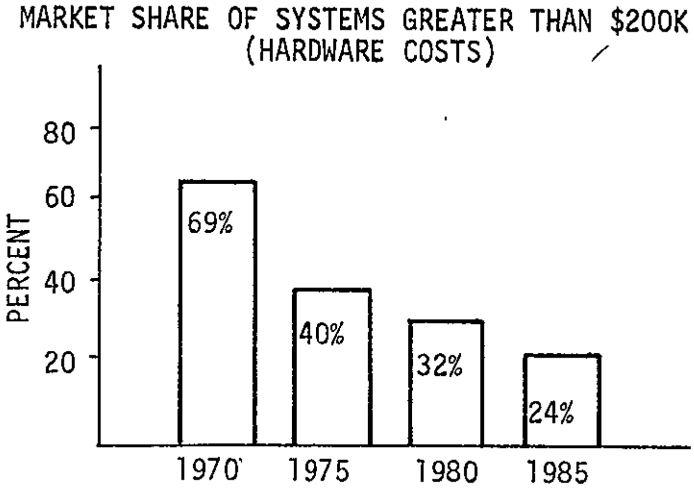


Figure 2

The $200K threshold was dictated by the available market data. Looking at supercomputers in 1985, they would represent less than 1% of an estimated $80B computer equipment market.

There are other market forces that we who would configure the future supercomputers must understand. The first is probably well understood by the attendees of this conference - by 1985, 90% of a computer systems cost will be for software.

Figure 3 illustrates the expected mix of hardware expenditures in the 1980 time frame - 25% for CPU and Memory, 35% for Input - Output devices, and 40% for Mass Storage.

COMPUTER/PERIPHERALS MIX: 1980

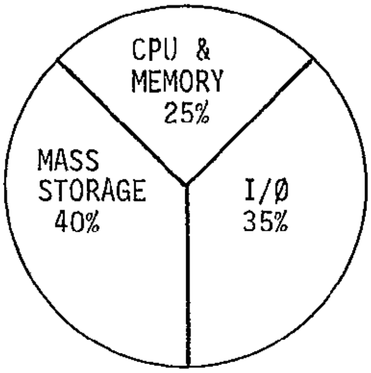


Figure 3

This market shift could have positive results for the supercomputer designer. Our requirements for very large, easy to use, cost effective mass storage devices have not previously been met, and perhaps increasing dollars for I/Ø devices will result in improved peripherals that will alleviate a low level but constant source of irritation to the supercomputer user.

On the negative side, the current and projected computer equipment environment does not support a large investment in the development of ultra high speed component technologies that would allow us to reach our supercomputer goals with conventional architectures.

COMPONENT TECHNOLOGY

Technological advances in the semiconductor industry during the past two decades have been spectacular. Manufacturers have increased the complexity of logic and memory circuits by five orders of magnitude while maintaining a 73% learning curve on costs.

These increases in functional capability, as illustrated in Figure 4, have resulted from advances in circuit architecture, devices structures, processing technology and imaging techniques. Projections are for this progress to continue even though current production technologies are approaching the limits imposed by the wavelengths of light on optical imagery techniques. Advances in electron beam and X-ray lithography should allow the production of a single-chip 32 bit microcomputer with one million bits of memory in the 1980's.

SEMICONDUCTOR CHIP COMPLEXITY

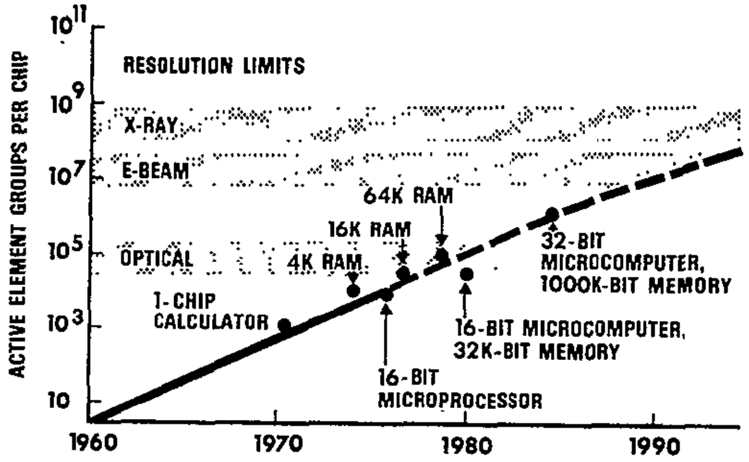


Figure 4

Perhaps of the most interest to supercomputer architects are the advances in memory technology. The extraction, display, and execution of parallelism within a program or set of programs is very memory intensive. Techniques previously abandoned as too costly may soon become cost effective.

The cost reduction trends of computer memory are indicated in Figure 5. Dynamic RAMs, currently available for 0.1¢ per bit, will be reduced by a factor of 10 in the next decade, and static RAM and ROM devices should follow a similar learning curve. The lower cost of programmable ROM can be of particular importance toward meeting usability goals. In addition, the entry of CCD memories, at prices 1/3 to 1/4 that of dynamic RAMs, will allow another level of buffering in the memory hierarchy to smooth the access and distribution of data from secondary storage devices.

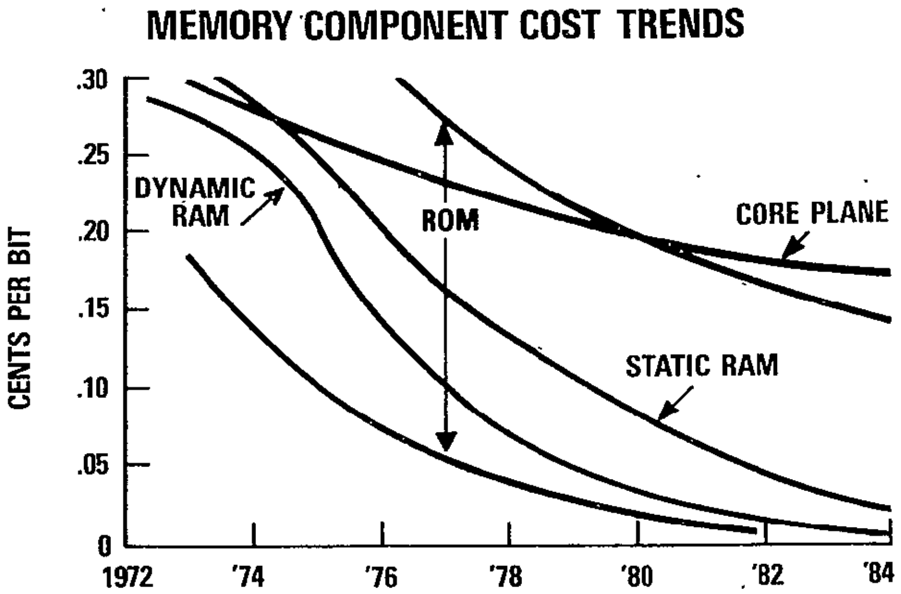


Figure 5

There is also a new tool in the secondary storage area - magnetic bubble memories. 92K-bit bubble memory devices, complete with all necessary control circuits, have been announced by Texas Instruments.

By 1980, with smaller bubbles, it is expected that each device will yield 256K bits of non-volatile storage. Figure 6 illustrates the current and projected cost comparison of bubble memory and magnetic disc storage media.

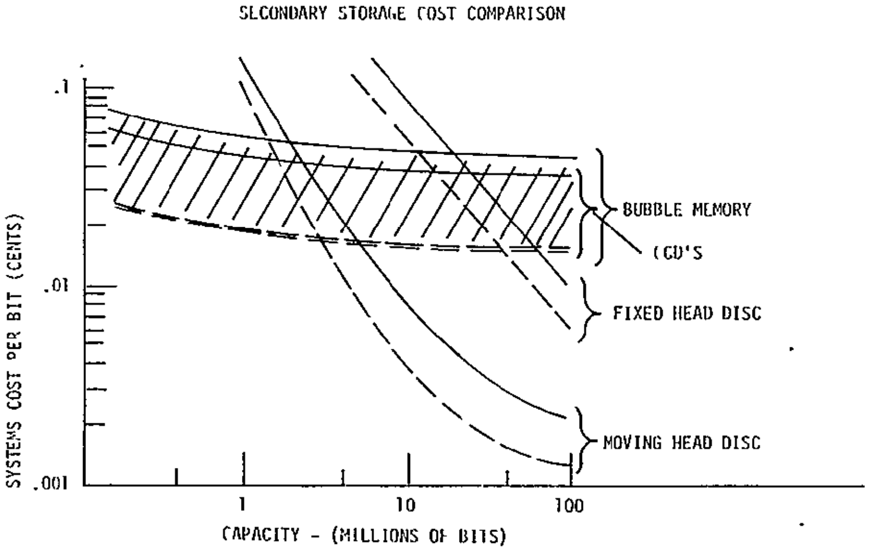


Figure 6

Electron-Beam-Accessed MOS (EBAM) is another developing technology for secondary storage. However, this technology has some limitations, such as limited life and expensive support electronics, but can be used to configure very large memories with fast access (30 μsec) and high transfer rates (10^6 BPS).

Notice that in the discussion of semiconductor technology advances, we have yet to mention ultra-high speed devices. The development record of the semiconductor manufacturers has not been impressive in this area. Ten years ago Emitter Coupled Logic (ECL) with 2 nanosecond gate delay was available in Small Scale Integration (SSI) circuits. Today, it has progressed to Medium Scale Integration (MSI) with a minimum gate delay of 0.8 nanosecond.

The limitation is the power dissipation constraints of the chip and package. The expense of the sophisticated cooling techniques and the transmission line quality interconnect required for these high speed/high power devices has limited their further development and utilization. MOS and I^2L, with their high density, low power, fewer processing steps characteristics, and respectable 5 nanosecond gate delay, will be the technology used in most logic applications of the future. Schottky TTL, and on a much more limited scope, ECL, will continue to be used for a wide variety of high-performance applications.

Progress has been made in the cooling, packaging and interconnect technology. The 19 layer ASC transmission line quality Printed Circuit Boards and the sophisticated cooling technique used by the CRAY I are prime examples. However, these solutions are expensive. Cost reductions for interconnection and packaging have not kept pace with the semiconductor learning curve. Costs for TTL logic on a per gate basis have been reduced by a factor of 60 in the past 10 years whereas the costs for assembled TTL has been reduced by a factor of only 15.

Therefore, to build truly cost effective large scale computation systems, we must learn to take advantage of the conventional speed, but very high density microprocessor and memory devices using conventional (low cost) packaging and cooling techniques.

ACADEMIC AND INDUSTRIAL RESEARCH

It is difficult to examine the R & D activities of the computer industry. Until breakthroughs are announced, only an interpretation of what the various manufacturers consider to be the critical issues can be obtained. However, one new vector machine has been described in the literature - the Burrough's Scientific Processor (BSP). With this design, Burroughs should prove or disprove the statement many have made about the ILLIAC - "the concept was good, but the implementation was flawed". The array memory answers the parallel PE access problem, the input and output cross bars address the data alignment and inter-PE communication problems, the CCD file memory answers the disc paging problem, and the hardware is fully exploitable from Fortran.

Research in the academic community falls in two major classes: the determination and measurement of program parallelism and the implementation of loosely coupled multi-minicomputer networks. These latter efforts, such as the CM* machine at Carnegie-Mellon and the PLURIBUS at Boston, have a formidable problem - inefficient and complex interprocessor communication of data and control.

Texas Instruments has under development for the FAA for Air Traffic Control a similar implementation called the Discrete Address Beacon System (DABS). This collection of more than 32 TI 990 minicomputers, with great attention to reliability and error recovery through hardware redundancy and error detection and software error recovery, will offer a cost effective solution to the application. However, fitting the application to the architecture is a long-term, expensive, ad hoc partitioning of the functional parallelism of the tasks to be performed and is cost effective only because of the large number of identical systems that will eventually be deployed.

Research continues in the definition of new languages that allow for the description of the application for maximum exploitation of parallelism. One of the more interesting of these is the single assignment languages/architectures being proposed by Jack Dennis of MIT and Jean-Claude SYRE of France due to the potential data directed hardware implementations that address the intercommunications of data and control problems and exploitation of program parallelism.

There is of course one problem with any new language - user acceptance. The momentum toward further refinement of sequential languages is not easily re-directed, as evidenced by the problems of getting vector extensions into standard Fortran. It appears for the near term we are stuck with Fortran, and application programmers are required to also be systems programmers and hardware architects to successfully generate high speed solutions to their problems.

ARCHITECTURE

Our success in the development of the future supercomputer lies in our ability to exploit parallelism and thus the high density memory/processor technology. We must regain our lost leverage by concentrating on the use of available technology as opposed to technology itself.

For example, we can utilize emerging "Distributed Processing" techniques to further reduce the processing requirements of the back-end "number cruncher". Low cost, conventionally programmed computers can perform the data preparation, data management, and output formatting, analysis and display functions. Although some problems in distributed processing still must be solved - i.e., effective file management structures with some hierarchy of storage control imposed on the system - the solutions will be generated by development in the mainstream of the computer business. The supercomputer user need only make a cost effective selection of these equipments and techniques.

The key characteristics of the successful supercomputer mainframe appear to be incompatible - simple but powerful; expandable without huge redevelopment programs; adaptable to different processing requirements; straight forwardly programmable; and cost effective but not necessarily hardware efficient. But I believe we can develop architectures with these attributes if we discard our sequential thought processes and the idea that we will somehow be successful in fitting applications to a predefined hardware structure.

First, we must develop a compiler that exposes all levels of parallelism within a single program (job). Ad hoc functional partitioning is not sufficient - the partitioning must be automatic, application independent, and include more than functional parallelism. Nor will the simple structural array parallelism of the past be sufficient. The compiler must display parallelism at the program, task, sequence, statement and instruction level in a machine-independent format. By remaining machine independent we can create an evolutionary hardware/software structure that can take advantage of hardware or software advances without major redevelopment. We also avoid the binding of addresses or resources at compile time, thus avoiding a recompilation to accommodate the loss of a processor or memory element or to take advantage of an expansion of processing/memory elements. Of course, the parallelism must be displayed in a format that is readily interpreted by a loader and/or directly by the hardware.

Only after the compiler is judged effective do we consider hardware. Again, I believe this will be an interconnected set of high-density processors and memory. The key to useful application of this network is the distribution of both data and control, including synchronization, across the network. The control distribution must be complete - that is, if any one node must perform synchronization monitoring and scheduling functions for other nodes then our success will be limited. The second key attribute is the simplicity of internodal communication - i.e., the communication protocol must be much simpler than those we have seen in the loosely connected minicomputer systems.

Techniques for handling the data and control distribution and intercommunications problems are very memory intensive. Merely documenting the program parallelism and synchronization requires memory beyond conventional requirements and simplification of communications requires a huge address space and thus even more program memory. However, optimization of memory size is less important in the era of 1-megabit memory chips with a free processor and ROM with each device. Operations (results) per unit time per dollar is the measure of our success.

Note that there has been no discussion of whether the network nodes will be processor/memory pairs or separate processors and memories, nor of the conventional architectural features of registers, pipelining or memory cycle overlapping. These features are merely processor optimizations that take advantage of local parallelism to improve sequential performance and, I suspect, often clouds our vision of the necessary architectural features to provide a step function in computational performance.

SUMMARY

The successful architecture of the future will be a network of conventional speed but high density microcomputer devices. The simplistic structure of these devices should offer greatly improved reliability and maintainability characteristics over implementations of ultra-high speed and high power systems.

But there remains many problems to be solved that will require the cooperation of the manager, programmer, architect, and user. Development funding must be spent only on efforts that offer step function improvements as opposed to mere enhancements or expensive software conversion efforts for small gains in performance. The systems software specialist and application programmer must cooperate to insure that usability goals are met and that maximum parallelism within the job can be exposed. (yes, even with FORTRAN source code!) Greater attention, both in funding and design, must be given to both hardware and software testability, maintainability and thus usability. The resource independent compiler and distributed control architecture described can enhance this usability if techniques for efficient error detection and localization can be developed.

The coming availability of very large, reliable, low cost memory devices has provided the vehicle that will allow the construction of a truly parallel, general purpose computer architecture. If the user provides the necessary funding and encouragement to system designers that understand the performance and usability requirements and are able to replace their ingrained sequential intellect with parallel thought processes, then a truly useful system that meets the performance goals will be available in the early 1980's.

D29

GIGAFLOP ARCHITECTURE, A HARDWARE PERSPECTIVE

N78-19807

Gary F. Feierbach
Institute for Advanced Computation
1095 East Duane Avenue
Sunnyvale, CA 94086

INTRODUCTION

Any super computer built in the early 1980s will use components that are available by fall 1978. It will have to cost less than $100 million since people are not acclimated to spending more than that amount for a given installation. An availability of greater than 90% will be demanded of such a facility to amortize the cost over the expected lifetime of the system. The architecture of such a system cannot depart radically from current super computers if the software experience painfully acquired from these computers in the 70s is to apply. Given the above constraints, 10 billion floating point operations per second (BFLOPS) are attainable and a problem memory of 512 million (64 bit) words could be supported by the technology of the time.

In contrast to this, industry is likely to respond with commercially available machines in the $10-15 million price range with a performance of less than 150 MFLOPS. This is due to self-imposed constraints on the manufacturers to provide upward compatible architectures (same instruction set) and systems which can be sold in significant volumes. Since this computing speed is inadequate to meet the demands of computational fluid dynamics, a special processor is required.

The following issues are felt to be significant in the pursuit of maximum compute capability in this special processor.

PERFORMANCE AND COST

It should be obvious that a processor will have to have multiple functional units in order to obtain the projected capabilities. An important trade-off must be made between functional unit power and the number of such functional units. If functional unit cost is plotted against power, then a knee-of-the-curve rule indicates increasing the computing power of a processing module until the incremental cost to obtain that power increases dramatically. The second most important factor influencing cost is useful memory bandwidth. A surface representing cost as a function of memory bandwidth and processor power as independent variables is shown in Figure 1. The steps in the memory bandwidth direction represent switching technologies from NMOS to Bipolar to using fast ECL register files. The line on the surface represents the cost as a function of memory bandwidth as it relates to processor power. The heavy section of the line represents a reasonable zone in which to select processor power for a functional unit. The choice may be narrowed by considering problem sizing (what are the natural dimensions of the problem being considered and are they commensurate with the number of functional units), function unit interconnection (the cost of which increases by at least O(N log N) where N is the number of functional units), and reliability considerations which usually dictate minimizing the number of processing modules.

RELIABILITY

It is not currently possible to build very large systems and expect all components to be operational at the same time. For memory modules, this means that information must be coded in such a way that error correction is possible. For processor modules, this means that spare modules must be built into such a system and a means provided for automatic switching on fault detection. It further indicates that fault detection must be built into the processor to initiate such an automatic response. Figure 2 shows reliability in mean time before failure (MTBF) in hours as a function of total system processing power (computed from past counts using current technology). The three curves represent systems with no error correction systems, with single bit error correction double bit error detection (SECDED) on memory and systems with memory SECDED and automatic processor switching on processor error detection (assuming hardware fault detection in each processing element). It's quite clear from Figure 2 that above 1 BFLOP both SECDED and processor switching are required.

MAINTAINABILITY

The maintenance of a large system poses problems in scale and complexity that must be faced by the system designers at the onset. The system must be comprehensible, accessible and testable. The ability to isolate faults with components in place is a necessity. These issues play an increasingly important role as the size of a system increases.

Figure 3 gives a summary of techniques that are helpful in bringing up a large system, keeping the mean time to repair (MTTR) low or maximizing the MTBF.

Starting with a comprehensible, modular design will minimize the system checkout phase at installation and the MTTR thereafter. A system whose complexity exceeds the capacity of those who would maintain it is in general not going to be maintainable.

Hardware features that aid the technician in fault location include SCAN IN and SCAN OUT which is simply a means for loading and reading all internal registers from the "front panel". The front panel itself does not have to be a real entity but merely another interface from the special processor control unit to the host processor or to a diagnostic processor. SECDED, parity, residue checks and other fault conditions should be available to this same "front panel" interface. Another useful feature is a programmable clock which will allow the machine to be single stepped, advance N clocks, advance N instructions, SCAN OUT every N clocks, etc. Such a clock would also allow a "reverse" clock action by stepping forward N instructions from some initial condition, then stepping forward N-1 instruction from the same initial conditions, then N-2, etc. Often machine bugs are difficult to find because the information necessary to locate the problem is destroyed by the problem. A "reverse" clock can easily pin down such a problem.

At some point the technician may have to actually look at signals with oscilloscopes or other such instruments. Conveniently located test jacks with appropriate signals, accessible back planes and easily removable subunits would aid such conventional troubleshooting.

Software diagnostic procedures should be used to isolate machine problems whereever possible. Thermal cycling, shock or mishandling can damage electronic equipment and can generally be minimized by extensive use of software diagnostic tools. A system level approach is desirable using a diagnostic monitor which runs a prescribed series of confidence tests which if fail, call in a lower level set of diagnostics to isolate the fault. Fault symptoms could also be sent through a simulator of the subsystem that failed which would exhaustively find all possible "stuck" faults (failures with constant symptoms over test duration) that produce those symptoms. This approach is in regular use on the ILLIAC IV system and is embodied in two programs, PESO and TRIP.

Software should allow any terminal on the host system to become the processor front panel, to allow simple programs to be directly entered and executed on the processor and to allow analysis of memory as dumps from the processor proceed through other diagnostic procedures.

Any terminal on the system should also have access to all relevant documentation on the processor. A large contribution to MTTR in the early period of the ILLIAC IV operation was due to technicians searching for relevant and up-to-date information.

No discussion of maintenance is complete without discussing training of technicians. In the case of the ILLIAC IV technicians, heavy emphasis has been placed on the use of software tools and equipment handling. The ILLIAC IV presents some special problems in the equipment handling area. Its fragile nature dictates ginger handling so technicians have been trained to handle the equipment with tender loving care (TLC). The equipment performance improvement on application of TLC was so dramatic that it is recommended that such training should be given all computer technicians.

MANUFACTURABILITY

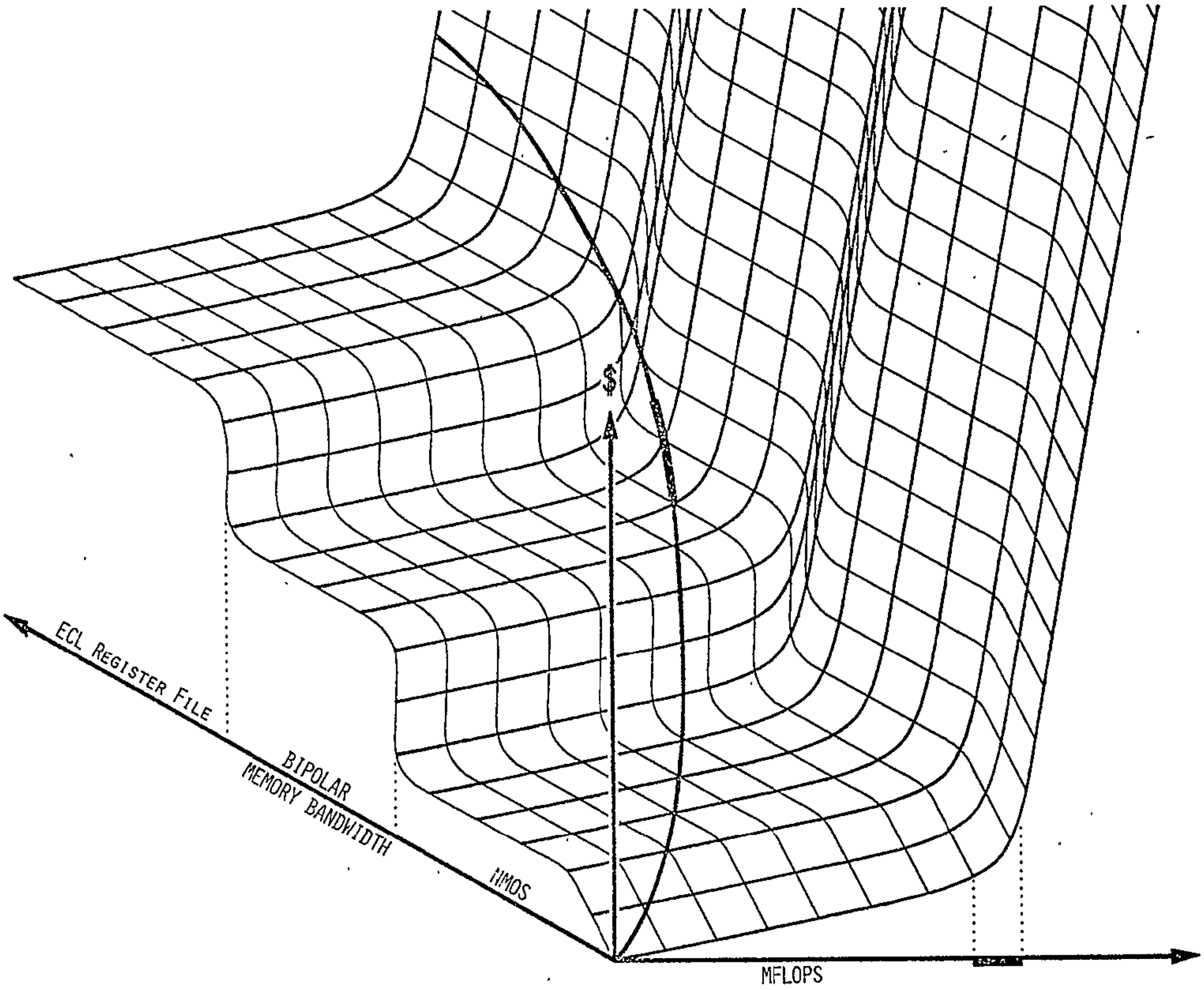
The system must be fabricated using interconnection techniques of very high reliability since such a system could have up to 30 million connections. Careful packaging design and vendor selection can meet this objective if followed by a rigidly enforced quality assurance program. The system should be assembled from a small variety of identical subunits. This is necessary for a successful application of a QA program and a reasonably short design/debugging cycle. Economies of scale and system comprehensibility are also achieved by this means.

Figure 4 contains a list of some of the questions or issues that must be addressed if the processor is to be successfully fabricated. Briefly, the quest for greater processor speed causes more power to be dissipated per gate on the one hand and closer proximity of parts on the other. This imposes constraints on level of integration, power distribution, cooling, packaging and interconnections that interfere to some extent with a top down design approach. A certain amount of design look ahead and back tracking is necessary to come up with a workable design that can indeed be manufactured and debugged.

To some degree, lessons learned from structured programming can be and are routinely applied to processor design. Modular design, for example, can allow the checkout of subunits, in large measure, to substitute for overall integrated system checkout. Exhaustive checkout of modules may be possible whereas exhaustive checkout of the overall system is rarely ever possible.

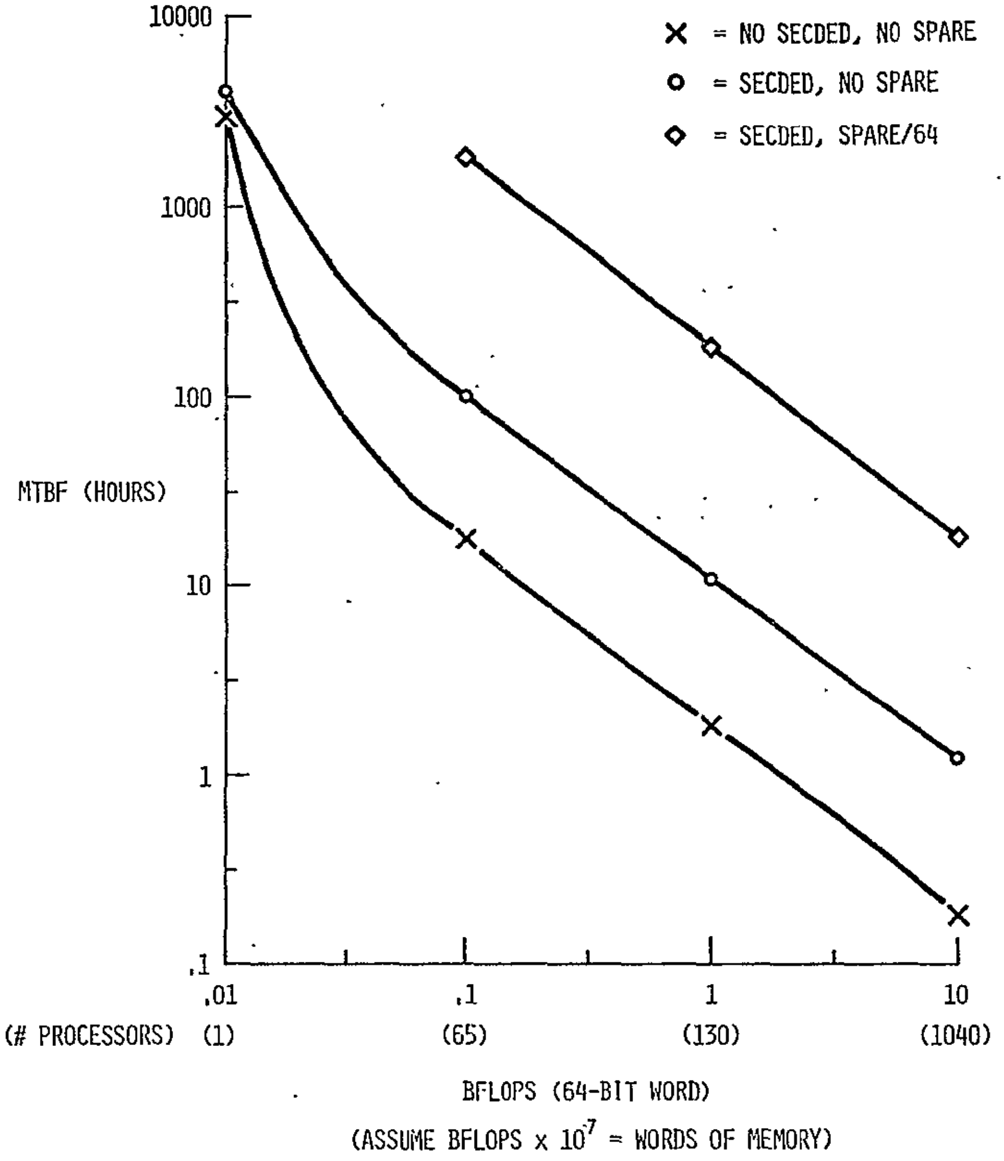
CONCLUSION

Designers of large processors of the type envisioned to do wind tunnel simulations will have all the problems one meets when designing smaller processors. These problems will reach a new level of visibility with the imposed availability requirements on the total system. Much care will have to be given to all aspects of the system design from the specification and testing of IC chips to architectural issues such as automatic processor switching if we are not to contribute yet another blemish to the history of super computers.



COST TRADEOFFS

Figure 1



SYSTEM RELIABILITY
Figure 2

FAULT ISOLATION AND MAINTAINABILITY

HARDWARE

SCAN OUT

SCAN IN

FRONT PANEL PROCESSOR

PROGRAMMABLE CLOCK

SECDED AND PARITY

ARITHMETIC RESIDUE CHECKER

FAULT TRAPS

TEST JACKS

SOFTWARE

DIAGNOSTIC MONITOR

CONFIDENCE TESTS (COMPLETE SET)

DIAGNOSTIC TESTS (COMPLETE SET)

PE SIMULATOR

ONLINE DOCUMENTATION

SOFTWARE FRONT PANEL

SELECTIVE DUMP AND DUMP SCANNING ROUTINES

SYMBOLIC DEBUGGER

DESIGN

COMPREHENSIBLE

MODULAR

TRAINING

TRAINING DOCUMENTATION

SOFTWARE EMPHASIS

TLC

FIGURE 3

MANUFACTURABILITY

ARCHITECTURAL IMPLICATIONS

PROXIMITY - HOW CLOSE DO SUB-UNITS HAVE TO BE?

ARE THERE EXTRAORDINARY COOLING PROBLEMS AT THIS DENSITY?

HOW IS POWER TO BE DISTRIBUTED?

MODULARITY - CAN THE SYSTEM BE THOROUGHLY TESTED BY THOROUGHLY TESTING THE SUB-UNITS?

CAN THE SYSTEM BE MADE OUT OF A MINIMUM NUMBER OF IDENTICAL SUB-UNIT TYPES?

INTERCONNECTION - ARE THE CONNECTIONS BETWEEN SUB-UNITS MINIMIZED?

COMPONENT AVAILABILITY - ARE CUSTOM CIRCUITS REQUIRED?

DOES THE DESIGN TAKE ADVANTAGE OF COMMERCIALLY AVAILABLE SUB-UNITS OR SUB-SYSTEMS?

IS THE TECHNOLOGY AVAILABLE TO ATTAIN THE DESIGN GOALS?

PACKAGING - IS A PACKAGING SYSTEM AVAILABLE TO MEET THE DESIGN GOALS OR DOES ONE HAVE TO BE PIONEERED?

DOES THE PACKAGING SYSTEM ALLOW FOR DESIGN CHANGES, TESTING, INEXPENSIVE REPAIR?

FIGURE 4

D30

A SINGLE USER EFFICIENCY MEASURE FOR EVALUATION OF PARALLEL OR PIPELINE COMPUTER ARCHITECTURES

W. P. Jones
Ames Research Center, NASA

1.0 INTRODUCTION

On the premise that early 1980 general purpose computers will not have sufficient computing power to achieve a hundredfold increase in performance over the CDC 7600, special purpose machines such as the STARAN, the PEPE and the CHI computers will evolve to optimize specific computational applications programs. Possible approaches to system architectures for the Numerical Aerodynamic Simulation Facility (NASF) should be analyzed with efficiency measures that include one that is based on what a single user perceives.

The critical design issues of the NASF from the user view are predictability of service, reliability of hardware and software, and feasibility of a computation. From the system developer view, cost, maintainability and flexibility of the facility are paramount. An approach to the design of the NASF that ensures flexibility of processor and memory interconnections solves two problems. The user can improve the effective rate of computation of a program by specifying that configuration most efficient for the current program. The system can optimize the allocation of this unique resource among several users by dynamically changing the configuration for each user. Parallel and pipeline machines to date exhibit a low degree of performance predictability. Consequently the feasibility of many computational problems, that is, whether or not a computational problem can be completed on the facility in less than one hour, is in doubt.

A precise statement of the relationship between sequential computation at one rate, parallel or pipeline computation at a much higher rate, the data movement rate between levels of memory, the fraction of inherently sequential operations or data that must be processed sequentially, the fraction of data to be moved that cannot be overlapped with computation, and the relative computational complexity of the algorithms for the two processes, scalar and vector, is developed. The relationship should be applied to the multirate processes that obtain in the employment of various new or proposed computer architectures for computational aerodynamics.

The relationship, an efficiency measure that the single user of the computer system perceives, argues strongly in favor of separating scalar and vector processes, sometimes referred to as loosely coupled processes, to achieve optimum use of hardware. Such optimum use can be estimated by a pre-run estimate of the fraction of sequential operations or sequentially processed data, the relative computational complexity, and the fraction of data that must be moved without overlapping computation. The development of applications programs for the NASF can be aided significantly by the use of this efficiency measure. More importantly, the measure will aid in the assessment of alternative designs for the NASF for specific applications programs that are to be developed for it.

2.0 DEFINITION OF TERMS

Let s = number of operations that are inherently sequential or units of data that must be processed sequentially, p = number of nonsequential operations or units of data that can be processed nonsequentially, and ℓ = total number of operations or units of data in the user's program. The time that is required to process the ℓ operations or units of data is ℓ/r where r = effective rate at which the user's program is executed.

Five ratios are introduced. The first, f, is defined as the fraction s/ℓ. Therefore, $0 \leq f \leq 1$. It is determined by the user's program on the assumption that all operations can be identified as purely sequential or not necessarily sequential. Those operations that are not sequential, the fraction $1-f$, are all those that can be processed, potentially at the maximum rate, r_p, whereas the fraction f is processed at rate r_s, where $r_s < r_p$.

Second, the effect of the relative computational complexity, k, is introduced to account for the degradation of performance that results directly from the user's selection of a computational algorithm. The implementation of an algorithm may not realize an n-fold speedup where n is the number of independent processors or stages with which to process the p operations or units of data. The value of k is in the interval $0 < k \leq 1$ and is defined here as the ratio of the number of operations resulting from the user's choice of computational algorithm to the number of operations that can be achieved if the architecture is utilized optimally. It is possible, however, to include in the value of k all manner of delays that result from the implementation of vector operations.

The third ratio, g, is the fraction of data, $m\delta\ell/\ell$, where $\delta\ell$ is, for convenience, a fixed block of data that must be moved m times at a rate r_T to complete the computation of ℓ units of data. Unless this data movement between primary and secondary memory is masked by a carefully designed data mapping, there is an inherent delay. Again for convenience, the time to seek the data in the backing store is not exhibited explicitly but is reflected in the value assigned to g, which lies in the interval $0 \leq g \leq 1$.

The two other ratios, α and β, characterize properties of the hardware. Define $\alpha = r_s/r_p$ and $\beta = r_T/r_p$. The value of α is in the interval $0 < \alpha \leq 1$. Generally, the value of β is in the interval $\alpha \leq \beta < \infty$. All rates are in units of data/second.

A final ratio, $\gamma = r/r_p$, is the dimensionless efficiency measure that is derived in the next section. The value of γ is in the interval $\alpha \leq \gamma \leq 1$.

3.0 MULTI-RATE EFFICIENCY MEASURE

A given computer architecture can be analyzed from the single user viewpoint as a multi-rate process. The user's program is determined to contain

$$s + p = \ell \tag{1}$$

operations or units of data. The fraction of the total number of operations or units of data that are amenable to speedup by user's exploitation of the architecture is

(2) $$\frac{p}{\ell} = 1 - \frac{s}{\ell}$$

and is the challenge to the numerical analyst who desires to utilize the facility and the computer architect who designs it.

The time required to complete the user's program is, therefore,

(3) $$\frac{\ell}{r} = \frac{s}{r_s} + \frac{p}{k r_p} + \frac{m\delta\ell}{r_T}$$

Rearranging with the aid of (2), and the terms defined in the preceding section obtain

(4) $$\gamma = \frac{1}{\frac{f}{\alpha} + \frac{1-f}{k} + \frac{g}{\beta}}$$

Representative values of γ are tabulated in Table I and the limiting cases are examined next.

4.0 LIMITING CASES

The four limiting cases are examined below for given finite values of α, β and k.

Case 1) $f = 0$, $g = 0$

If no part of the user's program is sequential, and data movement is completely overlapped, then (4) gives

$$\gamma = k$$

or $r = k\,r_p$. The effective rate of computation lies with the computer designer to optimize a specific application with the hardware so that $k \rightarrow 1$.

Case 2) $f = 1$, $g = 0$

If 100% of the user's program is sequential, but data movement is fully overlapped, then

$$\gamma = \alpha$$

or $r = r_s$, as expected.

Case 3) $f = 0$, $g = 1$

Again, if no part of the user's program is sequential, but data is not overlapped, the effective rate of computation is given by

$$\gamma = \frac{1}{\frac{1}{k} + \frac{1}{\beta}}$$

or $$r = \frac{r_T/r_p}{1+(r_T/k r_p)}\, r_p$$

For the ideal problem, $k = 1$. Then the effective rate approaches r_p as β becomes large, i.e. $r_T >> r_p$. Suppose $k \neq 1$, then r_T must be still greater than r_p.

Case 4) $f = 1,\ g = 1$

Finally if all of the user's program is sequential and data movement cannot be overlapped,

then $$\gamma = \frac{1}{\frac{1}{\alpha} + \frac{1}{\beta}}$$

or $$r = \frac{1}{1 + r_s/r_T}\, r_s$$

The effective rate of computation is less than the sequential rate, as would be expected of sequential processing with a two level memory.

5.0 EVALUATION OF PARALLEL OR PIPELINE ARCHITECTURES

The economic side of computer design suggests that a two or three level memory is inevitable for large scale computation. With the advent of electronic rotating memories to fill the gap between physical rotating memories and random access high speed memories, a rate $r_T = r_p$ is a conservative assumption. Also the delay in seeking a block of data in a level two memory is ignored in the following so that the g of a computation is determined by the specific data mapping that is required to accommodate a small level one memory. It is assumed further that more than 50% of the movement of data can be overlapped with the computation.

In Figure 1, a summary plot of γ as a function of f illustrates the impact of small amounts of sequential operations for various representative values of β and k derived from experience with the ILLIAC IV and other machines in this class. For the familiar example of a dual rate machine, the CDC 7600, $\alpha \simeq 0.2$; the use of a vector function library can produce values of k very near 1. The ILLIAC IV, on the other hand has a $\alpha \simeq 0.02$; some algorithms, though carefully programmed for maximum parallelism, realize only a k proportional to $\log_2 n/n$ or $k = 0.1$.

New designs can be readily assessed with this measure, or possibly a more refined measure to eliminate some of the assumptions such as fixed block size. A given computer design is cast into the simplest functional blocks, see Figure 2, and the efficiency measure, γ, determined for representative problems. A systematic comparison study of current ILLIAC IV class machines is underway.

6.0 THE TANDEM SEQUENTIAL-PARALLEL SYSTEM ARCHITECTURE

The severe degradation of effective computation rate due to small amounts of sequential operations, that is less than 20% of maximum rate, suggests that a sequential processor coupled tightly to a parallel unit will be of limited value, whereas a loosely coupled system of scalar and vector processors, scheduled and operated independently for the most part, will be the most efficient. Figure 3 illustrates a tandem system wherein the user who is accustomed to sequential processing interfaces only the processor labelled S. Vector and matrix operations are possible by a direct link to the processor labelled P by issuing subprogram calls from a running process on S. The subprograms and system programs are prepared by specialists.

The highly parallel programs, those with more than 80% parallel operations, may enter directly the second stage of the tandem and use the first stage only for pre/post-processing, again by subprogram calls to S. The efficiency measure does apply to this type of processing. Delays in moving data between S and P will be larger but here a compromise is clearly of benefit for while files are being staged at S or P, the processors can be made available to other users. Clearly, this is not a new idea. Programming languages such as CFD have attempted to provide this sense of machine independence. Ultimately this may lead to the most efficient use of the NASF. In the meantime, the ubiquitous FORTRAN language modified to accept vector and matrix subprogram calls that excite companion processes in the hard-to-use hardware, that are transparent to the user, appears to be the most expeditious route to efficient hardware utilization.

k = 1.0 β = 1.0

	f	g	k
α = 0.001	0.05	0.0	0.0196
	0.05	0.5	0.0194
	0.1	0.0	0.00991
	0.1	0.5	0.00986
α = 0.01	0.05	0.0	0.168
	0.05	0.5	0.155
	0.1	0.0	0.0917
	0.1	0.5	0.0877

k = 0.1 β = 1.0

	f	g	k
α = 0.001	0.05	0.0	0.0168
	0.05	0.5	0.0167
	0.1	0.0	0.00917
	0.1	0.5	0.00913
α = 0.01	0.05	0.0	0.0690
	0.05	0.5	0.0667
	0.1	0.0	0.0526
	0.1	0.5	0.0513

k = 1.0 β = 10.0

	f	g	k
α = 0.001	0.05	0.0	0.0196
	0.05	0.5	0.0196
	0.1	0.0	0.00991
	0.1	0.5	0.00990
α = 0.01	0.05	0.0	0.168
	0.05	0.5	0.167
	0.1	0.0	0.0917
	0.1	0.5	0.0913

k = 0.1 β = 10.0

	f	g	k
α = 0.001	0.05	0.0	0.0168
	0.05	0.5	0.0168
	0.1	0.0	0.00917
	0.1	0.5	0.00917
α = 0.01	0.05	0.0	0.0690
	0.05	0.5	0.0687
	0.1	0.0	0.0526
	0.1	0.5	0.0525

TABLE I

REPRESENTATIVE VALUES OF EFFICIENCY MEASURE, γ

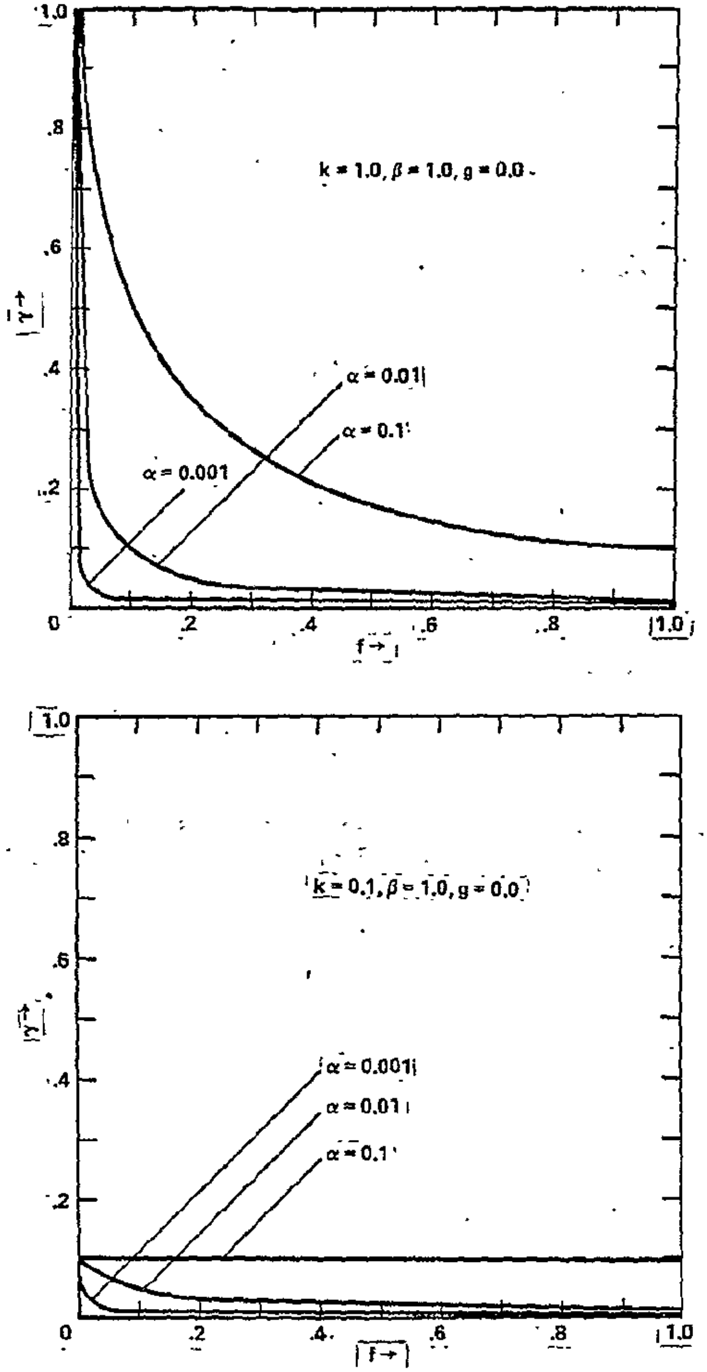


Figure 1. Efficiency measure, γ, vs fraction f for k = 1.0, 0.1

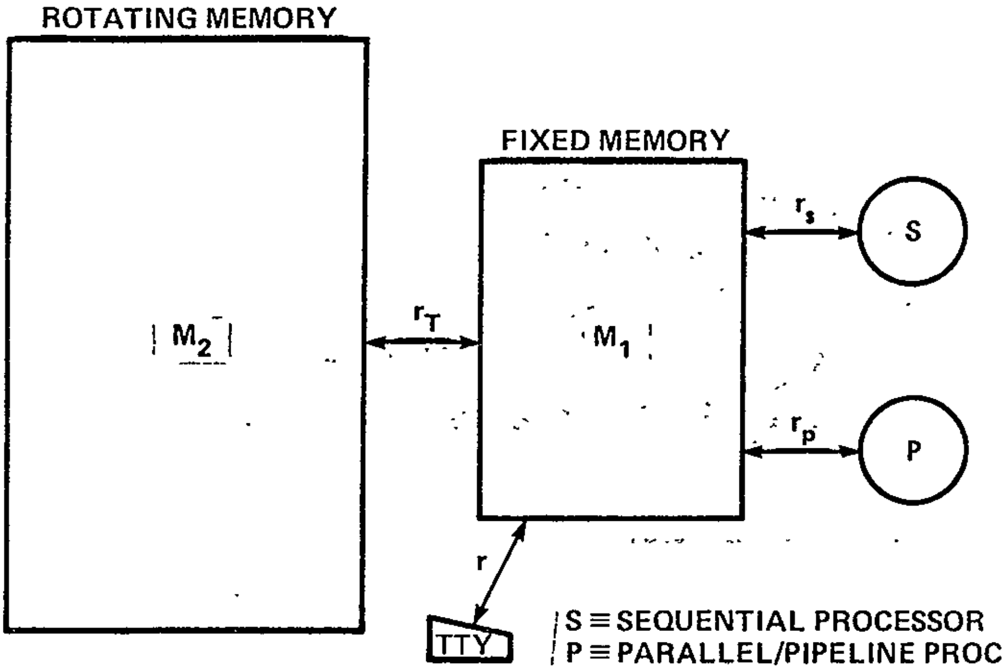


Figure 2. Tightly coupled system

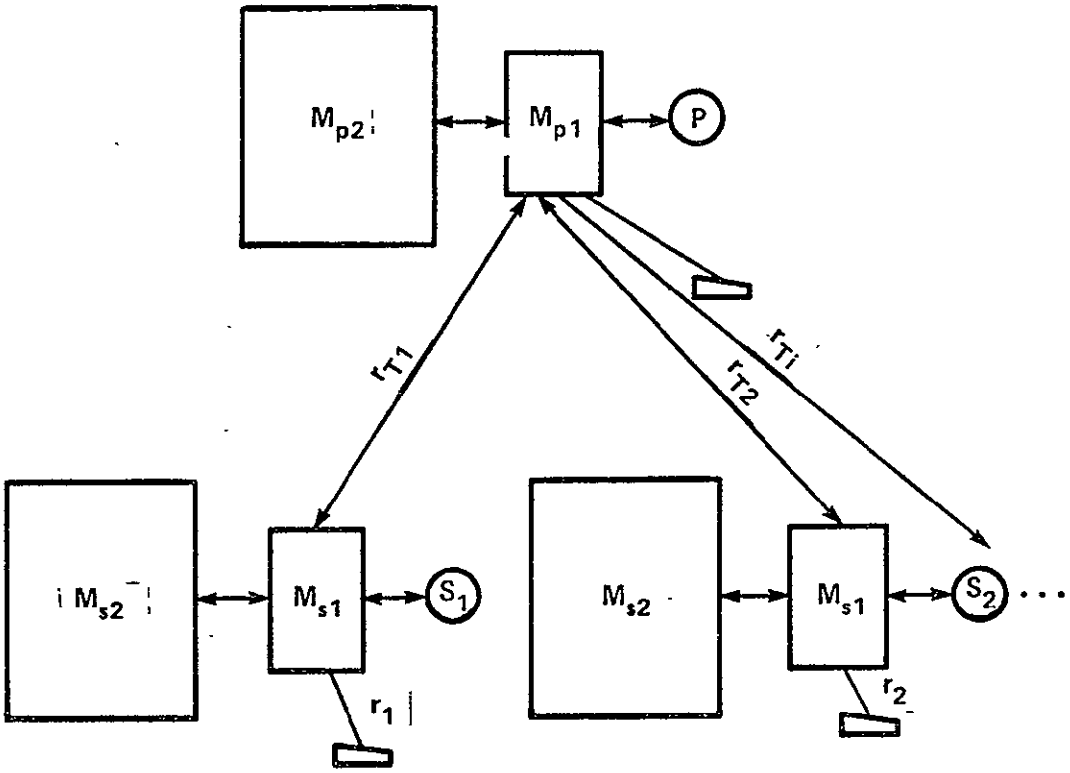


Figure 3. Tandem processing

D31

THE INDIRECT BINARY N-CUBE ARRAY

by

Marshall Pease
Staff Scientist
SRI International
(formerly Stanford Research Institute)
Menlo Park, California

Abstract:

A design for a high-performance computational array is proposed. The array is built from a large number (hundreds or thousands) of microprocessors or microcomputers linked through a switching network into what we call an "indirect binary n-cube array." Control is two-level, the array operating synchronously, or in lock step, at the higher level, and with the broadcast commands being locally interpretted into re-writable microinstruction streams in the microprocessors and in the switch control units.

The design is suitable for a large number of problem types. Study has been made of its suitablity for parallel computations over grids of various configurations in two, three, or more, dimensions and with various sizes in the different dimensions. Its use in matrix and vector operations, including matrix inversion, has been studied in detail. Its application to the FFT and other decomposable transforms has been studied, and to sorting and related tasks. It has been found that the design is suitable for these processes, and that the high parallelism of the array can be utilized fully with suitable choice of the algorithm.

The key to the design is the switching array. By properly programming it, the array can be made into a wide variety of "virtual" arrays which are well adapted to a wide range of applications. While not yet studied in detail, it is believed that the flexibility of the switching array can be used to obtain fault-avoidance, which appears necessary in any highly parallel design.

The use of a switching array, rather than a fixed set of interconnection paths, can be expected to increase the cost of the system by an amount that is not severe. In return, a much wider range of applications, and of algorithms for a given application, can be handled. In addition, it becomes relatively easy to double the size of the array at any time, allowing for its incremental growth. The use of a switched array, and of the indirect binary n-cube array in particular, appears attractive.

The work reported here was supported by the National Science Foundation under Grant GJ-42696.

I. INTRODUCTION.

In this paper, we present a possible design for a highly parallel computational facility using a large number of microprocessors or microcomputers. The feasibility and need for such a facility does not need to be argued here. It is our contention, however, that the architectural principles that should be used have not been unambiguously established, and that there is need for continued study of alternative approaches.

The need being addressed here is for a machine that will handle the equations of fluid dynamics in three dimensions under various boundary conditions. The principal application is the simulation of wind tunnel measurements, although other important application areas exist. A high degree of parallelism is needed because of amount of data that must be processed, and the number of iterations that are needed.

Parallelism, in the broad sense, includes pipelining and the use of combinatorial units for various arithmetic and logic functions. The particular types of problems addressed here, however, are strongly iterative in both time and space. It seems intuitively desirable to make use of this property, employing a design that reflects the geometry of the problem. We can visualize a two or three dimensional array of units, each of which is capable of performing the complete cycle of calculations at a point. We do not exclude the possibility of pipelining or other techniques within the units, but we see the central problem as that of organizing the computational units into an integrated array.

Whether the computational units should be microprocessors or microcomputers-- i.e., whether each unit should contain its own memory or not-- is a separate issue that largely depends on the economics of memory technology. If the units are microcomputers and do contain significant working memory, additional backup memory will certainly be required. If they are microprocessors, they will still need internal registers. The question, therefore, is not whether, but how much memory should be included in the units. While acknowledging the significance of this problem, we will not address it here. We will use the term "microprocessor" indiscriminately, without regard for the amount or kind of memory it may contain.

The critical issue, as we see it, is to obtain the required communication among the microprocessors. If this is obtained through an intermediate set of working memories, the problem is still one of making certain that each microprocessor has the necessary sets of data when it needs them. The nature of the computational processes requires a tremendous amount of data transfer. Some data must be transferred into and out of each microprocessor prior to, or during, each iteration. To use the array efficiently, a very large inter-microprocessor bandwidth must be provided.

The obvious solution to the bandwidth problem is to provide direct inter-microprocessor lines that will link the entire set into a grid that is more or less identical with the computational grid. The method of approach can be modified to accommodate the interleaving process that is commonly used in fluid-dynamic problems. However, in this approach, the array is made to correspond, directly and physically, to the computational grid, probably a rectangular grid in two or three dimensions.

We contend, however, that this approach is unnecessarily limiting. There is a different method of obtaining the required communication that achieves much the same effect without serious sacrifice of cost or simplicity, and that permits a flexible choice of the array's apparent configuration.

We argue that flexibility in the array connections is highly desirable, providing it can be achieved without serious sacrifice, for several reasons. First, even given a particular type of application and a particular algorithm, there will arise the need for different grid sizes. We will want to be able to use the available parallelism in different ways. Second, new algorithms will be developed for the given application, and it is undesirable that the design of the array should limit what algorithms can be considered. Third, other application areas exist or will arise which need a comparable facility, but may require a quite different configuration. Since we cannot know exactly what will be needed for these future uses, it is desirable to provide as much flexibility as is feasible.

We propose the use of a switching network to provide the high inter-microprocessor bandwidth required without having to freeze the communication patterns of the array. The penalty of this approach is the cost of the network itself, plus programming complications introduced by the delay in the network. While a full cost analysis has not been done, it is believed that the additional cost need not be great compared to the cost of the array itself, and that the other penalties are also relatively insignificant.

In the next section, we describe a particular type of switching network that seems particularly attractive, which makes the array into what we call the "indirect binary n-cube array." We are not proposing a particular logical design for this network; there are many variations that are possible, and the selection of a particular design should be made only after detailed cost and performance analyses based on particular technologies. It is the general type of switching network that interests us.

In the following section, we describe the general method of control for the network that we envision, and discuss how it can be integrated into a complete system. The proposed control system allows establishing a set of "virtual arrays" each of which can be established by a single command. The array then looks like a particular set of connections, such as a right-shift connection in a rectangular array with particular dimensions. The concept provides the simplicity of a hard-wired set of connections, but with the option of changing the connection patterns as required.

II. THE SWITCHING NETWORK AND THE INDIRECT BINARY N-CUBE ARRAY

An example of the type of switching network that we find most attractive is shown in Figure 1. The circles represent the microprocessors, labelled from 0 through 15, or, more generally, from 0 through (2^n -1). The boxes represent elemental switches, or "switch nodes", that can be put in either of two states, direct or crossed, as indicated in Figure 2. Flow through the network is from left to right, as indicated by the arrows. The numbers in parentheses on the right indicate how the lines are connected back to the microprocessors.

The design shown in Figure 1 assumes that the microprocessors have sufficient memory so that most calculations can be executed within them without addressing external memory. An alternate design uses two such networks to connect the microprocessors to and from a set of independent memories.

The detailed properties of this network, as well as its abstract definition, have been discussed elsewhere [1]. Lawrie [2] has described a similar network which he calls an "omega network" and has described some of its properties. Here, we will state without proof some of its more relevant features.

As may be seen from Figure 1, the switch nodes, the boxes of Figure 1, are arranged in a sequence of levels, labelled S1, S2, S3 and S4 in Figure 1. In general, with 2^n microprocessors, there are n levels of switch nodes. If the microprocessors are conceived as being at the vertexes of an n-cube, or hypercube in n dimensions, each switch node, when crossed, causes the interchange of data along one edge of the n-cube. Each edge is represented by one switch node, and the nodes at a given level correspond to a set of parallel edges. It is these properties that have led to the name of the array, "the indirect binary n-cube array."

The representation of Figure 1 is not meant to imply the actual structure of the switching network, and particularly not its partition into chips. Nothing is implied, either, about the bandwidth of the lines between switch nodes. These decisions require detailed performance and cost-tradeoff studies that have not been made. Figure 1 should be regarded as a functional diagram, rather than a design.

There are two other factors that may need to be considered in an actual design. First, there is a great deal of symmetry in the connections shown in Figure 1. This can be used if it is desired to build the array incrementally. The array of Figure 1, for example, can be doubled in size by replicating it, and then adding a single additional level of switch nodes on the right. If incremental growth is important, the necessary symmetries should be retained in partitioning the network among chips.

The second consideration is that one might wish to include other capabilities in the switch nodes. Since a preliminary design study of the switch nodes suggests that pin limitations are almost certain to be dominant, it appears feasible to do so. One capability that is likely to be desirable is for latching, so that a switch node can be controlled by the data on its input lines. This would permit use of a version of Batcher's bitonic sorting algorithm for sorting and generating arbitrary permutations. Other capabilities can also be considered.

The operation of the switching network, as it is shown in Figure 1, is described in terms of what we call a "unit transfer." By this is meant passing data once through the network. In a unit transfer, each microprocessor can transmit one byte out, and receive one byte, where a byte is defined by the width of the lines in Figure 1. If the array contains 2^n microprocessors, and a byte is m bits, the total bandwidth is m(2^n)/t, where t is the delay time of the network. All communication between microprocessors is via unit transfers.

It is not asserted that a unit transfer is necessarily trivial. If the array is large, there are many levels and a significant delay can accumulate. However, a unit transfer is the smallest communication process that exists. Further, the delay associated with a unit transfer is constant, so that compensation for it can be programmed.

The key question is what communication patterns can be obtained by unit transfers. This question is considered in detail, and an analytic answer obtained, in reference [1]. We have found that all the communication patterns required for handling partial differential eqations over the commonly used grids are obtainable as unit transfers if the different dimensions of the grid are powers of two.

A study has also been made of matrix operations, including both matrix multiplication and inversion. Algorithms have been developed for matrices whose sizes are compatible with the number of microprocessors, which use the parallelism efficiently, and which require only unit transfers.

It appears that a switching array of the type illustrated in Figure 1 is suitable for the applications and algorithms being considered here.

The use of a switched array does involve some additional cost when compared to a hard-wired array. The number of switch nodes for 2^n microprocessors is $n2^{(n - 1)}$, which is large if n is large. The number of chips may be considerably smaller, depending on the byte size and how the network is partitioned, but will still be large. However, the chips will be relatively simple in design. It is expected that they will be relatively cheap, compared to the microprocessor chips. The additional cost may be relatively minor.

The delay through the switching network is also a factor if n is large. Since the delay between chips is likely to be much larger than the delay within a chip, the amount of the delay depends not only on the technology used, but also on how the network is partitioned. However, as long as we can depend on needing only unit transfers, the delay is fixed and predictable, so that compensation for it can be built into the program.

The major advantage obtained is flexibility. The network can be programmed to execute, as a unit transfer, a wide range of data transfer patterns. It can be said, in fact, that the network has been found capable of executing all of the transfer patterns that are required for all the algorithms that we have considered of likely importance for such an array.

III. CONTROL

The general type of control system that we have envisioned for the array is indicated in Figure 3. It is a two level system. Top level control is exercised by the box labelled "controller" at the top. This unit issues broadcast commands to the microprocessors and to a set of switch controllers. At this level, the array operates in "lock-step."

At the second level, each microprocessor interprets a given global command into a sequence of micro-instructions. The sequence may be different in different microprocessors, depending, for example, on whether it is handling a boundary point or an interior one. In a single microprocessor, a given command may be differently interpretted at different times, depending on a previous test of the data. This permits a microprocessor to execute different computations according to the physical regime that is involved. It is assumed that the microprograms are rewritable so that appropriate changes can be entered as part of the initialization for a run.

The switch controllers also accept the global command and interpret it as sequence of control bits for the switch nodes. The switch controllers need be little if any more than a read-only or write-occasionally memory.

As seen from the controller, the switching array appears to have only those transfer modes that have been established by the codes stored in their memories. The controller calls any one of those modes by a simple global command. The controller sees the switching array as implementing a specific grid, say a (2^p) x (2^q) rectangular array (where p + q = n), and understands its own commands as calling for a unit shift in this array, right, left, up or down. The codes stored in switch controllers establish this virtual array.

If other shifts in the virtual rectangular array are needed, such as a diagonal shift to implement an interleaving process, they can be added by appropriate entries to the switch controllers. If a different virtual array is required, such as one of those convenient for matrix inversion, it can be established by reloading the switch controllers with the appropriate codes.

The details of these manipulations of the switching network for many of the desirable communication patterns have been worked out and are given in reference [1]. It is sufficient, here, to say that they are known and can be implemented. The proposed switching network is a very flexible one, and the control scheme outlined allows using the flexibility in a way that is convenient for programming.

IV. CONCLUSIONS

It seems evident that any computational facility such as that considered here will be a limited-purpose one. Only certain algorithms can make efficient use of the high parallelism that is envisioned. Further, the nature of the relevant algorithms imposes a critical requirement for inter-microprocessor communications that is likely to force a design which is unsuitable for many purposes. The fact that we are forced to use limited-purpose designs makes it more important to seek to reduce the limitation as far as is feasible.

The use of a switching network to provide the array inter-connections leads to a design which has great flexibility with minimal compromise of cost or performance.

In particular, the proposed network, which creates the indirect binary n-cube array, seems a particularly attractive candidate. It has all the flexibility that is likely to be needed. Its cost remains to be evaluated, but seems unlikely to be excessive. It adds delay, but the delay is fixed and can be handled in the programming.

References:

[1] M. C. Pease III, "The Indirect Binary N-Cube Microprocessor Array," IEEE Trans. Comput. Vol C-26, pp 458-473, May 1977.
[2] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," IEEE Trans. Comput. Vol C-24, pp 1145-1155, Dec. 1975.

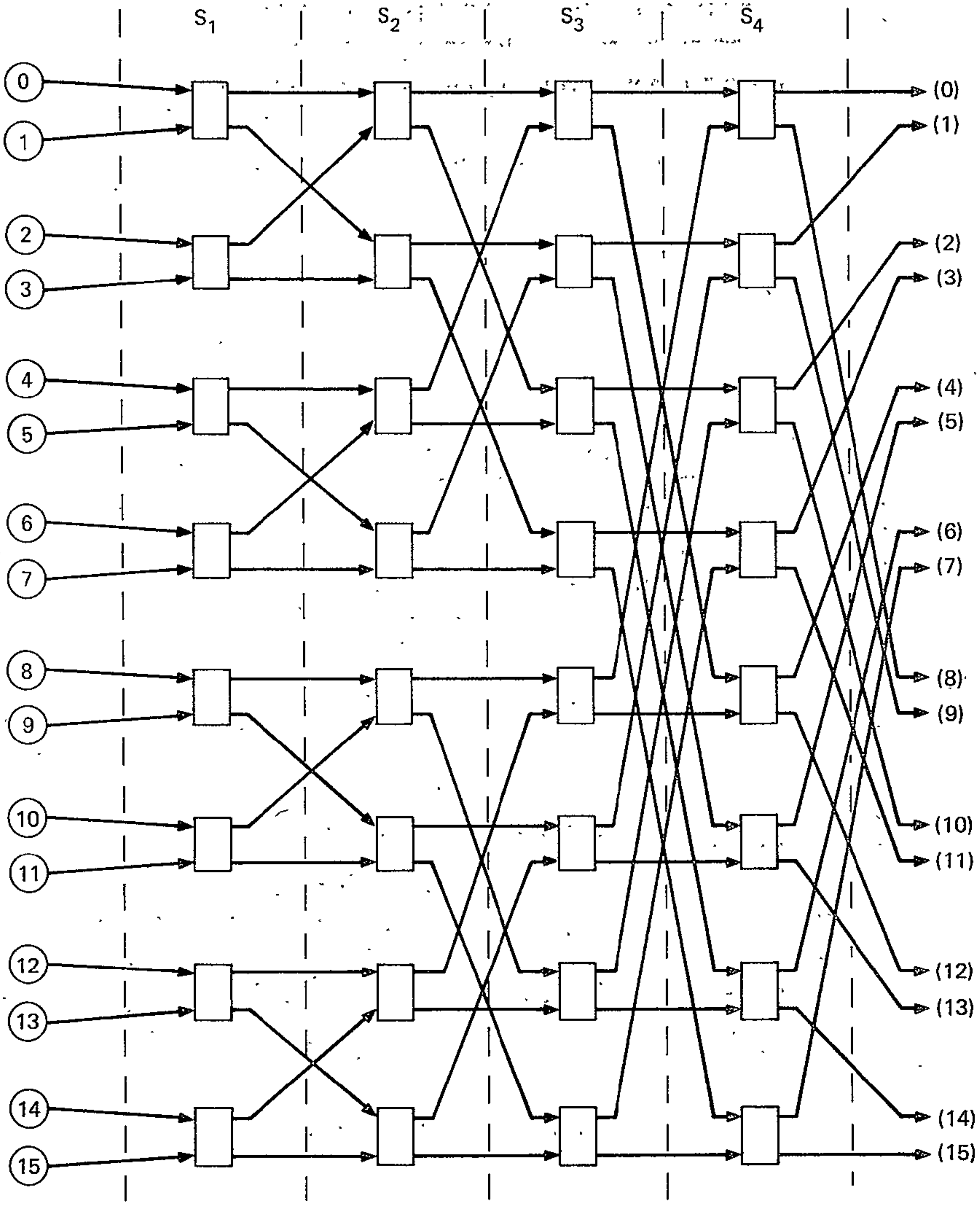


FIGURE 1 THE INDIRECT BINARY 4-CUBE ARRAY

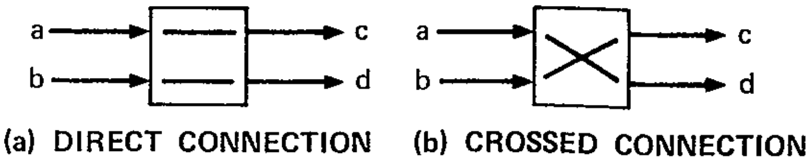


FIGURE 2 THE SWITCH NODE

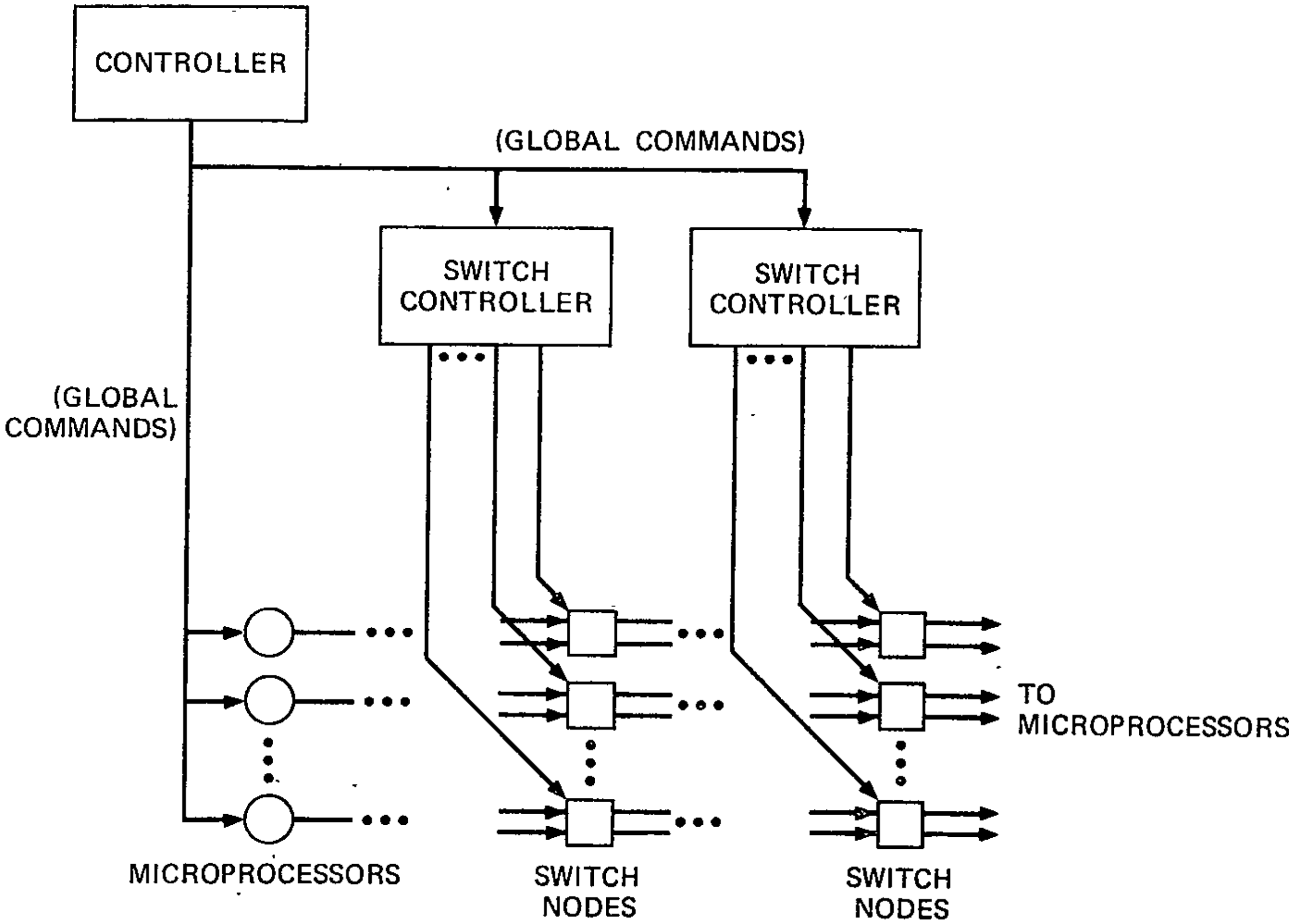


FIGURE 3 OUTLINE OF CONTROL SYSTEM

Methodology of Modeling and Measuring Computer Architectures for Plasma Simulations

Li-ping Thomas Wang

Center for Plasma Physics and Fusion Engineering

University of California

Los Angeles, California 90024

ABSTRACT

Computer simulation in plasma physics has evolved to be a very promising field during the past decade and its results can check against physical theories and experiments in a more integrated point of view. However, it is demanded that a more capable and much faster computing system be needed to help understand plasmas and to pursue satisfactory precision. In the first part of this paper a brief introduction to plasma simulation using computers and the difficulties on currently available computers is given. Through the use of an analyzing and measuring methodology - SARA, the control flow and data flow of a particle simulation model REM2-1/2D are exemplified. After recursive refinements the total execution time may be greatly shortened and a fully parallel data flow can be obtained. From this data flow, a matched computer architecture or organization could be configured to achieve the computation bound of an application problem. In this paper a sequential -type simulation model, an array/pipeline-type simulation model, and a fully parallel simulation model of a code REM2-1/2D are proposed and analyzed. It is found this methodology can be applied to other application problems which have implicitly parallel nature.

The study of plasma physics and fusion technology is considered to be one of the most complicated sciences in the world, although it began as a science about fifty years ago. A plasma is a quasineutral gas of ionized and neutral particles at a very high temperature. When two lighter nuclei approach one another with sufficient speed to overcome their electrostatic repulsion, a collision occurs which may produce another heavier nuclei and release fusion energy. Due to the very high temperature and the instability of the plasma itself, people still do not have full confidence in the success of a large scale fusion reactor. However, in addition to conventional theoretical and experimental approaches, another method was developed to help understand the behavior of plasmas -- computer simulation[1,2]. By using computers, a plasma can be normalized and its behavior can be simulated; also the numerical results can be checked against the theories and experiments. Computer simulation has already made very significant contributions since the past fifteen years[3]; nevertheless, the existing computing tools are virtually not satisfactory enough to most people who are involved in this promising field. In this paper the difficulties in plasma simulation are to be reviewed and a methodology of modeling and measuring suitable computer architectures is to be proposed.

COMPUTER SIMULATION OF PLASMAS

Because of the long range nature of electric and magnetic forces between charged particles, plasmas exhibit what are called collective motions which many particles act in coherent fashion. Over about twenty-five years our direct experience with plasma is still very limited and its behavior has proved to be complex, and probably much more complex than anticipated a decade ago. Therefore, one flexible, economical and fundamental method for trying to get some more understanding of plasmas is through numerical modeling. Fortunately, the computer simulation now appears to be the most powerful method for understanding plasmas and their confinements.

1. Finite-Size Particles

Computer simulations of plasma using particles has evolved during the past decade from point-particle model through line sheet, to the so-called finite-size particle (FSP) model.[2] In the FSP method, the finite-size particles or extended particles, instead of points of particles, are used to play a very important role in the simulation. Such extended charged particles interact via Coulomb forces when they are separated by large distances, but the force falls off to zero as they interpenetrate each other. By using FSP scheme, the total number of simulated particles, and thus calculation time, can be greatly reduced. FSP simulation model now proves itself to be very time-saving and its results are in good agreement with theories; therefore, it now becomes the most popular method in plasma simulation.

2. Mesh Background

In a system of plasma simulation, the region can be divided into many grid points which are uniformly equal-spaced. Present methods convert the charge positions into charge densities associated with each grid point and then solve for the field at each grid point.[2] Thus, the field and then the force on the particle is obtained by suitable algorithms from the fields at nearby grid points. Quite a few algorithms have been studied and developed in the past such as

Nearest Grid Point [NGP], Multipole Expansión, and Subtracted Dipole Scheme [SUDS]. The time required to compute the fields for M x M grid points is proportional to MlnM if Cooley and Tukey's Fast Fourier Transform (FFT) is employed. Generally, the number of particles is much greater than the mesh size M and then the force calculation is much quicker than that for point interacting charges.

3. Time Steps

Digital computers have offered both fast computing speed and precise floating-point calculation to plasma simulation in those years and made it a very promising field[3,4]. However, partially due to the discrete characteristics of digital computers which are available nowadays, a plasma is simulated step by step in simulation time scale, viz., causally in time. Also particles are processed or pushed by the uniprocessor in a one-by-one manner. The simulation usually terminates when it is considered to be long and equivalent to an observation period long enough in the experiment time scale. Consequently, larger numbers of simulated particles and larger number of grid points are bound to spend longer execution time on a conventionally sequential computer, a run of more time steps will certainly cost more money.

4. Behavior of Plasmas

The property or behavior of a plasma is primarily represented by a group of charged particles. Initial condition of a simulated plasma system can be made by placing those particles at certain grid locations according to their corresponding distribution in space, and giving them certain associated velocity according to their corresponding velocity distribution. Particle locations and velocities vary with electrical and magnetic fields, which vary with particle locations and velocities in a later time step; then a basic loop occurs and proceeds over and over. The algorithm which governs the particles usually consists of Maxwell's equations and Newton-Lorentz's equation of motion, all in the finite-difference form. The size and boundary of a mesh are important to the behavior of a plasma because the former resolves the plasma particles and the latter confines the simulation system. The behavior of a plasma is abstracted from following the movement of these particles and diagnosing the fields associated with the grid points. Some of the fields are kept on record for post-processing and display in order to examine the microscopic behavior of a plasma, such as the dispersion relation correlation of waves, power spectrum, etc.

DIFFICULTIES IN COMPUTER SIMULATION OF PLASMAS

There is hardly any branch of physics today that has not made use of computers in some form or other. It can be truly said that there has been a decisive impact of computers on plasma physics, yet there have always been problems which could progress no further because of the lack of suitable computer systems and the too general purpose design of most of today's computers. The lack of suitable computer systems makes some of the two-dimensional and most of the three-dimensional simulations out of the question[4], while the too general purpose design of today's large-scaled computers makes the simulation experiments very slow and therefore, very expensive. As one physicist said, "It is not surprising that the situation at the present time does not in any fundamental sense differ from that of the past. One might say it is more clear that we are now more aware of the role computers can play in physics and we can identify problems that would be solved if only our computer systems were not so limited".[5] The finite capacity of memory, slow execution rate of uniprocessor, unmatched data transfer rates between memory hierarchies, intolerable machine vulnerability and non-real-time control really make the growth rate of plasma simulation lag with respect to what should otherwise be expected. Past experience shows that further analysis and measurement of the nature of existing simulation models is urgently needed, in order to obtain more stringent requirements of a future computer system which would be better suited to plasma simulation. In the following we list some of the major difficulties with which people in plasma simulation have been confronted during the past years:

1. Memory Space

One particle has one position component and three velocity components, in total four memory words in a 1-2/2 D code; or two position components and three velocity components, in total five momory words in a 2-1/2 D code; or three position components and three velocity components, in total six memory words in a 3 D code. Number of field variables depends on the type of code: electrostatic, magnetostatic, or electromagnetic. The total number of memory words of the fields depends on the type of the code, as well as the the size of mesh. However, the total number of memory words of particles is proportional to the number of particles. For example, a 2-1/2 D relativistic electromagnetic (REM 2-1/2 D) code with 10^6 particles and a 128 x 128 mesh may occupy (5 + 1) x 10^6 + 10 x (128 x 128) = 6 x 10^6 + 163840 = 6,163,840 memory words if one extra word for relativistic factor for each particle is needed and there are 10 field variables with the same mesh size. Most of today's available computers cannot afford such large memories, although a code may usually occupy more than this figure.

2. Execution Time

Figure 1 shows roughly the CPU times which will be spent for typical runs of a 3D particle code and a 3D fluid code[4], which simulates a plasma by using fluid-like equations instead.

3. Multi-Run of Simulation Codes

From Figure 1 it is surprising that for a single run which is barely enough for one laboratory experiment, we need the complete dedication of an entire week of the CPU time. Investigators generally need a series of experiments. Furthermore, serious research generally needs a series of experiments concurrently with only one parameter varied, which is denoted as "multi-run" of simulation codes. Apparently today's non-multi-run experiment on a single processor and its intolerably long running time leave the computer simulation proponents in a very embarrassed and uneasy situation. A computer network, composed of either super-computers or microcomputers, may probably solve this problem.

4. Post-Processing Problem

After the simulation run terminates, usually a bulk of historical information of field variables is recorded, time step by time step, on a secondary memory device. This information is kept for diagnostic use, such as checks for dispersion relation, correlation of waves, and power spectrum of wave modes. In this post-processing task at least two problems arise: the need of a huge memory storage and the lack of adequate displaying tools. Volumes of historical information have to be stored on secondary memory devices if there is no space left for them on primary memory devices. The need of an adequate displaying tool would be very urgent should a careful microscopic diagnosis be required. In case a real-time control of experimental plasma is demanded, the post-processing problem would become more significant than batch tasks.

FLOW OF CONTROL

Figure 2 shows the sequential flow of control of a typical 2-1/2 D relativistic electromagnetic model (REM2-1/2D) which has been used for the study of plasma effect on synchrotron radiation at UCLA. At first, all the particles are placed uniformly on the grid and their velocities are normally distributed. Then the basic major loop begins from advancing the particles half of the distance that they should be pushed in one time step in order to calculate the current density on the grid. A second particle advancing for the charge density calculation is followed a half time step later. Then the control switches into Fourier space by taking Fourier analysis of the current and charge source fields. In Fourier space the transverse electric and magnetic fields are updated as described by Maxwell's equations, which includes Poisson's equation, and the system diagnosis such as energy conservation is made. (These microscopic diagnostics and measuring are very crucial to a fundamental plasma simulation system.) The transverse fields are then transformed back to real space in order to calculate the new velocity of each particle according to Newton-Lorentz's equation of motion, and after all particles are updated the control flows back to the beginning of the major loop for another time of particle advance. Only if the termination condition is satisfied, the major loop ends and the post processing starts.

FLOW OF DATA

Fig. 3 is the flow of data depicted in UCLA's GMB (Graph Model of Behavior) form and it is associated with the control flow shown in Figure 2. In this figure, it may be clear how data flows in each control node or flows out of it, so that the data dependency on the control flow path can also be determined very easily. (Although in a sequential control flow there is no accessing conflict, it may happen in a parallel GMB flow of data graph.) From these two graph models and their associated time delays the execution time of a basic loop of REM2-1/2D code can easily be calculated as follows: (refer to Fig.2)

$$T_{loop} = [(t_2 + t_3 + t_4 + t_5 + t_{10}) \times N + (t_6 + t_7 + t_8 + t_9)] \times NEND$$

RECURSIVE REFINEMENT

The sequential flow of data of Fig. 3 and the time calculated from above explain why the sequential flow of control of Fig. 2 is not a satisfactory simulation model. It can be found that there are many data independencies in the flow of data graph which can be further improved to get another flow of data graph with shorter execution time. After certain times of iterative modifications we may come up with a data graph and its associated control graph shown in Fig. 7 and Fig. 6. The total execution time of a basic loop of the same REM2-1/2D code can be calculated now as follows: (refer to Fig.8)

$$T_{loop} = (t_{VU} + t_{CA} + t_{FFT} + t_{FU} + t_{IFFT}) \times NEND$$

MODELING AND MEASURING METHODOLOGY - UCLA's SARA SYSTEM

A few plasma simulation proponents have made attempts of modeling their application problems recently on different types of advanced computing systems, such as the ILLIAC IV[5,8], STARAN, ASC, CDC STAR-100, CRAY-1, and CHI AP-90[7]. The ILLIAC IV is an array-type computer with 64 parallel processors, while CHI AP-90 is a highly overlapped computer with two pipelines: adder and multiplier. Both of the two modern computers offered better measures than that of a sequential computer, but not very significant. The key issue is that there are varieties of arithmetic operations involved in the simulation codes. The part which is well fitted to the particular feature of that computer is usually a small fraction and thus most of the arithmetics are suffering instead. For instance, a solo IF and GOTO statement in one of the 64 parallel processors causes a shut-down of all other 63 and is definitely a painful waste. Accordingly, in this paper we are not aiming to propose a best computer system for solving the above mentioned difficulties, since the criteria for the "best" computer system have not been set up yet, and it is not easy to do so. In a more practical manner, we introduce a useful modeling and measuring methodology - UCLA's SARA (System ARchitects Apprentice) [9-11], to formalize the intended behavior of a particle simulation model on a certain type of computer. By using GMB[12], now a subsystem of SARA, the control flow and data flow of a simulation model can be properly expressed and associated with a measure by which success of the model and the computer system can be evaluated. (Now the implementation of SARA methodology is still in progress and will be fully ready for use very soon.) The SARA system was designed and developed to decrease the gap between intent and behavior of a digital system[9]. It allows multilevel system design in order to manage complexity through a refinement process. It also provides the computer-processable tools for separating structure from associated behavior in a synthesis model. The control flow graph and data flow graph are two useful methods in GMB which we borrow here to analyze and measure our simulation models.

Sequential Model:

Fig. 2 and Fig. 3 are the control flow and data flow graphs of REM2-1/2D model running on a conventional sequential computer. Here are some remarks which should be pointed out about this sequential model:

1. Particles are uniformly distributed initially but can walk randomly in a later time.
2. Particles are called by their ID numbers (in natural order) and could be distinguished along the simulation.
3. A doubly periodic rectangular mesh is embedded as the background and its resolution should be good enough for the FSP scheme.
4. Particle data and field information are supposed to be in primary memory; if secondary memory is needed there is no time delay assumed in this case.

Array/Pipeline Model:

Fig. 4 and Fig. 5 are the two types of flow on an array-type computer with a limited primary memory, such as the ILLIAC IV. They could also be applied to a pipeline-type computer such as the CHI AP-120B, at this level, although they have differences in the deeper level design and the execution time, associates with the processor in each control node. Several points are made here about this array-type model:

1. Except those which are in operation, all the particles and their associated data are stored in secondary memory (e.g. disk).
2. The subprogram "Velocity Update" is moved up to the first node of the basic loop in order to avoid the second pass of a particle in one loop.
3. A mesh is embedded and its resolution requirement is needed as before.
4. Field information is stored in primary memory all the time.
5. In case primary memory is not large enough, both field information and particle data are stored in second memory and locally moved to primary memory for operations.

For pipeline computers, the remarks for the simulation models are the same as those of an array-type, except the way they are processed. The array computer operates on particle data or field information simultaneously while the pipeline computer does it in a one by one manner, but with a certain degree of overlapping. Therefore the execution time, or delay time in the terminology of SARA-GMB, associated with each control node will be different.

Fully Parallel Model:

Fig. 6 and Fig. 7 are the two flows on a fully parallel type computer (including FFT) although it does not really exist today. From the result shown in Fig. 8, it may be seen that the total execution time for a basic loop is about 0.6 microsecond. That is approximately the lower bound of parallel computation for this REM2-1/2D model. Plus some overheads, system diagnostics, and safety factor it may increase up to 1 microsecond. However, some remarks should be made about this fully parallel model:

1. Particle data are stored individually in a number of processing elements which is equal to the number of particles.
2. A processing element could have many parallel or pipeline ALUs (Arithmetic/ Logic Units) and that number is enough for parallel processing at any instance when parallel computation occurs.
3. A whole set of source fields or E & B forces, which may be assigned or accessed by particles from any positions, should be stored in the memory of each processing element.
4. Random walk in a later time of particle movement is allowed.
5. Particles are called by ID numbers and are distinguishable all along the simulation.
6. A fully parallel plus pipeline hardware of FFT is required.
7. The overall system could be a network of existing computers or microcomputers on chips.

DISCUSSIONS

SARA has several other modeling and measuring features which include TRANSLATOR and SIMULATOR: the former translates the two flows (control and data) in a machine processable form while the latter simulates a token machine through SIMULATOR and an interpreting program PLIP, which interprets the intended behavior of the model.

As shown from the control flow graph of the fully parallel model (Fig. 6) a bottleneck emerges at the fast Fourier transformation of source fields. A dedicated FFT hardware via microprocessors has been proposed[13]; it is found that the computation speed does not mainly come from the electronic circuit but also from the parallel organization. Other proposals with almost the same idea have been made or tested recently such as TRW, MIT, etc.

After the model is properly terminated as tested by SARA, some measurements such as the total execution time can be measured. The graph of data flow could be used as the blueprint for a data-flow computer[14] which would be dedicated and specialized to that particular application with better measures. Of course, a computer system can only be constructed from those building blocks at the bottom level of multilevel system designing.

CONCLUSIONS

Plasma simulation by the use of computers is a very promising field today and as long as energy crisis remains as the first priority problem to be solved in the future, it is demanded that a much faster and more capable computing power be needed to help understand plasmas.

By reviewing the difficulties in conventional computational techniques of plasma simulation, we reveal that more detailed control flow and data flow of a model need to be carefully studied, in order to get an efficient data-flow computer, which is able to provide fully parallel computations.

The processing elements in the fully parallel computer may either be interpreted as a computer network, or a bunch of microcomputers. The fast computing speed does not drastically come from the state-of-the-art electronic circuitry, but from the parallel organization of computers and pipelined arithmetic/logic processors. The function components may not be those chips off the shelf today, however, their manufacturing cost is going down for sure in the next few years. A designing methodology SARA is introduced to help analyze and measure the simulation models in order to get a better design of a future CTR computer. It is found this idea applies not only to plasma simulations, but to all kinds of application problems with implicit parallel nature such as fluid simulations.

REFERENCES

1. Birdsall, C.K., and J.M. Dawson, "Computers and their Role in the Physical Sciences," Fernbach and Traub (Eds.), Chapter 13, Plasma Physics, Gordon and Breach Publisher, 1970.
2. Dawson, J.M., "Computer Simulation of Plasmas", Astrophysics and Space Science, 13, 1971, pp. 446-467.
3. Dawson, J.M., "Contribution of Computer Simulation to Plasma Theory" Computer Physics Communications, 3, Suppl. 1972, pp. 79-85.
4. Dawson, J.M., "Computer Applications In Controlled Thermonuclear Research" Atomic Energy Commission 1974 Controlled Fusion Program Report.
5. Buneman, D., "The Advance from 2D Electrostatic to 3D Electromagnetic Particle Simulation", The Second European Conf. on Computational Physics, Munich, April, 1976.
6. Zacharov, B., "Development of Computer Systems in Physics," Computer Physics Communications, 3, SUPPL., 1972, pp. 50-62.
7. Kamimura, T.; J.M. Dawson, B. Rosen, G.J. Culler, R.O. Levee, and G. Ball, "Plasma Simulation on the CHI Microprocessor System", PPG-248, Plasma Physics Group, Physics Department, University of California, Los Angeles, December, 1975.
8. Miller, R.H., "Design of a Three-Dimensional Self-Gravitating N-Body Calculation for ILLIAC IV," Section II-E, Quarterly Report No. 48, Institute for Computer Research, The Univ. of Chicago, August 1, 1975.
9. Estrin, G., "Modeling for Synthesis -- The Gap Between Intent and Behavior," Proc. of the Symposium on Design Automation and Microprocessors," Palo Alto, California, February, 1977.
10. Gardner, R.I., "State of the Implementation of SARA", Proc. of the Symposium on Design Automation and Microprocessors", Palo Alto, California, February, 1977.
11. Gardner, R.I., G. Estrin, and H. Potash, "A Structural Modeling Language for Architecture of Computer Systems," Proc. 1975 Internt'l Symposium on Computer Hardware Description Languages, New York, N.Y., Sept., 1975, pp. 161-171.
12. Gardner, R.I., "A Graph Model of Behavior for Digital System Design", Ninth Hawaii International Conf. on System Sciences, Jan., 1976.
13. Wang, L.T., "FFT Hardware Design Via Intel 8008 Microprocessors," Internal Report, Dept. of Computer Science, Univ. of California, Los Angeles, May, 1975.
14. Dennis, J.B., D.P. Misunas, C.K. Leung, "A Highly Parallel Processor Using a Data Flow Machine Language", Computation Structures Group Memo 134, MIT, Jan., 1977.

PARTICLE SIMULATION:

Number of particles	10^6
Operations/(particle·time-step)	300
Averaged speed/operation	20 ns
Simulation time/(particle·time-step)	6 us
Simulation time/time-step	6 sec
Number of time-steps/day	1.4×10^4
Total time-steps required/run	10^5
Total CPU time/run	7 days
Equivalent experimental time scale	10^{-7} - 10^{-6} sec

FLUID SIMULATION:

Number of grid points (100 × 100 × 20)	2×10^5
Number of field variables/grid-point	10
Estimated operations/(grid-point·time-step)	3000
Averaged speed/operation	20 ns
Simulation time/time-step	12 sec
Number of time-steps/day	7000
Total time-steps required/run	5×10^4
Total CPU time/run	7 days
Equivalent experiment time scale	10^{-4} - 10^{-3} sec

Fig. 1 CPU time estimation of both particle and fluid models; three-dimensional magnetostatic code

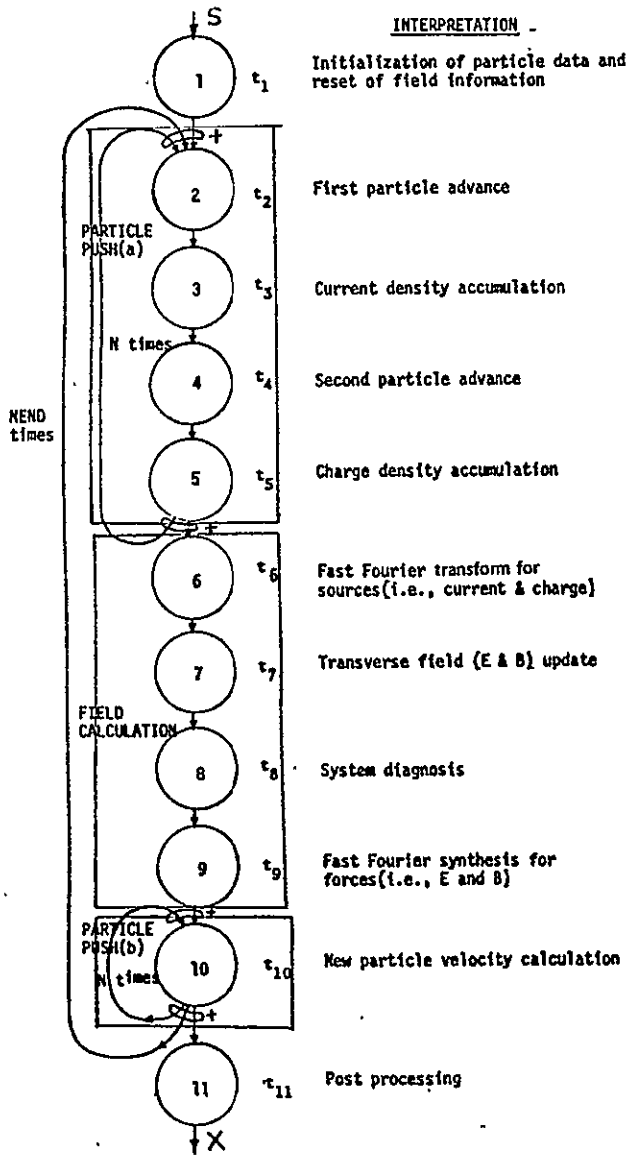

S
INTERPRETATION
1 t_1 Initialization of particle data and reset of field information
2 t_2 First particle advance
PARTICLE PUSH(a)
3 t_3 Current density accumulation
N times
4 t_4 Second particle advance
NEND times
5 t_5 Charge density accumulation
6 t_6 Fast Fourier transform for sources(i.e., current & charge)
7 t_7 Transverse field (E & B) update
FIELD CALCULATION
8 t_8 System diagnosis
9 t_9 Fast Fourier synthesis for forces(i.e., E and B)
PARTICLE PUSH(b)
N times
10 t_{10} New particle velocity calculation
11 t_{11} Post processing
X

Fig. 2 GMB's Flow of Control Graph of REM2-1/2D (Sequential Model)

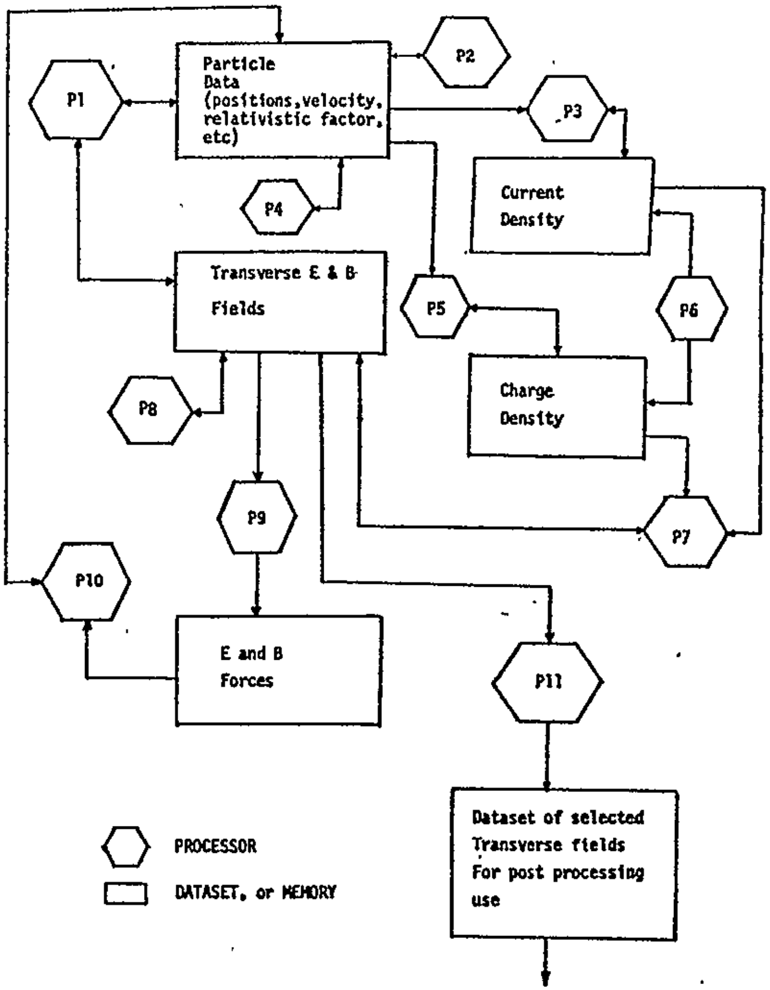
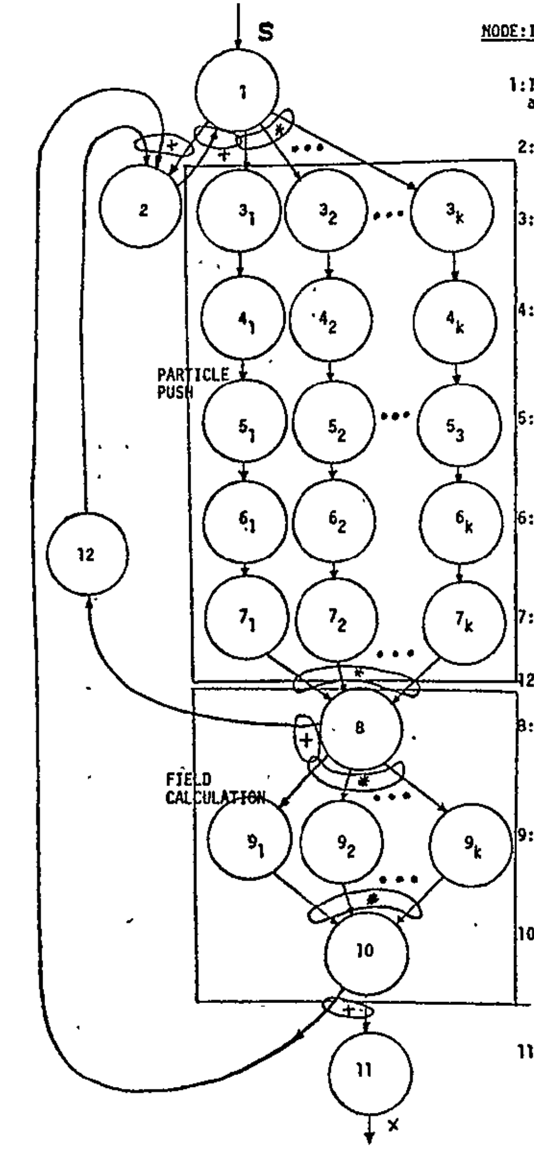

P1
P2
Particle Data (positions,velocity, relativistic factor, etc)
P3
Current Density
P4
Transverse E & B Fields
P5
P6
Charge Density
P8
P9
P7
P10
E and B Forces
P11
Dataset of selected Transverse fields For post processing use
PROCESSOR
DATASET, or MEMORY

Fig. 3 GMB's Flow of Data Graph of REM2-1/2D (Sequential Model)



NODE:INTERPRETATION

1:Initialization of particle and reset of field information

2:Loading of particle data from secondary memory to primary memory (buffer)

3:Velocity Update(1≤i≤K)

K: machine multiplicity

4:First particle advance(1≤i≤K)

5:Current density accumulation (1≤i≤K)

6:Second particle advance(1≤i≤K)

7:Charge density accumulation (1≤i≤K)

12:Updating particle data on secondary memory

8:Branch for iteration/Fast Fourier transform of source fields(current & charge)

9:Transverse field update

10:Fast Fourier synthesis of E and B forces

11:Post processing

Fig. 4 GMB's Flow of Control Graph of REM2-1/2D (Array Model)

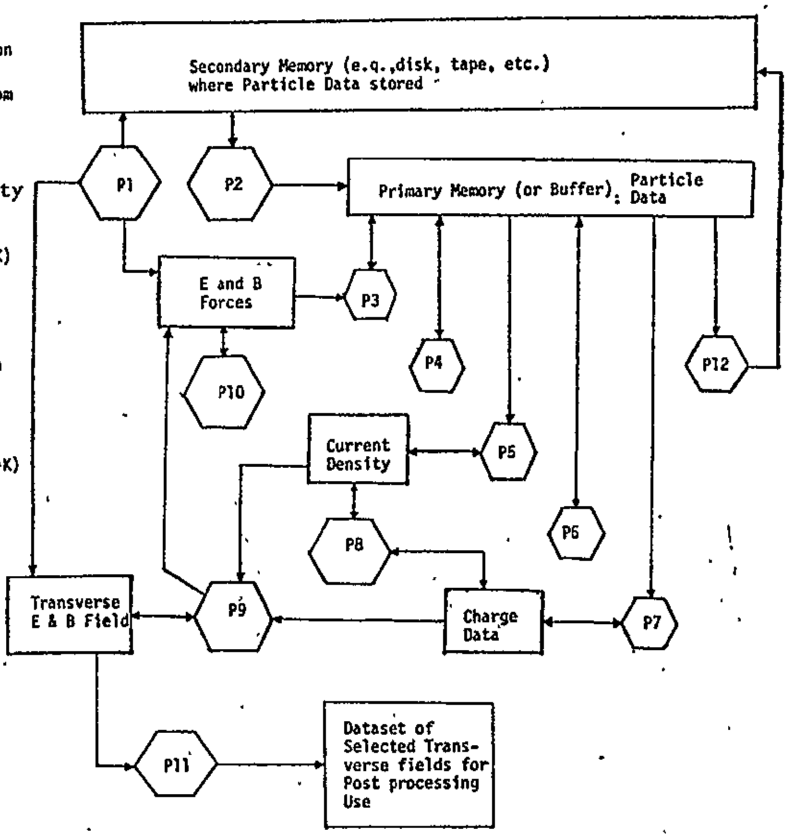

Secondary Memory (e.q.,disk, tape, etc.) where Particle Data stored
P1
P2
Primary Memory (or Buffer): Particle Data
E and B Forces
P3
P4
P12
P10
Current Density
P5
P6
P8
Transverse E & B Field
P9
Charge Data
P7
P11
Dataset of Selected Transverse fields for Post processing Use

Fig. 5 GMB's Flow of Data Graph of REM2-1/2D (Array Model)

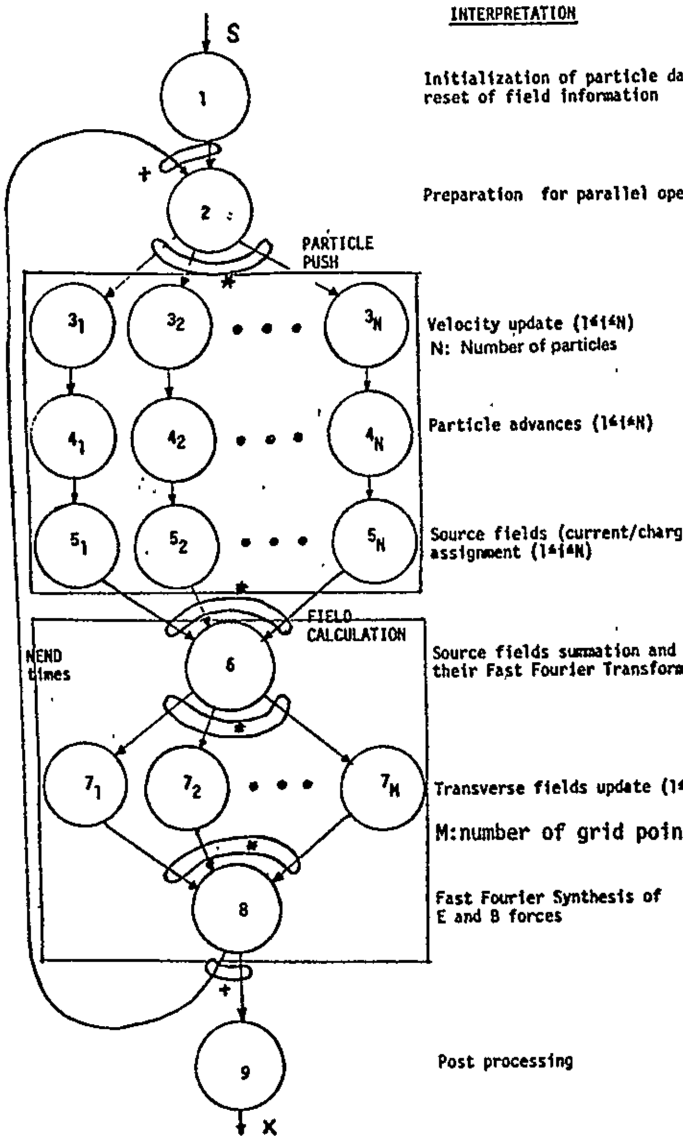

INTERPRETATION
S
1
Initialization of particle data and reset of field information
+
2
Preparation for parallel operation
PARTICLE PUSH
*
3_1
3_2
3_N
Velocity update (1≤i≤N)
N: Number of particles
4_1
4_2
4_N
Particle advances (1≤i≤N)
5_1
5_2
5_N
Source fields (current/charge) assignment (1≤i≤N)
FIELD CALCULATION
NEND times
6
Source fields summation and their Fast Fourier Transforms
7_1
7_2
7_M
Transverse fields update (1≤j≤M)
M:number of grid points
8
Fast Fourier Synthesis of E and B forces
9
Post processing
X

Fig. 6 GMB's Flow of Control Graph of REM2-1/2D (Fully Parallel Model)

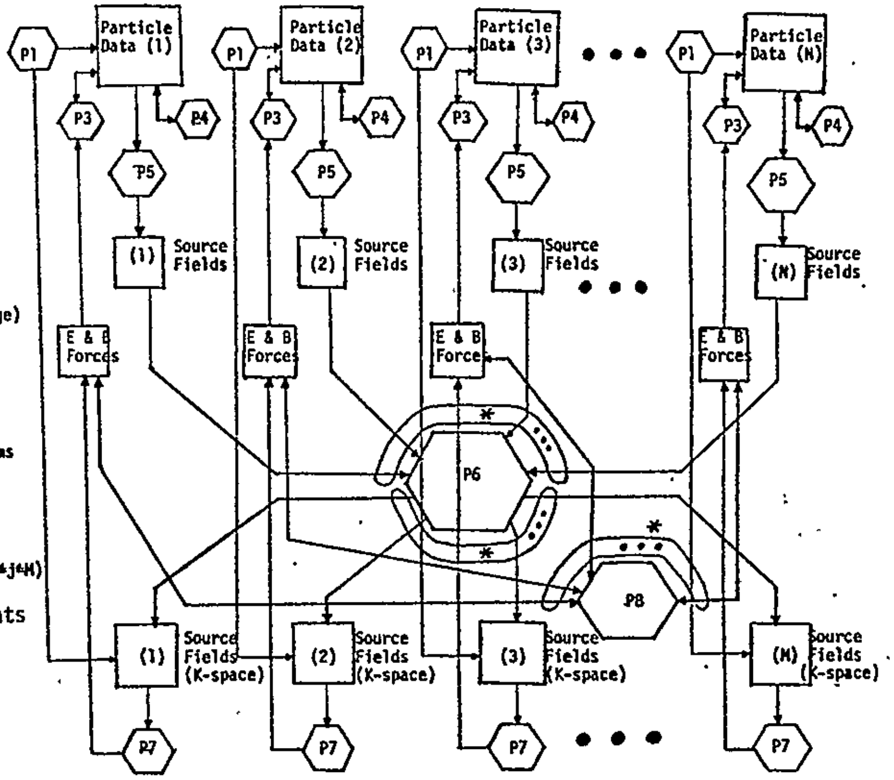


Fig. 7 GMB's Flow of Data Graph of REM2-1/2D (Fully Parallel Model)

Node ID number (from Fig.6)	programs or subprograms ID	sequential CPU time	multiplicity	fully parallel CPU time	estimation
	PARTICLE PUSHING [P.P.]				
3	Velocity Update [V.U.]	T_{VU}	N	$t_{VU} \leq \frac{T_{VU}}{N}$	$21\ t_{cpu}$
4 & 5	Charge/Current Assignment [C.A.]	T_{CA}	N	$t_{CA} \leq \frac{T_{CA}}{N}$	$6\ t_{cpu}$
	FIELD CALCULATION [F.C.]				
6	Fast Fourier Transform [FFT]	T_{FFT}	M	t_{FFT}	$2 \log_2\sqrt{M}\ t_{cpu}$
7	Field Update [F.U.]	T_{FU}	M	t_{FU}	$\frac{60}{2\times 6}\ t_{cpu}$
8	Inverse Fast Fourier Transform [IFFT]	T_{IFFT}	M	t_{IFFT}	$2 \log_2\sqrt{M}\ t_{cpu}$

For example, if $N = 10^6$, $M = 128\times128$, $\sqrt{M} = 128$ and averaged instruction time $t_{cpu} = 10$ ns,* then the total CPU time for one time step, or loop, is

$$
\begin{aligned}
T_{cpu}(\Delta t) &= t_{VU} + t_{CA} + t_{FFT} + t_{FU} + t_{IFFT} \\
&= 210 + 60 + 2\cdot70\cdot2 + 50 \\
&= 600 \text{ ns}
\end{aligned}
$$

*However, this figure is subject to change due to different t_{cpu} on different computers.

Fig. 8 CPU time estimation of lower bound computation speed of REM2-1/2D

SESSION 10

Panel on TOTAL SYSTEM ISSUES

John M. Levesque, Chairman

TOTAL SYSTEM ISSUES

Prepared Remarks by John M. Levesque
R & D Associates
Marina Del Rey, California

This presentation deals with one of the most important issues related to the use of a raw computer resource; that of how the programmer defines his calculation for subsequent execution on the computer. The presentation deals specifically with the questions: What language should the programmer use? and How should the programmer structure his program?

A question of importance to this panel is how one utilizes a raw computer resource capable of one billion floating point operations per second, to solve a problem whose solution is dependent upon such a resource being used efficiently.

Since the Billiflop machine necessary will have multiple parallel and/or segmented functional units to obtain such a speed, the system programmers, software writers and users are forced with the non-trivial task of writing operating systems, compilers and application programs to utilize such a capability efficiently.

The software problem of giving the user access to the available power of the machine has reared its head often in recent experience associated with use of the CDC 7600, CDC Star, ILLIAC IV and CRAY 1. The question which keeps on being asked is: When machine X is capable of performance rates 5-10 times that of the CDC 7600, why, in actual performance, is one lucky to get a factor of two over the CDC 7600?

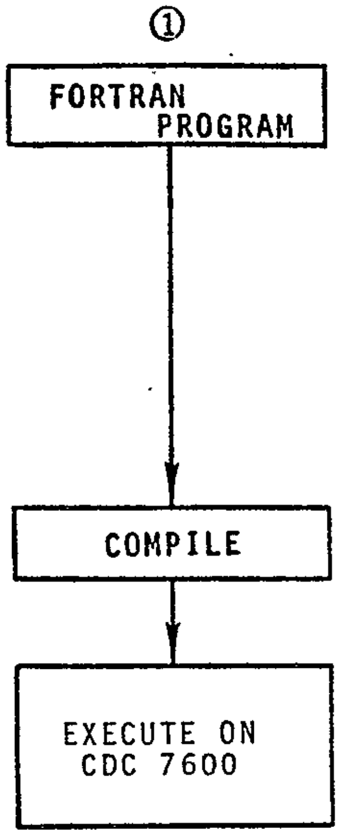
The answer lies in the fact that we now must understand how these new supercomputers get their potential speed-up, and use them accordingly. For example, consider the question of the performance rates obtained from user codes.

YEAR	MODEL	COMPUTER POWER
1955	IBM 704	1
1960	IBM 7090	5
1965	CDC 6600	25
1970	CDC 7600	125 (250)
1972	CDC STAR	25 (1000)
1975	CRAY 1	250 (725)

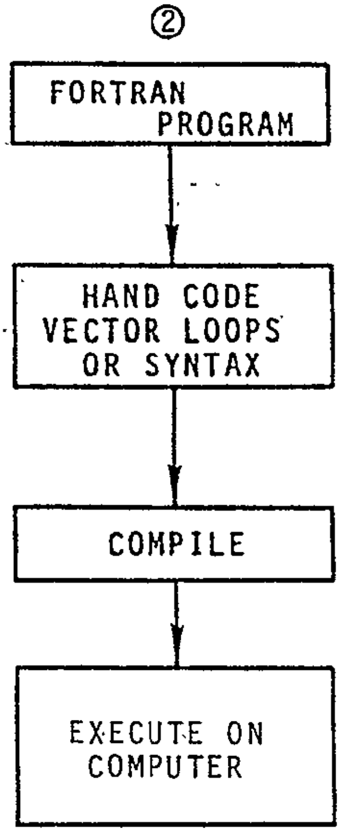
The above table describes the evolution of hardware technology which has offered computer users enhanced computational rates without significant software development. This sequence has proceeded with little help from the software experts. However, notice that the later developments (CDC 7600, Star and CRAY 1) supply small factors in scalar usage, while much higher factors (number in parentheses) can be obtained through utilization of the special vector units. This typically requires the computer user/programmer to rewrite his program into a form amenable with array or vector operations.

$$\text{PROBLEM} \xrightarrow{\text{PROGRAMMING}} \text{PROGRAM} \xrightarrow{\text{COMPILING}} \text{MACHINE CODE}$$

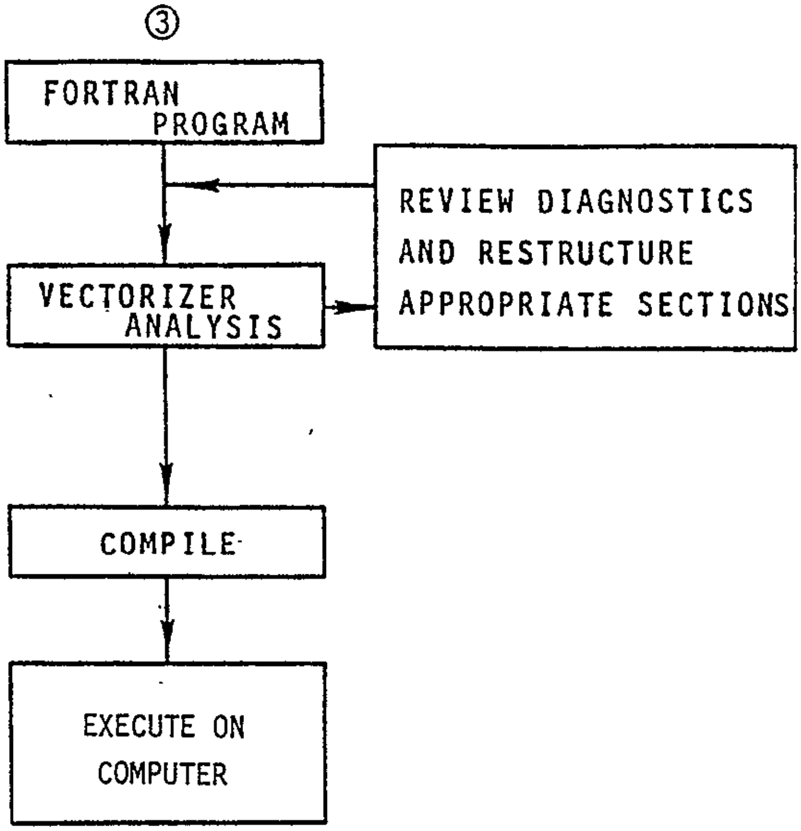
This chart illustrates the two processes used in solving a given problem on a particular machine. With the advent of machines with multiple-segmented functional units, the programming must necessarily become more sophisticated by using methods to solve the problems which can employ array operations. Also, computation techniques must consider how to aid the programmer in using the machine efficiently.



The normal technique for running a program on a computer is to use the available FORTRAN compiler. This results in small programming effort and transportability; however, poor execution rates are realized.



Another technique is to hand code those portions of the program which use the majority of the central processing time. This represents a large programming effort and no transportability; however, excellent execution rates are obtained.



A new technique consists of using an available pre-compiler to aid the programmer in developing efficient programs for vector or array processors by first analyzing the FORTRAN code to determine where vector or array operations may be performed. Diagnostics are then supplied on the vectorizability of the code. Finally, once the programmer is happy with the vectorization of the code, the pre-compiler will generate the appropriate vector syntax. This results in the transportability of 1 and efficiency of 2, with the programming effort larger than 1 and smaller than 2.

While the programmer cannot expect the Vectorizer to perform the entire task of optimizing a program, he must consider the following:

STEPS IN CONVERTING A PROGRAM TO VECTORIZABLE FORTRAN

I. PRIOR TO RUNNING THROUGH THE VECTORIZER:

ANALYZING ALGORITHMS TO DETERMINE IF A MORE "VECTORIZABLE" ALGORITHM IS FEASIBLE

ANALYZING FORMULATIONS OF A PARTICULAR ALGORITHM TO DETERMINE IF A MORE "VECTORIZABLE" APPROACH CAN BE UTILIZED

ANALYZING PROGRAM FLOW TO DETERMINE IF THE PROGRAM IS STRUCTURED TO FACILITATE VECTOR OPERATIONS

ANALYZING THE PROGRAM TO DETERMINE WHERE IT USES THE CENTRAL PROCESSING TIME

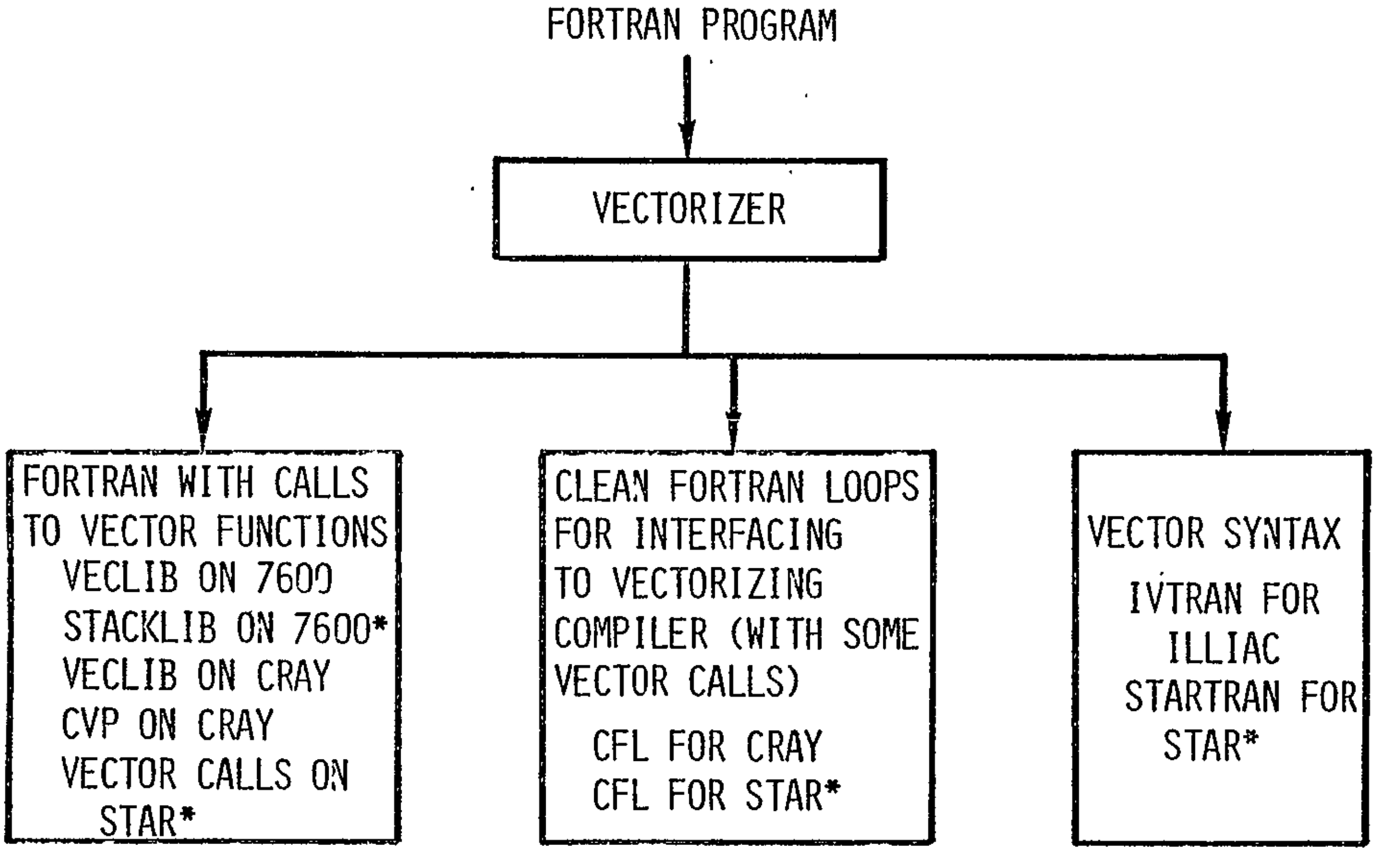
II. TO BE DONE AFTER VECTOR DIAGNOSTICS ARE EXAMINED:

ANALYZING DECISION PROCESSES TO DETERMINE IF DECISIONS CAN BE EITHER ELIMINATED OR SEPARATED FROM THE COMPUTATIONAL PROCESSES

ANALYZING CONTENTS OF DO LOOPS TO ASSURE THAT STATEMENTS ARE INDEPENDENT AND EXECUTABLE ACROSS THE VARIABLE ARRAYS

The transportability of the resultant FORTRAN code is extremely important, as this chart illustrates.

VECTORIZER OUTPUT OPTIONS



FORTRAN PROGRAM

VECTORIZER

FORTRAN WITH CALLS TO VECTOR FUNCTIONS
- VECLIB ON 7600
- STACKLIB ON 7600*
- VECLIB ON CRAY
- CVP ON CRAY
- VECTOR CALLS ON STAR*

CLEAN FORTRAN LOOPS FOR INTERFACING TO VECTORIZING COMPILER (WITH SOME VECTOR CALLS)
- CFL FOR CRAY
- CFL FOR STAR*

VECTOR SYNTAX
- IVTRAN FOR ILLIAC
- STARTRAN FOR STAR*

*CURRENTLY NOT AVAILABLE

Of course, other important issues exist in making a raw computer resource into a more friendly system. Issues such as I/O bandwidths to mass storage devices and/or other computers in the facility cannot be overlooked. The hardware and software components necessary to link the entire system together must be such that all I/O paths can handle the necessary transfer rates to interface a raw computer resource to the data sources and user resources.

PERSPECTIVES ON THE PROPOSED COMPUTATIONAL AERODYNAMIC FACILITY

Mark S. Fineberg
McDonnell Douglas Automation Co.
St. Louis, Missouri

Good morning. I am pleased to have the opportunity to share my perspectives with you. I think I should first state my preference as to NASA's role. I would like them to be the "path finders", be on the leading edge providing a technology boost. Where there are trade-offs between giving more effective service and advancing the state-of-the-art, NASA should take the largest step possible for all of us. One aspect of that is that the facility must be planned to maximize spin-off benefits.

Would we use a computational aerodynamic facility if NASA offered it? I suspect we would if...if the very real management problems were solved, and if it were a worthwhile tool. But, at best, I see a short life for it. I cannot believe the fantastic computer progress we have seen is going to suddenly stop. If NASA has the capability, say in '82, we will have it in '85, and in '90 every junior college will be playing the same game. This does not necessarily mean ten minutes per run; it means at a reasonable cost.

I would also like to quickly discuss the ten minute criterion. On the positive side, it is a good idea to pick a specific criterion as a benchmark, and in many respects, one number is as good as any other. On the other hand, ten minutes is an awkward interval, too long for interactive response, yet, if it is batch on a heavily used facility, the execution time is not a criterion at all, response time is. Response time is dependent upon queue size which is in turn determined by how much dead time we are able to afford. This implies that the cost per operation determines response time, not raw speed. For example, if we had a ten minute machine and a load of six ten-minute jobs every hour, the response would be very slow (infinite in theory). But if a twenty minute machine were available at one-third the cost, we could buy three for the same money. The three slower machines could provide excellent response time with the same load.

This is a major concern. I see no evidence that the sensitivity of the cost per job against raw speed has been studied. If the twenty minute machine is in fact less than half the cost of the ten minute one, there is no reason to build the faster computer. But I don't have the slightest idea what the relative costs are.

If there is a common thread to my rather random impressions, it is that things are not all that different. Software and the interface with people are major concerns. The primary hardware parameter is still simply "Bang for the Buck".

TOTAL SYSTEM CAVEATS

Wayne Hathaway

Ames Research Center, NASA

Per Brinch Hansen (1) has defined a computer operating system as follows:

> An operating system is a set of manual and automatic procedures which enable a group of people to share a computer installation efficiently.

While this definition was intended to describe only a computer operating system, it is also very applicable to the total system concept of a supercomputer facility. That is, such a total system facility should provide a set of manual and automatic procedures, together with the hardware to carry out these procedures, which enable a group of users to share the facility, and thus solve their problems, efficiently.

There are of course many potential problems which can occur when designing such a total system facility, and I would like to discuss some of these by tackling many of the major words in the above definition.

EFFICIENTLY

This is actually the word that I dislike most in the definition, primarily because today it is well recognized that effectiveness is much more important than efficiency. As an indication of what I mean, I would like to give the following distinction between efficiency and effectiveness:

> Efficiency is doing things right --
> effectiveness is doing the right things.

Of course "doing the right things" means doing the useful things, doing the things which are important -- actually solving problems. It can also mean doing only the important things, meaning not trying to do more than can reasonably be done. In any computing facility, regardless of size, there will be some jobs that simply cannot or should not be done. The bigger the facility is, however, the more temptation there will be to try to do everything, to be all things to all people. If there is to be any hope of the facility ever being useful -- being effective -- then this temptation must be fought at all cost. This also manifests itself in the actual development of a facility, especially one which attempts to extend the state of the art significantly. Such extending is fine if kept under control, but one must be very careful not to try to extend the states of too many arts at once. Computer architecture, component technology, programming languages, operating systems, communications techniques -- advances in any one of these areas would be great, two might even be better, but to attempt all five would almost certainly be a disaster.

COMPUTER INSTALLATION

What is meant by the term "computer installation" above? The hardware, of course. But also a lot more -- documentation, consulting services, tape libraries, data communications facilities, even multiple computer systems networked together. Thus we -- the system designers and implementors -- must be very careful to impress upon the user the full range of services which the modern computer installation can provide. And of course such services must have the traditional attributes: reliability, availability, serviceability, security, capacity, and so forth. But modern facilities must also have one other important attribute: friendliness. If the user is to be effective he must be reasonably happy, and this can be achieved only when the facility is friendly.

SHARE

There are two sides to the concept of sharing a computer installation. In the first place, users are competing, competing for CPU time, competing for disk storage, competing for programming assistance. But they are also co-operating, co-operating in sharing programs, data files, documentation. The facility must be designed to make the competing side of sharing as transparent and painless as possible, while emphasizing the co-operating side. It must not only make sharing available, it must make it attractive, even unavoidably so. An example of this from my ARPANET experience is a paper that I recently co-authored with several other ARPANET users. We had a rather limited amount of time to spend on the paper, and thus used the network extensively. We sent mail, shared files containing drafts, made comments "on" each others' working copies. And the paper was written on time and accepted -- all over thousands of miles and without a single meeting or even phone call among the participants!

MANUAL AND AUTOMATIC PROCEDURES

I'm sure everybody agrees that operating systems provide automatic procedures, and that such procedures are important and must be carefully designed and implemented. But the word that I would like to stress here is manual, because I feel that the manual procedures in use at a facility can be much more detrimental to effective use of the facility than the automatic procedures. Perhaps this is because automatic procedures (that is, actual operating system services) are much more interesting to the typical system implementor. At any rate, how many times have you run into such "insurmountable" obstacles as having to walk across the street in the rain to pick up a listing, or not being able to check out your tape to take it with you, or having to turn your deck in and wait three hours to have it interpreted? And another example, again from ARPANET experience: the Campus Computing Network (CCN) at UCLA actively sells time over the network, and a large portion of their revenue comes from network users. Use of their system is in fact quite easy from a remote site, and it takes only a matter of minutes to become familiar enough to use it effectively. Unfortunately it takes a minimum of one to two weeks to get the required forms mailed, signed, and returned to allow you to begin to use the facility!

GROUP

Traditionally the customers of a particular computer center were fairly well defined and reasonably close to the facility: students at a university, researchers in a development shop, managers using an information management system. Today, however, such groups can be extremely large, spread all over the country, and under many separate managements. This of course presents many new problems to system designers and facility managers; the Sears-Roebuck mail order house must be run differently than the corner drug store. I should also point out that the group of users which are serviced by a modern computer center includes all classes of users, system programmers, mid-users, and end users. In fact, most of the attendees at the Workshop are mid-users rather than end users, because they are researchers producing new codes that actual engineers and designers will use as production tools. They are the mid-users producing the tools whch will be used by the end users. Everybody in the user group should of course be able to use the facility effectively, not just the traditional end user.

PEOPLE

The last word I would like to discuss is in fact the most important -- the *people* that use the facility. To paraphrase Mr. Lombardi, people are not the most important thing, they are the *only* thing. Without people, there is no reason for the facility whatsoever. As an example, there has been much discussion on the difference between data and information, with all sorts of attempts to describe one or the other. The distinction that I prefer is simply that data becomes information only inside a human being's head.

Another thing that facility designers must keep in mind is that *their* only reason for existence is to make sure that the facility in fact meets the users' needs, the needs of people. A little anecdote illustrates this well: whenever a ship attempting to dock accidentally rams the pier, you hardly ever blame the pier. If we build a facility that doesn't meet the needs of the users, it is rather silly to say "Damn users, they built the pier in the wrong place again." Nor is it reasonable to expect the users to run back and forth on the beach moving the pier -- *we* must aim at what is needed and make sure we hit the target.

My closing point is that if we *don't* do these things, we are likely to get the following comment from the user community, and it is the *last* thing we want to hear:

> We are faced with an insurmountable opportunity!

(1) Brinch Hansen, Per. *Operating System Principles*. Prentice-Hall, Inc., 1973.

D34

A HIGH LEVEL LANGUAGE FOR A HIGH PERFORMANCE COMPUTER

R. H. Perrott*
Institute for Advanced Computation
Sunnyvale, California

Abstract

During the last two decades there have been many developments in computer component technology enabling faster execution speeds. Unfortunately there have not been comparable developments in software tools. The result has been that for sequential computers, the cost of software production has risen substantially and the software has been unreliable and difficult to modify. However recent software engineering techniques have enabled the production of reliable and adaptable software and at a reasonable cost.

The proposed computational aerodynamic facility will join the ranks of the supercomputers due to its architecture and increased execution speed.

At present, the languages used to program these supercomputers have been modifications of programming languages which were designed many years ago for sequential machines. If history is not to repeat itself, a new programming language should be developed based on the techniques which have proved valuable for sequential programming languages and incorporating the algorithmic techniques required for these supercomputers.

*On leave from the Department of Computer Science, The Queens' University, Belfast.

I. INTRODUCTION

The last twenty years have seen the design and development of several generations of computer hardware components each giving rise to a faster processor speed; the more recent increases in the number of operations performed per second have been obtained by a revolution in computer architecture, rather than component technology, leading to the introduction of high performance computers such as CDC STAR-100, CRAY 1 and Illiac IV. Unfortunately there has not been a comparable investment of time, money and research into the development of programming languages or software production tools to utilize the technological and architectural advances.

The net result of the imbalance of research and development effort for sequential machines has been that for most installations the cost of software production has increased in comparison with its subsequent use.

The reliability of the software has also been suspect while its adaptability or modification has been a difficult, and at times an impossible, task. There is every possibility that the same pattern will be repeated for high performance computers if an effort is not made to develop software which will make these supercomputers easier to operate and easier to program.

However, the development of new techniques under the various headings of 'structured programming ,' 'stepwise refinement' and 'software engineering' has led to the introduction of languages and techniques for sequential computers which produce software of improved quality and reliability and at a reasonable cost. Hence it is now possible to apply this knowledge to design and implement a higher level language for a high performance processor.

Most of the languages currently used to program supercomputers are extensions of languages which were specifically designed many years ago for sequential machine architectures. It is now apparent that these supercomputers require a language created in their own generation using, as far as possible, the experience accumulated in language design techniques and incorporating the new approaches that are necessary in writing algorithms for these supercomputers.

Since the proposed computational aerodynamic design facility probably will have a similar architecture but an operational speed surpassing any of the existing supercomputers , the same requirements can be regarded as necessary for its programming language.

II. HARDWARE

The decrease in the cost of computer components and the corresponding increase in their reliability has led to the construction of more powerful computers based on a uniprocessor configuration. However, the resultant speed increases have still not been sufficient to satisfy the demands of computational fluid dynamics and other scientific users. The types of large problems being addressed or planned require a significant increase in processing power in the very near future; the advance of knowledge has led to problems which only a few years ago were considered impractical. Hence users can neither afford nor desire to wait on the next generation of sequential computers.

A common theme which can be identified in most large scale applications involves the manipulation of vectors and arrays - operations which are repetitive on sequential machines. On this basis,the most promising approach in providing the extra computational power required is to duplicate the already existing hardware components. The extra arithmetic and logic units can be organized to reflect the nature and the structure of the application and produce many more calculations per second.

In such problems the vector replaces the scalar as a unit of data which is required to be manipulated and the arrangement and organization of the processing units should reflect this. Also, it is nearly always the case that similar operations are required to be performed on different data - the instruction sequence is the same, only the data is different. Hence an arrangement of the processing units into a vector or an array would appear to be the most promising method of providing the extra computational power.

However, the combining of two or more processing units requires that the processors be synchronized so that the data which is being manipulated is not corrupted. The programs on such systems face the possibility of introducing time dependent coding errors which are difficult, if not impossible to detect by normal program debugging methods. Only if very precise and easy to use synchronization concepts can be found and implemented is there any chance of a user being confident that his data is not being corrupted. To place the burden of synchronization upon the programmer (via the programming language) can only cause his attention to be directed away from his main task of developing a large scale program.

However, such synchronization problems can be avoided if the processing units are constrained to act in step obeying the same instruction sequence, and if each processing unit is allowed to access one portion of memory only, and is forbidden by the hardware to access any other locations. Under these conditions, the corruption of one processing unit's data by another is impossible.

The programmer can then manipulate a large data base on a vector or array basis safe in the knowledge that the corruption of his data is impossible, and free from the problems of processor synchronization. Such an approach has been successfully developed and implemented in other high performance computers. If the computational aerodynamic facility adheres to such an architecture, it raises a major difficulty which is present in other supercomputers and which must be overcome in the design of a new higher level language, viz., the aligning of the data within the processing units' memory. Unless this problem is solved satisfactorily, the performance of the machine will be severely deqraded.

III. SOFTWARE

The programming language is the framework in which the programmer formulates his thoughts in solving problems in his particular field or discipline; as such it should provide the user with a notation (or enable him to construct one) with which he is familiar or with which he feels comfortable. The syntax should not be a barrier or inconvenience to his use of the machine requiring him to distort his method of solving his problem. In other words the language should enable the user to isolate the relevant features of his problem; such a process is known as abstraction and is one of the most powerful tools available for the construction of computer programs.

On a machine in which it is possible to perform parallel computation on the data, the parallelism should be readily apparent in the syntax of the language. Since the language is the means of communication between human and machine, this will have benefits for both parties. Firstly the user, by the use of these parallel features, will be able to construct more efficient algorithms for solving his problems. Secondly, the compiler will be able to generate more efficient object code, and thus eliminate the effect involved in the automatic detection of such parallelism.

The language should be developed on the principle that it should give as much assistance as possible to a programmer in the design, documentation and debugging of his programs. Such a language will then enable a clear expression of what a program is intended to achieve. This should be accomplished by defining a language which will support various levels of program development ranging from the overall design strategy down to the coding and data representation. It will also enable the cooperation of several programmers on a single project and help ensure that separately developed subprograms are successfully assembled together. The language should be developed as far as possible without specific reliance on a given order code and storage organization to enable its implementation on other supercomputers and thus ensure the portability of programs among different research workers at different installations.

The language should promote the self documentation of programs; documentation is an integral part in the design of a program and the language should encourage and assist with this process. Programs will then be readable which will enable them to be easily understood; each well chosen identifier can do more to indicate the intended meaning than several lines of explanatory text. Self documentation has additional benefits in error detection and program debugging and will also faciliate the modification of a program after it has been commissioned.

Since errors occur in well structured programs written by well trained programmers on sequential machines, errors can be regarded as inevitable with parallel programs. Hence the programming language should offer as much help as possible in detecting and eliminating errors. Obviously the initial design decision and subsequent documentation will play a large part in reducing the effort involved in error detection. The choice of language features should reduce as far as possible the scope for coding error or at least guarantee that such errors are detected at compile time before the program executes. Other errors should be detected at run time.

The language should facilitate the optimization of a program. This could take the form of statement counts which indicate that part of the program which is most heavily executed and therefore to be considered closely when trying to improve the program's performance. This will also have the benefit of giving a greater insight into the working of the program. Execution timings should also be provided for all or part of the program to indicate the most time consuming, and therefore another section in which to improve performance. The language should also provide the facility of selective dumping in a form which is easy to diagnose, and enable the tracing of selected portions of a program both at the statement and the procedure level.

The main objectives for such a language should be as follows

i) simplicity

The constructs of the language should be simple and easy to learn, based on the fundamental concepts which are involved in the algorithms for computational aerodynamics. The number of constructs should be simple to understand in all possible situations and interactions , i.e., each construct should be capable of being defined independent of the other constructs. If the constructs adhere to this objective, the language will simplify the use of such a supercomputer and make it more accessible rather than inhibit access or understanding.

ii) ruggedness

This is concerned with the prevention or early and cheap detection of errors. The language should not give rise to errors which have machine or implementation dependent effects and which are inexplicable in terms of the language itself. The compiler should therefore be totally reliable in those constructs which are offered by this language and these constructs should be difficult to misuse. The language should provide automatic consistency checks between data types which provide added security. Such checking is well worthwhile as it enables the programmer to have a greater confidence in the code he produces.

iii) fast translation

Since programs will be compiled and executed many times during their development stage, it is important that the speed of compilation is fast. This will discourage users from independently compiling parts of their program which can lead to errors with the interfaces or changes to the data structures.

iv) efficient object code

Rather than rely on the speed of the computer to reduce the effect of inefficient object code, the language should be designed to produce object programs of acceptable compactness and efficiency. This does not mean that every single odd characteristic of the hardware should be used at any cost. The language should reduce the quantity of machine dependent software which inhibits the development of improved designs. Machine dependent procedures should be written only when it is impossible to reduce the operation to existing procedures and achieve comparable efficiency.

v.) readability

The finished programs should be immediately readable by the author and his co-workers; the emphasis should be to bias the syntax of the language towards the human rather than the machine. As mentioned previously, the reading of a program is an important step in the detection and the elimination of coding errors and is therefore more important than writability. This will enable one programmer to take over when another leaves or a programmer to understand his own program six months later.

IV. CONCLUSION

The above objectives have been those which have been successfully achieved in the design of sequential languages and it is believed can be applied to a language for an aerodynamic design facility. However, certain compromises will be necessary due to the special architecture and techniques which must be used to design algorithms for such a facility.

It is fair to point out that any new language will meet a certain amount of opposition from those users of other languages who understandably are reluctant to change. Only if the benefits of this new language are widely explained and justified and the programming of such a supercomputer is shown to be easier will the language have any chance of success.

The mismatch between hardware and software development effort for the supercomputers is already apparent, and through time will probably increase if a new language or new constructs are not developed which will make them more usable and enable the construction of reliable software.

The computational aerodynamics community is presented with the opportunity to insist that a new language based on well tried and proven techniques is developed. Such a language would have benefits for not only the aerodynamics research community but also for other scientific research workers.

ACKNOWLEDGEMENT

The author wishes to acknowledge the influence of C.A.R. Hoare on the ideas expressed in this paper.

V. REFERENCES

[1] Brooks, F.P. Jr. (1975)
The Mythical Man Month
Addison - Wesley

[2] Cheatham, T.E. Jr. (1972)
The recent evolution of programming languages.
IFIP 1971 298 - 313
North - Holland Publishing Co.

[3] Dahl, O.J., Dijkstra, E.W., Hoare, C.A.R. (1972)
Structured Programming
Academic Press 1972

[4] Hoare, C.A.R (1973)
Hints on programming language design
Stanford Report CS-403

[5] Hoare, C.A.R (1972)
The Quality of Software
Software - Practice and Experience 2, 103-105

[6] Wirth, N. (1974)
On the design of programming languages
IFIP 1974 386-393

[7] Wirth, N. (1975)
An assessment of the programming language Pascal
IEEE Trans. on Software Engineering 1,2,192-198

[8] Wirth, N. (1976)
'Programming Languages' in Software Engineering
Published by Academic Press 1977

D35

USER INTERFACE CONCERNS

David D. Redhed
Boeing Computing Services, Inc.
Seattle, Washington

Being a part of this program is a bit uncomfortable for me and probably somewhat puzzling to you. As best I can tell, most of the participants are known by name in their field and my name on the program had to look like a misprint. For this and other reasons, I feel compelled to give you some insight into my background and interests. This way, if you do not like what I am going to say, you will have a rational basis for rejecting it.

My fundamental interests are in computing systems rather than the engineering technology which uses them. However, most of my years at The Boeing Co. have been spent trying to help the engineers survive while trying to use computers. Computing systems designers and builders remind me of an observation Marshall McLuhan made in his book Understanding Media. Some one had criticized the looks of the Citroen car. McLuhan observed that the designers of the car never imagined that anyone would look at it. Sometimes I think that computing systems developers never really imagine that anyone is going to use the system for any real concrete purpose.

I have been super-sensitive to the difficulties of a dominantly non-computing oriented user who has a job to do and needs to use the computer for it. So you must bear this in mind when you try to interpret my remarks. I do not apologize for this, I am merely warning you.

Something else you need to know about me is what I have been doing during 1977. I have been on a vector processor study that has resulted in my trying to actually use one of these vector computers. Below is a list of the primary interests of this project:

1) effects on algorithm development
2) effects on software development
3) implications for current software
4) measurement of performance and cost
5) useability of the system when accessed remotely

The first two are oriented at assessing the effects of vector processors on the way we do our algorithm development and the way we construct the resulting software. The third is aimed at learning about demands on our current production software as the use of vector computers increases. The fourth one is an obvious cost/performance evaluation. The fifth one is not one of our original interests, but showed up after we began doing some work on the STAR-100.

We have learned quite a bit about these topics, although number 4 remains a bit fuzzy. I originally had intended to talk about an aspect of number 2 with respect to compilers, but the past two days have convinced me that I need to talk about number 5.

By the end of yesterday's sessions I could see three possible goals for the Numerical Aerodynamic Simulation Facility (NASF).

1) a computational fluid dynamics(as opposed to aerodynamics) algorithm development tool.
2) a specialized research laboratory facility for nearly intractable aerodynamics problems that industry encounters.
3) a facility for industry to use in their "normal" aerodynamics design work that requires high computing rates.

For goal 1, the current approach seems reasonable. For goal 2, it also seems reasonable, although a somewhat broader based computing facility concept may be required. Goal 3, I believe, is unreachable with the current approach and especially in the approximate schedule set forth - in use by 1983. I do believe that pursuit of goal 1 and goal two should continue. Some of the requirements outlined in the last two days seem a bit inconsistent to me, but that will likely get settled in time. I think that the general industry will be well served by this project. What I do object to is the presentation of the image that the NASF will be an industry tool in the sense of goal 3.

Having just spent several months working on STAR-100 from 2000 miles away, I want to share with you what I think is the central system issue for industry use of such a computer - the quality of the user interface as implemented in some kind of a front end to the vector processor.

At Boeing, we are moving steadily towards a situation where the dominant mode of interaction with the engineering computing facilities is via an interactive terminal. Not many programs are interactive in nature, but the input is prepared, jobs are entered and controlled, and results are digested in an interactive manner. More recently, some of this work is getting distributed out to minicomputers. This interactive approach is how I began with my STAR work and after several months of pretty successful work with it, I can tell you this: I do not know one engineer at Boeing who would put up with that interface for even one day, assuming that he really had to get some work done. He would find some other way to do it.

I can take time to give you only one concrete example. Assume a user has prepared an input deck and now goes through the following logical steps to use the data:

- execute a program
- examine the results (an error is found)
- edit data
- execute a program
- examine the results (no errors found)
- route the output

To accomplish these six steps he must input a total of 24 commands (exclusive of editing, etc.) through the terminal keyboard. A majority of the 18 extra commands are due directly to the awkward relationship between the front end and STAR-100.

I have no intention to single out CDC as a poor designer of systems, for I do not think that adequate user support for working with a high speed computer like STAR exists in any commercially available software. CDC shows up most clearly because they are the only commercially available system for Boeing. This panel is concerned with total system issues, and I have not seen any design considerations from the two contractors with respect to front end facilities. They both maintain that their standard medium systems will do the job. All I know is that this is not true for STAR today and it is going to take a lot of work before it gets significantly better.

If goal number 3 is not of central interest, then my concerns are not appropriate for NASA and the NASF. But any vendors who hope to market less ambitious computers than the NASF should take note.

SESSION 11

Panel on SPECIALIZED FLUID DYNAMICS COMPUTERS

David K. Stevenson, Chairman

SPECIALIZED COMPUTER ARCHITECTURES FOR COMPUTATIONAL AERODYNAMICS

David K. Stevenson
Institute for Advanced Computation
1095 East Duane Avenue
Sunnyvale, CA 94086

In recent years, computational fluid dynamics has made significant progress in modelling aerodynamic phenomena. Currently, one of the major barriers to future development lies in the compute-intensive nature of the numerical formulations and the relative high cost of performing these computations on commercially available general purpose computers, a cost high with respect to dollar expenditure and/or elapsed time. Today it is appropriate to consider specialized computers to address these problems in order: to permit current techniques to demonstrate their capability to be used in a routine engineering fashion; to investigate the relative merits of the different mathematical and physical approaches to these problems; to accelerate the evolution and development of existing and new methods to increase our understanding of aerodynamic properties such as turbulence; and to increase our ability to employ useful numerical models in the initial design of rigid bodies which must exhibit specific properties in the presence of fluid flows. Fortunately, today's computing technology will support a program designed to create specialized computing facilities to be dedicated to the important problems of computational aerodynamics; one of the still unresolved questions is the organization of the computing components in such a facility, and it is this question which this paper addresses.

We begin by reviewing the characteristics of fluid dynamic problems which will have significant impact on the choice of computer architecture for a specialized facility. First and foremost is the very large data base which one encounters in these problems. The large size arises from two major causes: the three-dimensional nature of the physical model and the high resolution required along each dimension in order tc represent the phenomena of interest. Next, for any given solution technique, the large data base is accessed along very regular patterns, and the number of conceptually distinct accessing patterns is relatively few (on the order of ten to twenty). In addition, the data base is usually viewed through a relatively small computational window moving through the data -- information associated with each node of a grid interacts either with a small neighborhood of surrounding grid nodes or with nodes along a line. Generally speaking, a moderate amount of floating point calculation is performed with the data in this window (from ten to a hundred operations per datum), and the computational stencil -- or form of the computation -- involves relatively complex interaction of computed quantities. Finally, many sweeps through the data base are required to solve a given problem (either to reach a steady-state or to observe a transition phenomenon), although many computational windows could be passing over the data base, independently and concurrently,at a time.

The above characteristics of the fluid dynamics problem dictate some of the characteristics of a specialized computer to be dedicated to this problem. The large data base and small computational window suggest an hierarchical memory will be both cost-effective and computationally feasible. The regular accessing patterns, and their small number, suggest tailoring the capabilities of the data paths between the stages of the memory (although "fixed" paths will impact the ability to solve various sized problems efficiently). The possibility for independent and concurrent processing of slices of the data base suggests the attractiveness of some form of parallel processing, although increasing the processing capability of the computer places greater demands on the bandwidth of the data accessing (and rearrangement) mechanism within the memory hierarchy. And the complexity of the computational stencil employed in these problems suggests the attractiveness of a sophisticated processing module (sophisticated both in processing capability and in local memory organization).

Given these qualitative characteristics of a specialized computer, it is interesting to consider various alternatives in the organization of the components of such a computer. Since most of today's numerical formulations of the problem readily admit a high degree of parallelism in the computation, we will concentrate on computer architectures which support parallel computation, namely, pipeline and array architectures.

A pipeline approach arranges multiple processing modules in an assembly line fashion, with part of the computation being executed at each stage of the multi-stage unit. Data is brought up from the memory system and pushed through the pipe, then returned to the memory. One of the main bottlenecks of this architecture is the pathway between the memory and the processing station. Not only must this pathway have a high bandwidth to feed the pipeline, but it must also be fairly sophisticated to permit the efficient access of the memory under several distinct accessing patterns. One way to alleviate this burden is to make the pipeline more sophisticated: by adding a local memory to the processing station, more of the computational stencil can be executed during each pass of data through the memory-to-pipe pathway. As noted above, fluid dynamics formulations tend to have complex computational stencils, so one expects that the more successful specialized processors which follow a pipeline philosophy will incorporate a local memory.

Of course, associating a local memory with the processing system is one of the defining characteristics of an array architecture, the main difference between an array and an "intelligent" pipeline (pipeline with local memory) is that in an array, each processing station is simpler than a high performance pipeline, and therefore there are proportionately more processing elements than pipelines for equivalent computing power (interestingly, for a given level of performance, the chip count to implement either approach is about the same). An array architecture, however, allows two significant departures from the above outlined pipeline architecture.

The first departure involves the location of the memory-to-processor pathway which in a pipeline philosophy must be between the memory and the processing station (and, incidentally, must be bi-directional). This path provides two functions: to get data to the processing station, and to get the right data to the right processing station at the right time for processing (data alignment). In an array processor, this later function can be performed by a processor-to-processor pathway (which need only be uni-directional). Thus, information in such an array need flow from processing station to processing station only when it resides in the "wrong" station, in contra-distinction to a pipeline-based architecture wherein information must flow through the corresponding network for any processing to be performed. It is this observation which permits an array architecture to occupy a greater spacial domain than a pipeline architecture (or any architecture which is committed to a centralized computing station), and hence an array has a greater potential for high performance.

The second departure lies with allowing each processing element to execute code independently. This is possible since an array's processing element, being simpler (and slower) than a pipeline, is less voracious in consuming operands, and hence the instruction fetch and decode mechanism can be considerably simpler than what would be required for a comparable capability in a multiple-pipe configuration. The added flexibility in each locus of computation being able to perform different computations has some benefit, but for the application area under consideration, most algorithms currently in use would seldom exploit this capability fully. Thus one would expect an array architecture would have its primary mode of operation be a fully synchronized (lock-step) execution where each processing element performs the same operation on its local data. Such operation eliminates the overhead of synchronizing independently functioning computers when information needs to be interchanged. Also because of performance considerations, one would expect future array processors to be able to overlap the transmission of operands among the processors while the processors themselves are computing; due to the nature of aerodynamic simulation algorithms, such a capability would be quite attractive.

There is a final area of concern in the architecture of an array computer, namely the method by which the processors are interconnected (or are connected to the large memory if a pipeline-like approach is employed). The most straightforward way is a fixed intercommunication pattern (for example, nearest neighbors in a two-dimensional grid). In this approach, more complex data flows must be simulated using multiple steps. The difficulty with this approach lies in the fact that for different formulations of aerodynamic problems (say, space-oriented versus frequency domain), different connections are needed. There are also problems with treating problems of varying sizes. The alternative approach is to have an electronic switching capability in the network itself, which would be programmable by the user to effect whatever communication pattern the problem at hand requires.

There are two aspects of a specialized computer which are independent of the particular architecture; these are reliability and programmability. A high performance computer using current technology will consist of many components. As the number of components approaches the mean time between failure of one component, the frequency of a component failure increases to the point where individual users are aware of system failures. To prevent this requires a system design whereby the system can continue functioning correctly in the presence of failed components. For memory components, this implies error detection and error correction. For processing components, this means error detection capability in some form: residue arithmetic, selective monitoring and emulation, or duplicate arithmetic units. Smaller, stand-alone computers also have a problem with reliability -- in this case not because of the large number of components, but because each problem runs a very long time.

The other general aspect of a specialized computer is its programmability. Since the processor is specialized for a reason, the programmer will have to be cognizant of the nature of the specialization and will probably be required to deal with this specialization in the syntax of the programming language; the alternative is to defeat the purpose of the specialization. On the other hand, too arcane a programming facility runs the risk of being unmanageable by a programmer, again defeating the purpose of specialization, or even the purpose of the facility's existence to begin with. This suggests that a specialized computer should be "overdesigned"; that is, memory buffers should be larger than strictly necessary to relieve the programmer/compiler/operating system of some of the difficulties in managing very large data bases; and bandwidths should be greater than strictly necessary to increase the convenience of choosing block sizes for data transmission and scheduling their movement. A machine too highly tailored runs the risk of being usable (programmable) for too narrow a range of problems, that is, of becoming obsolete with respect to the problems it can address long before it becomes obsolete in the technology it possesses to solve those problems.

SUGGESTED ARCHITECTURE FOR A SPECIALIZED FLUID DYNAMICS COMPUTER

Bengt Fornberg

Applied Mathematics 101-50
California Institute of Technology
Pasadena, California
91125

Abstract: Future flow simulations in 3-D will require computers with extremely large main memories and an advantageous ratio between computer cost and arithmetic speed. Since random access memories are very expensive, a pipeline design is proposed which allows the use of much cheaper sequential devices without any sacrifice in speed for vector references (even with arbitrary spacing between successive elements). Also scalar arithmetic can be performed efficiently. The comparatively low speed of the proposed machine (about 10^7 operations per second) would be offset by a very low price per unit, making mass production possible.

Introduction.

Future computer needs in fluid mechanics cannot be met by large conventional general purpose computers operating sequentially on one instruction at a time. Problems in 3-D flow simulations will involve too many operations and require too much high speed memory to be economical on such systems. After a preliminary discussion of speed and memory constraints, a specialized design is proposed.

Operation speed.

The two commonly proposed alternatives to sequential processing for an increase of operation speed are parallel and pipeline designs. Their main advantages (+) and disadvantages (-) appear to be

Parallel (Type ILLIAC IV or even larger arrays of processors)

+ A very large number of identical processors can be mass produced cheaply.

+ The operation speed is proportional to the number of processors and essentially unlimited.

- The fixed number of processors forms a very rigid structure. In particular:

  1. The penalty for scalar operations (or operations on short vectors) is very large.

  2. Problems have often to be partitioned or duplicated to fit the number of processors.

  3. Since wires between processors have to be minimized, data flow between processors far away may be slow and awkward.

  4. If the array of processors is very large, the computer is likely to be efficient for only a very limited number of difference schemes in very simple geometrics.

Pipeline (Type CDC STAR 100)

+ The vectors can be of any length

+ The penalty for scalar operations is very reasonable.

+ The main high speed memory is mostly referred to in sequential sections instead of in a random manner. This may allow the use of very inexpensive devices (bubbles, CCD, electron beam, etc.) which are fast only for vector references.

- The operation speed is more limited than it is for giant parallel arrays.

High speed memory.

Present large computers have hierarchies of memory including small high speed memory (core or semiconductor) and a large slow memory (discs). Such a memory hierarchy works well for general purpose computing since different sets of data are used with different frequency. Hierarchies are not suitable if all elements of a very large data base are referred to with high frequency and equally often. A general purpose system is normally considered to be reasonably balanced in speed and memory size if it takes about 1 second to access all the words in the memory. For finite difference methods in fluid mechanics we normally need very large grids with few operations per grid-point. Large linear systems also have few operations per entry in large coefficient matrices. A more reasonable time for a system designed for such applications might be 100 to 1000 seconds. Present giant machines have developed in the opposite direction. They have very small main memories compared to their processing speed. ILLIAC IV, CRAY -1 and STAR 100 are all in the range .001 to .05 seconds.

Suggested machine design.

We believe the key to a future machine for fluid mechanics must lie in the use of very cheap and very large (> 100 M words) main memory. In most cases, results from runs are not needed urgently (exceptions are real time calculations like weather prediction). An alternative to one big superfast machine would be to have many less fast (and much cheaper) machines. Each machine could be dedicated to a problem and run on it for a long time (up to some months in extreme cases). Such execution times are probably still much less than the design, programming and debugging time for large programs.

The memory of this computer must be fast for vector references. Spacing other than one must also be possible without loss of speed (for example running row wise over a matrix stored column wise). This can be achieved in the following way: Suppose we have a large number of shift memories (implemented for example by bubbles, CCD or electron beam devices) consisting of continuously circulating loops of, for example, 131 words each. (131 is just bigger than the useful lengths 2^7 and $2^7 + 1$ and is a prime (which will turn out useful)). At one position of the loops, there is a read and write station. Let us assume one full shift cycle all through this loop takes 50μs. This is how long we may have to wait if we want to read a scalar from the memory. If we want to read all the 131 words to a fast random access buffer, the total time would again be 50μs. No waiting would be needed in this case since we can transmit the elements immediately as they become available. In a few years time it may be feasible to put some 200 loops on a chip for a cost of < 10$ per chip. If so, a 100 M word memory would cost < 40K$. We can number the 100 M words 1, 2, 3, . . . , 10^8 and put them in the shift registers as in figure 1. Below the shift registers is a 'switchboard' which feeds the outgoing pipeline (some top levels of switches can be put on the memory chips). The delays due to many levels of switches are not critical in pipeline operations, in particular since transfers are to a buffer memory and not directly to a processor. If we want to transfer a vector of length 131 starting from word number 1, the first shift register is fed to the pipeline (all switches in fixed position connecting the pipeline to the first read/write head). Assume now we want to transfer 131 words with any spacing not a multiple of 131, for example words 1, 4, 7, 10, ... , 391, with spacing 3. At each shift position one and only one of these numbers will be at a read/write station (here we use the fact that 131 is a prime). By turning the switches properly the words 1, 4, 7, 10, . . . 391 (in scrambled order) are fed through the pipeline. The numbers arrive to a random access buffer and the order is unscrambled when they are stored. We see that any vector of length less than or equal to 131 words with any spacing (apart from multiples of 131) can at any time be transferred in 50μs or less to the buffer memory. Since the whole switchboard can be duplicated, several (for example, 4 outgoing and 2 ingoing) pipes can be handled simultaneously. This would allow a continuous transfer rate of some 8.10^6 words per second. A pipeline processor with very moderate speed (5-10 times 10^6 operations per second) can work on the buffer memory. This buffer memory is similar in idea to the 8 64-word registers in CRAY-1. Here the buffer should be much larger but also much slower (i.e. cheaper). If the scalars which are in current use are kept in the buffer, the penalty for scalar operations would be very small.

Compared to present giant machines with 100 M operations/sec, 1 M word memory at a cost of 10 M$, the proposed machine may (in large production) have a speed factor 1/10, memory factor 100 and cost factor 1/100. To minimize system complexity (and expensive system software) we do not think machines of this kind should be synchronized into any form of array system. They can be used individually placed at an ordinary computer center using available peripherals when occasional input or output is needed. A very large computer center (about 20 M$) based entirely on these computers could have a conventional central processor to handle I/O, compilations and basic system tasks for some 50-100 individually working machines. The different machines would be dedicated to different problems (or same problem with different parameter values, initial conditions etc.)

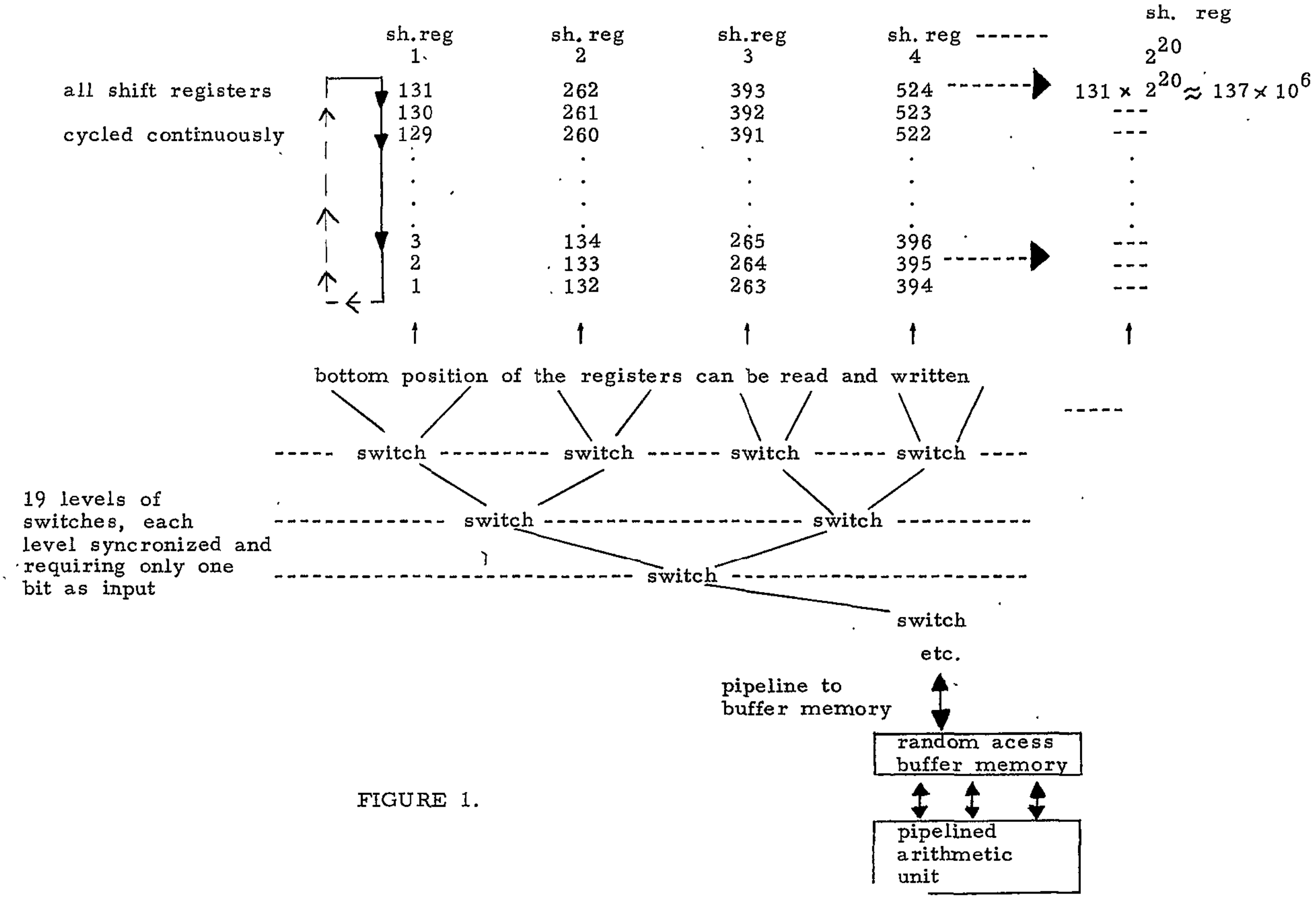

sh.reg 1
sh.reg 2
sh.reg 3
sh.reg 4
sh. reg 2^{20}
all shift registers
cycled continuously
131
130
129
3
2
1
262
261
260
134
133
132
393
392
391
265
264
263
524
523
522
396
395
394
$131 \times 2^{20} \approx 137 \times 10^6$
bottom position of the registers can be read and written
switch
switch
switch
switch
switch
switch
switch
switch
etc.
19 levels of switches, each level syncronized and requiring only one bit as input
pipeline to buffer memory
random acess buffer memory
pipelined arithmetic unit

FIGURE 1.

MICROPROCESSOR ARRAYS FOR LARGE SCALE COMPUTATION

William H. Kautz
SRI International
Menlo Park, California

ABSTRACT

An important new direction in computer architecture centers around the achievement of very high computational power (capacity, speed and reliability) through the use of tens of thousands of microprocessors, micromemories, and switch modules, all interconnected into a large homogeneous network using one of certain advanced connection schemes. When surrounded and supported by conventional computers and memories, such a machine holds potential for out-performing both conventional and array-based computers of the mid-1980's by one to two orders of magnitude, at least for particular classes of applications amenable to high parallelism, such as aerodynamic simulation. The homogeneous feature of this machine concept also implies size extendibility, fault tolerance, and improved flexibility to handle a variety of algorithms of interest.

This architecture achieves its very high speed and throughput largely through a combination of extreme parallelism, well-scheduled data streaming, and high-speed switching. The parallelism is achieved mainly through the use of 10,000 or more conventional identical microprocessors, each of which has a small amount of memory, and a comparable number of high-speed micromemories. The microprocessor array is supported in turn by a hierarchy of minicomputers and larger computers, whose task it is (a) to issue broadcast commands to each microprocessor, according to whichever one of three or four computational regimes is appropriate at that point; (b) to control how many and which grid points of the solution space are assigned to each microprocessor (in order to uniformize computation rate over the array and to circumvent defective components), and (c) to control the flow of data from backup memories through the micromemories. Very high-speed data streaming is achieved by a well-designed memory hierarchy composed of small, high-speed semiconductor random-access micromemories; small to medium-sized linear memory modules, such as EBM or CCD circulating memories, which have very high bandwidths despite their relatively large access times; back-up units consisting of head-per-track discs; and ultimately conventional disc systems and parallel-access tape-based massive memories. A new, high-speed switch module is employed between microprocessors and micromemories, and within part of the memory hierarchy itself.

Past work has demonstrated the theoretical feasibility of this novel type of machine architecture. Current work is addressing the design of technologically efficient interconnection configurations and the development of new computation algorithms that are especially efficient for highly parallel computation.

INTRODUCTION

The rapid evolution of integrated-circuit semiconductor technology presents a challenging opportunity to construct in the mid to late 1980's a large computer out of tens of thousands of microprocessors and memory modules. A brief study of some of the more crucial aspects of this potentiality reveals that there is a definite possibility that such a "core" machine configuration, when appropriately supported by a peripheral base of more standard processors and memories,has the potentiality for outperforming conventional and array-base super-computers on a number of particular applications, including large fluid-flow problems such as wind-tunnel aerodynamic simulation.

The purpose of this paper is to briefly expose the reader to this novel architectural alternative--its principal features, the design issues implied by it, and the major problems that need to be solved before such a machine can be considered to be competitive from an engineering standpoint.

This machine concept has not been greatly explored in the past, probably because it falls outside of the mainstream of thinking about how to solve information and data processing problems using "computers." Indeed, to adopt an unconventional architecture means that other aspects of computer design and development may have to change as well, not the least of which would be the algorithms for problem solution, both high- and low-level languages, and the software generally.

Note the inverse approach taken here. Normally one starts with the computational problem and asks for the best available technology to solve it. Instead, we start here with a technological capability, as yet unexplored, and ask if there is a worthwhile problem that it will help to solve. While this view is neither necessary nor sufficient as a design philosophy, it can be a very helpful adjunct to escape from traditional ways of thinking about a problem, and to creatively expand the range of possible solutions.

MACHINE CONCEPT

The approach espoused here is to organize the computation around a richly interconnected network of tens or hundreds of thousands of single-package microprocessors and memories. These components will be high production, low-cost items, similar to the units available today but not necessarily merely faster and larger versions. of them. The low cost will assure that the central "core" of the new machine has a hardware cost well below $1,000,000 in the mid-1980's. Extrapolation of current trends suggests that the cost-speed trade off will have advanced by the mid-1980's to the point where very high speeds (perhaps 1/2 the maximum available) can be achieved for a small fraction of the cost of maximum speed. It behooves the designer to try to find a way to achieve his high computation capacity by using a larger number of slower circuit packages.

It is easy to verify that the total computing power, measured in millions of floating-point operations per second (MLOPS), of such a mass of hardware is very large indeed--well above that of other proposed supercomputers. The question that remains is, is there a way to organize and interconnect these components so as to provide the balance of speed and capacity needed for programs and data, so as to convert this massive potential computing power into an effective computation rate for the solution of practical problems. To accomplish this objective, conventional ways of resolving design issues may have to be changed. It may be necessary to develop new computational algorithms as well.

This machine concept may turn out eventually to be workable for general-purpose computation. One can imagine a massive, virtually homogeneous array of processor-memory cells carrying out problem decomposition and the execution of tasks, subtasks, sub-subtasks, etc. in a highly parallel mode. However, the problems of how to create an anthropomorphic model for information processing so that it is effective and efficient are very great. Some restrictive assumptions are needed to reduce these problems to manageable proportions.

The most natural assumption is to restrict the machine application to one characterized by extreme parallelism in computation and data, and by a largely predictable data flow. These conditions are satisfied by many fluid flow problems and by aerodynamic simulation based on the Navier-Stokes equations in particular. A machine configuration appropriate to the solution of this problem will now be described.

MACHINE DESCRIPTION

This machine configuration is shown in Figure 1. The core machine consists of three kinds of elements. First, there is a very large number of essentially identical microprocessors (μPs)(which may be miniprocessors by the time such a machine is actually designed and built). Each μP contains the basic microprocessor capabilities, and includes a modest amount of memory of its own for working data and program storage, plus a partially changeable microprogram control memory. Second, the core contains a comparable number (or perhaps twice as many) of micromemories (μMs). These random-access units constitute the main working store. Third, the μPs and μMs are crossconnected in groups μPs at a time, say) by a set of permutation switches (PSs). Most of the computer's computational intricacy and power are contained in these three elements.

The PSs provide an interconnection capability sufficient for the following three purposes: (a) to share and exchange data between nearby μPs, to the extent that this is needed for the particular algorithms and programs used; (b) to permit rapid changes of μP-μM connections, corresponding to the assignment of 1 to 4 or 5 grid points of the computational array to a single μP; and (c) to provide reconfiguration of μPs and/or μMs in the event of the failure of a single element. Feature (b), in conjunction with the changeable microprogram capability of the μPs, permits different computational regimes (laminar flow, turbulence, boundary and interface conditions, etc.) to be in effect in different portions of the computational array, without paying penalties in speed that would be required if the entire array had to wait upon the slowest grid-point computation. The flexibility in assignment of grid points to μPs may also allow non-rectangular regions to be handled efficiently, but this feature is highly problem-dependent and may not turn out to be achievable.

The μMs are backed up by a bank of linear memory modules (LMMs) whose function it is to stream the appropriate grid-point data to the μMs in the three dimensional directions, corresponding to the three passes required in an algorithm based on separation of coordinates. The LMMs contain special switching, to be described shortly, for coordinate rotation. The LMMs are supported in turn by back-up memories (BUMs), probably head-per track discs, which serve to exchange data with the LMMs in correspondence with successive time steps. The BUMs are supported in turn by archival memories, in the form of mass storage devices.

Corresponding to this memory hierarchy, the processor hierarchy at the left side of Figure 1 provides program control of the μPs, and controls the data flow and switching associated with the μMs, the LMMs, and the switching paths in the PSs. Each array processor (AP) serves a modest number (perhaps 128) of μPs. It supplies to each the variable portion of its microprogram code, appropriate to the computational regime and appropriate to the grid points that are mapped onto a particular μP at a particular time. It also sets up the switching paths in each PS. It then distributes broadcast commands to the μPs, again in accordance with the various computational regimes. The control processors (CPs) manage the operation of the APs and the high-level memory transfers between the LMMs, the BUMs, and the archival memories. Finally, a host computer supervises the CPs and the system as a whole.

MACHINE ELEMENTS

It is anticipated that the μPs and μMs will be available when needed as high-production LSI components. The switch module PS is not required for existing computer systems, and is not yet on the market. However, it is logically simple and is feasible to develop and manufacture. The function and gate circuitry of a basic 2 X 2 switching cell and the network of cells required for an 8 X 8 PS are shown in the portions of Figure 2. Note that each elementary cell contains a binary storage element whose state determines whether the cell inputs and outputs are connected in straight-through or crossed manner. The set of all such cell flip-flops in PS may be regarded as a storage register whose contents defines the overall permutation of inputs to outputs of the PS as a whole. Algorithms for deriving the register contents appropriate to a prescribed permutation are available and are not difficult to implement.

If b-bit bytes rather than single lines are switched within a single switching cell and PS module, the storage register may be shared. Simple calculations lead to the following costs:

$$G = (6b + 3)n(\log_2 n - 1) \text{ gates (for } n > 2)$$
$$T = 2nb + 4 \text{ terminals}$$
$$D = 4 \log_2 n + 2 \text{ gate delays.}$$

The value of b should be doubled for bilateral signal transfer. However, note that these modules, like most memory modules, can be bit-sliced. They are therefore stackable for switching any number of bits per word without incurring additional delays or increasing the number of terminals per module.

Table 1 lists the costs of a few PSs that are probably the most likely alternatives for practical packages. Since these modules are terminal limited rather than gate limited and must operate at maximum speed, they would be realized today with ECL technology.

n	b	uni/bil.	G	T	D
2	8	uni.	51	38	2
2	8	bil.	99	68	2
16	1	uni.	432	38	14
8	1	bil.	450	38	10
8	4	uni.	432	68	10
4	8	uni.	204	68	6

Costs of Typical PSs

Table 1

The LMM, on the other hand, is not terminal limited. It can readily accommodate the extra terminals and the small amount of switching needed for a limited degree of data exchange in the normal course of data transfer, even if such transfer is carried out in parallel using 8-bit bytes. A possible LMM is shown in Figure 3. It contains two linear memories (LMs), which may be thought of as CCD memories in today's technology. The module contains two data inputs (south and east) and provides two data outputs (north and west). A small permutation switch PS, whose paths would be set up in between the movement of data blocks, controls the flow of data from the module inputs and the two LMs to the module outputs and back to the LMs. As indicated in the bottom line of Table 1, a switch PS for 8-bit bytes can be implemented with as few as 204 gates--a trivial addition to a large integrated-circuit package.

That such a two-input, two-output LMM is sufficient for transposition of a data array is suggested by the diagram in Figure 4. The numbers within each LMM represent the element indices for a 4 × 4 matrix folded down once along its principal diagonal. By setting the internal switching of the non-diagonal modules in accordance with that shown in Figure 4b, the data flow corresponds to circulation along rows. If the switching of Figure 4c is used instead, circulation by columns results. This simple arrangement may readily be extended to matrices of arbitrary size and to three-dimensional data arrays (co-ordinate rotation). One may also use arrays that are fully rectangular rather than triangular, to avoid the special diagonal elements.

Thus; transposition of data arrays can be achieved in the natural course of data transfer, by storing the data in circulating memories to which a small amount of switching has been added.

MACHINE FEATURES

Probably the most outstanding feature of the proposed machine configuration is the flexibility provided in the interconnection network between the microprocessors and their corresponding memories. In view of the expected low cost of even high-speed switching, there is no need to constrain the switching pattern to that of a two- or three-dimensional rectangular array having only nearest-neighbor connections. Moreover, reconfigurability is probably essential for any machine architecture having thousands of parallel-acting elements, since the mean time between failure will very likely be significantly longer than the duration of the most difficult computation problems.

This interconnection flexibility can also be utilized to improve the efficiency of computation, by varying the assignment of grid points to processors. That is, several grid points at which simple computation cycles are in progress may be handled on a single processor, while grid points at which a long computational cycle is required can be assigned one processor each. As a result, the schedule of computation is not necessarily dominated by the most complex computational regime, and the computation rate is made higher and more uniform over the array than it would otherwise be,

The flexibility might also be applied to reduce the precision of calculation, by widening the grid-point spacing in those portions of the array where less accuracy is required. However, such modifications would also change the pattern of data flow and may not turn out to be practical.

Finally, the combined flexibility in interconnections and in microprogram control within the processors may be important factors in machine development, for accommodating new computation algorithms for aerodynamic simulation, which are continually being improved and extended. Since the nature and degree of these improvements and extensions cannot always be accurately estimated in advance, it is very important to include as much flexibility as possible at the time of machine design.

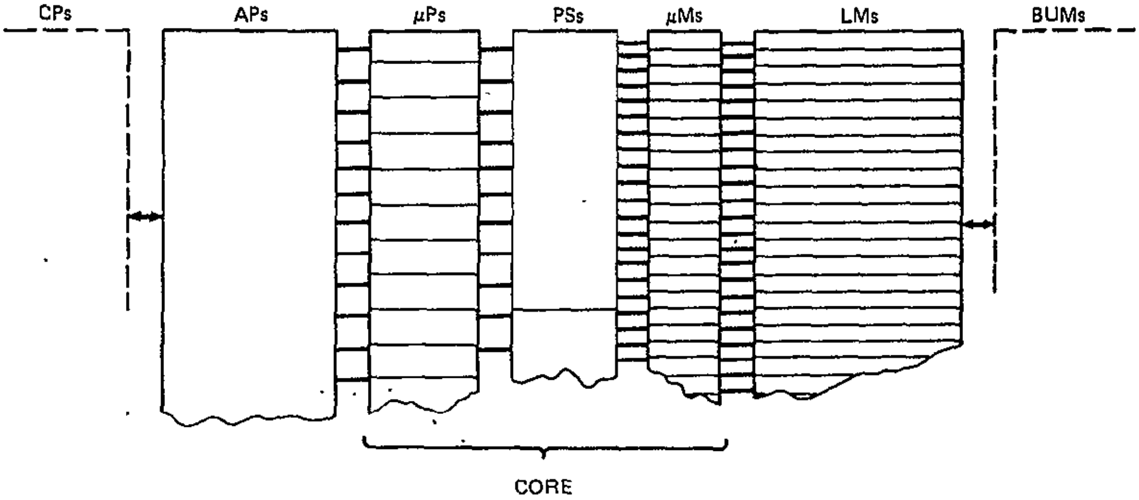
CONCLUSIONS

Rough calculations indicate that competitive performance (throughput, speed, and reliability) can be achieved with the suggested machine configuration, using the hardware technology expected to be available in the mid-1980's--but only if certain problems in software development, data management, microprogram control, and interconnection configurations can be solved. Some algorithm development may be needed as well, and should be attempted in any case for whatever improvements may turn out to be possible for aerodynamic simulation. These problems are deemed to be neither simple nor very difficult, and therefore constitute good research risks.

Software development includes here the design of a high-level language for expressing algorithmic variations and combinations and for formulating problem conditions; the design of a time-efficient operating system for the hierarchy of supporting processors; and the design of one or more lower-level languages for interfacing the "core" machine with the supporting peripheral computers. The challenge of data management is to achieve a sufficient degree of flexibility in choosing and controlling data structures appropriate to the problem class of interest, in order to handle as wide as possible a range of applications, simulation models, and algorithmic alternatives, without unduly increasing the cost of this flexibility, and without reducing performance significantly. Preliminary design effort should also treat the problem of fault tolerance, including both hardware and programmed error detection, reconfiguration control, and maintenance.

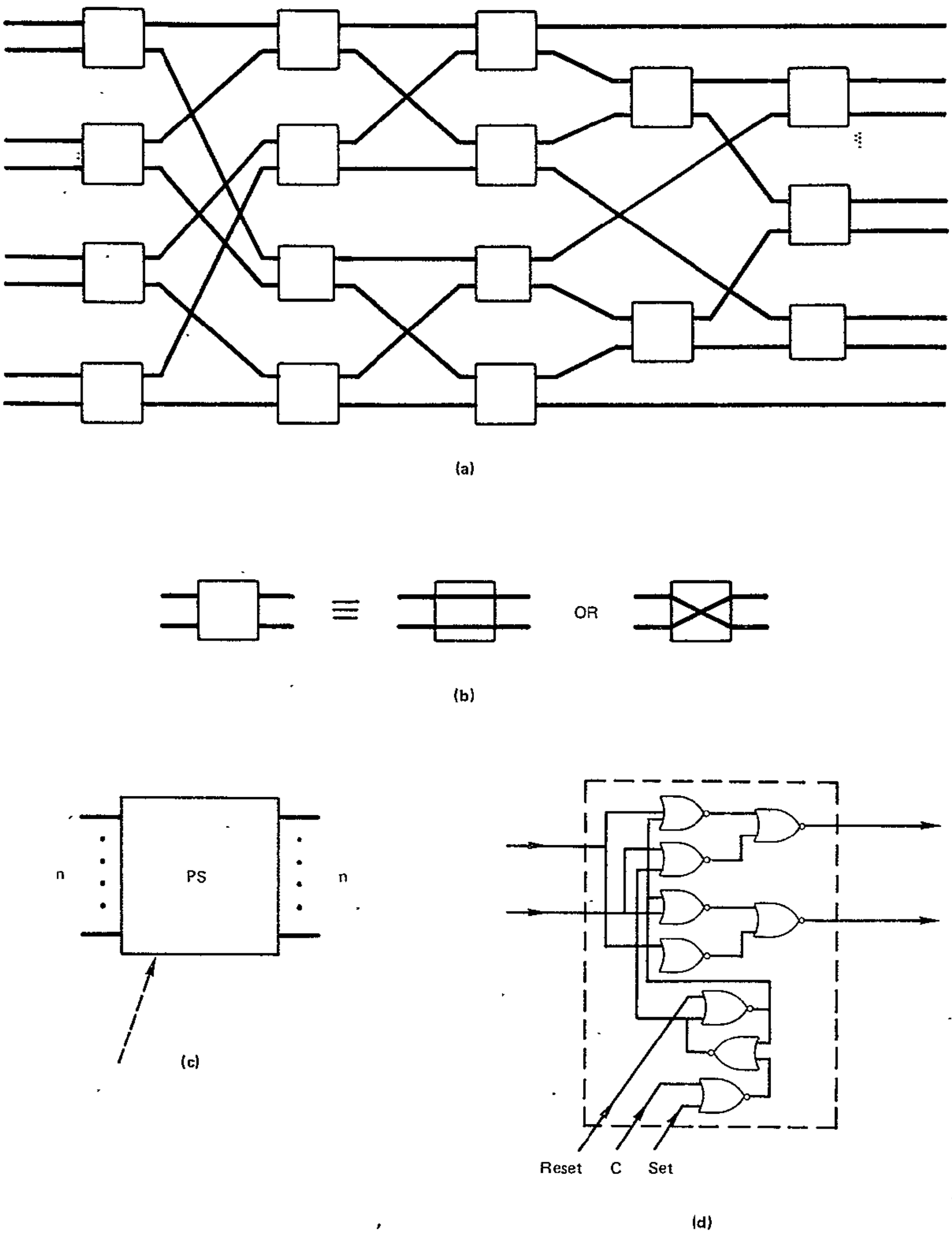
Many problems in research and preliminary design remain before engineering feasibility of the suggested machine architecture can be established. Even if this effort is successful, additional design and development effort will be required before realistic cost estimates can be made. Nevertheless, the novel approach treated here appears to have considerable potential for improving the estimated performance of a supercomputer for Navier-Stokes simulation in the mid-1980's, compared with even optimistic extrapolations of the performance of present day supercomputers such as ILLIAC-IV, CRAY-1, and STAR-100, all of which have more conventional architectures than proposed here. In addition, the processing and data-flow flexibility provided by the new approach, while not yet proven, offers a way of capitalizing upon future improvements and algorithms in technology and a protection against unforeseen difficulties that may arise in the design and development.

It is recommended that this architectural alternative be given serious consideration in long-range planning for future computers for aerodynamic simulation and related applications.



TA-790522-3

FIGURE 1 ORGANIZATION OF THE PROPOSED COMPUTER



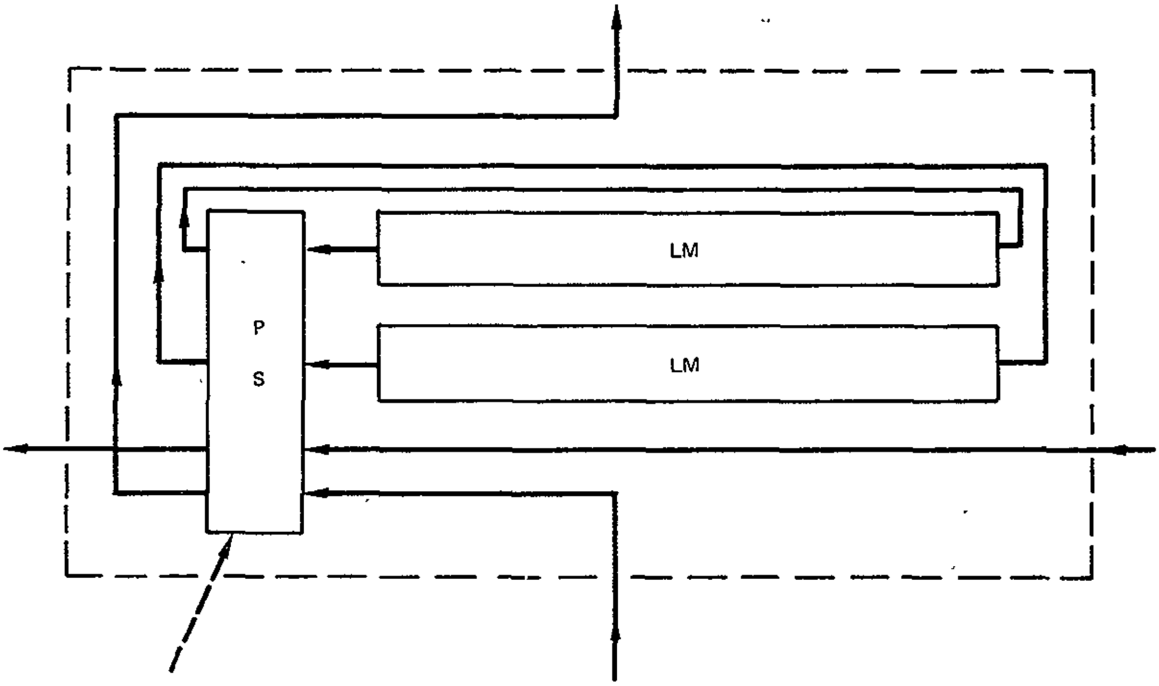
(a)

(b)

(c)

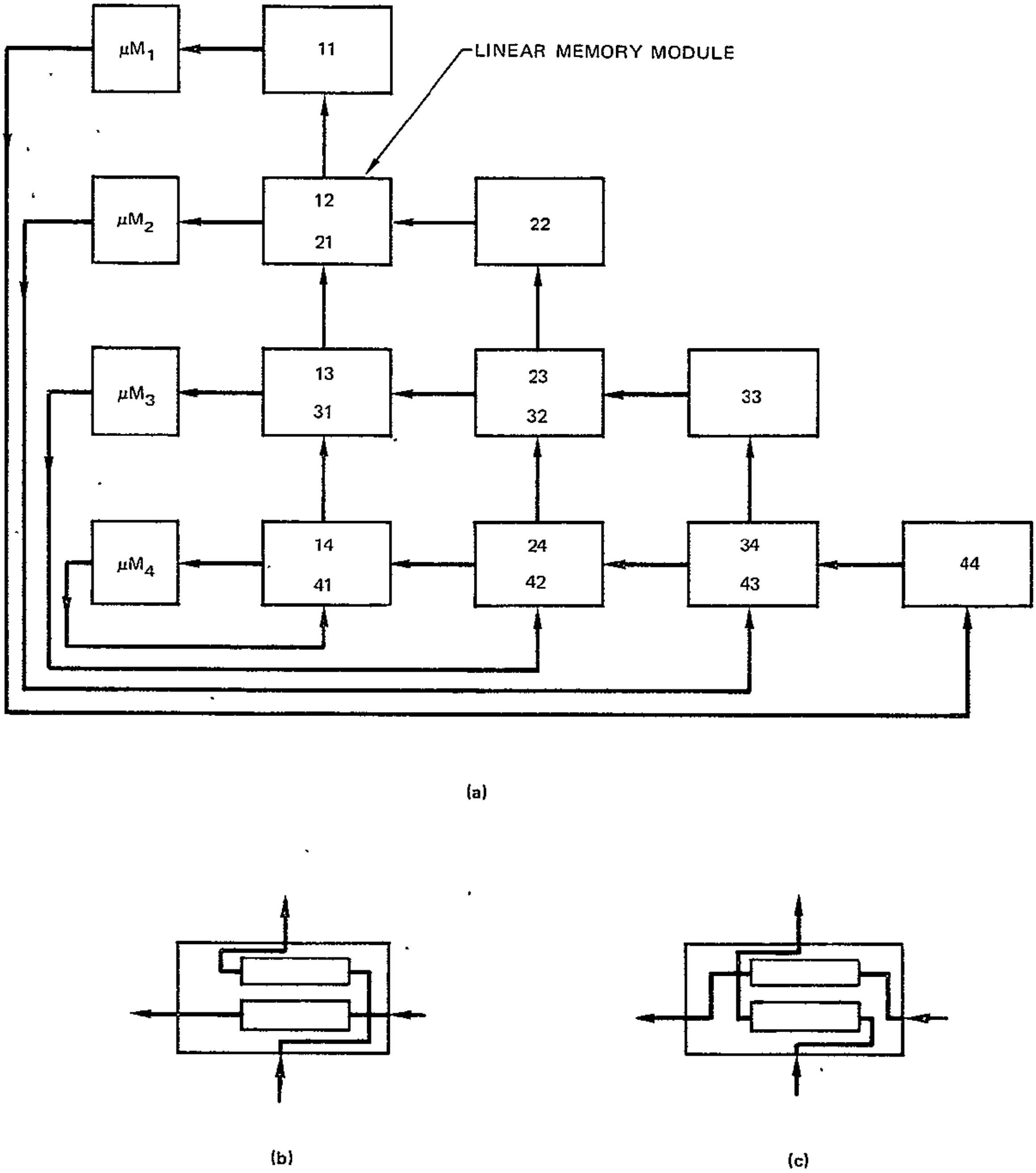
(d)

FIGURE 2 PERMUTATION SWITCH



TA-790522-5

FIGURE 3 LINEAR MEMORY MODULE



TA-790522-6

FIGURE 4 TRANSPOSITION ARRAY

D39

N78-19817

FEASIBILITY OF A SPECIAL-PURPOSE COMPUTER TO SOLVE THE NAVIER-STOKES EQUATIONS

E. C. Gritton
W. S. King
I. Sutherland
R. S. Gaines
C. Gazley, Jr.
C. Grosch
M. Juncosa
H. Petersen

The Rand Corporation
Santa Monica, California

ABSTRACT

This paper presents the preliminary results from an investigation of the feasibility of designing and developing a special-purpose computer for numerically simulating the onset of turbulence by solving the Navier-Stokes equations. It is concluded that orders-of-magnitude improvements in computer performance can be realized with a parallel array of thousands of fast microprocessors. In this architecture, wiring congestion is minimized by limiting processor communication to nearest neighbors. The study shows that when certain standard algorithms are applied to a viscous flow problem and existing LSI technology is used, performance estimates of our conceptual design show a dramatic decrease in computational time when compared to the CDC 7600.

I. INTRODUCTION

The possibility for obtaining direct numerical solutions of the Navier-Stokes equations for turbulent flows and flows undergoing transition to turbulence has intrigued the fluid dynamics community for some time. Recently, Case et al. (1973, 1975), reconsidered the question: Is the direct numerical simulation of turbulence possible with the most modern computers? They concluded that the construction of a high performance general-purpose computer with conventional architecture to do large Reynolds number simulations appeared impractical for the next 10-15 years.

Although we are in the era of large computers that will process hundreds of millions of instructions per second, researchers are interested in performing more accurate simulations of physical phenomena creating a demand for still larger computers. Even using large-scale integrated micro-circuits, improved heat-extraction techniques, innovative hardware design and creative numerical techniques, improvements only on the order of factors of 2-5 are anticipated for a general-purpose computer with conventional architecture in the early 1980s. This performance is inadequate to simulate numerically many fluid dynamic problems without questionable phenomenological modeling.

Communication within the computer also limits performance improvement, for instructions, data, and results are transmitted at a finite rate over a finite distance. Sequential handling of the massive amount of data generated in a super-computer is thus rate limited. Attempts to overcome this problem in large sequential architecture have produced more complex hardware designs.

To penetrate this performance barrier, some researchers have proposed the concept of a large computer consisting of a parallel array of thousands of fast microprocessors. Data are divided and distributed among a number of microprocessors that simultaneously perform identical operations. The architecture of such a computer is strongly dictated by the repetitious but parallel characteristics of the problem to be solved. Thus, one should incorporate the nature of the simulation problem in the design of the hardware, the numerical analysis, and the development of software for such a computer. Numerical simulations that require many different types of operation on a single set of data could not exploit the advantages of this parallel arrangement since many processors will stand idle while a few are working.

With the limitations of current architecture in mind, Rand (Gritton et al., 1977) conducted a corporate funded study in collaboration with members of the California Institute of Technology and Old Dominion University faculties to investigate the feasibility of developing a special-purpose parallel array computer, which would simulate viscous flow around bodies with realistic geometries and include some aspects of the development of turbulence. The Rand study (Gritton et al., 1977) focused on special-purpose parallel array computers because, as we noted earlier, we anticipate that even in the early 1980s the fastest serial general-purpose computer will be insufficient to cope with such simulations.

Specialization of the computer design to a specific class of problems also maximizes the advantages of parallelism since the specific numerical methods can be built into the hardware. Economies in system cost can be realized in a special-purpose computer because the design and development of complex operating hardware and software are greatly reduced. The simple operating system of a special-purpose computer also implies that computational overhead is substantially decreased. Other economies may be derived from specialization when problem optimized hardware and software are incorporated in the design study. Rand has evolved an approach which accomplishes this for a class of problems in fluid mechanics.

This paper highlights the major conclusions of our preliminary study, and summarizes the results from our continuing program as well as those from the Rand Workshop. A conceptual design of an array processor for solving the Navier-Stokes equations is discussed and estimates of performance are presented.

II. THE RAND CONCEPT OF A NAVIER-STOKES COMPUTER

The Rand concept is to design the computer architecture around the mathematical structure of a specific set of differential equations, the Navier-Stokes equations. Applications to other specific problem areas and differential equations may be feasible. To insure maximum performance improvement, the design of the computer should be carried out by a multidisciplinary team consisting of experts in computer architecture, numerical analysis, algorithms, and software systems. This will insure that the machine architecture is tailored correctly to the simulation requirements.

The initial Rand study of a preliminary concept for a Navier-Stokes computer took place between October 1976 and March 1977. This conceptual design of a machine consisted of an array of 10,000 identical microprocessors that are arranged in a 100 × 100 matrix. To reduce wiring congestion, the communication for each processor is limited to communication with its nearest neighbors and with the control processor or processors. Each processor includes some memory space for storing algorithms and carrying out simple calculations. This feature is important because it permits individual processors to be distinguished from each other not only in their position in the array and the data they contain, but also in terms of the specific functions they perform. Three-dimensional problems are solved in having each processor represent a point in a two-dimensional plane, and the storage on the processor is used to represent data from nearest neighbors in the third dimension.

The potential benefits of this parallel architecture are enormous. We estimate that each processing element can carry out a 64-bit fixed-point multiplication in 5 microseconds (double precision multiplication, each word 32 bits long). Though this is quite slow compared with the speed of today's best computers, an array of 10,000 such elements performs simultaneously 10,000 multiplications in that time or approximately 2 billion multiplications per second. We also estimate that each processor could transfer one word to a neighboring processor in about 3 microseconds. Again, this is quite slow by today's standards, but with 10^4 processors the data rate is about 3×10^9 words/sec, or about 10^{11} bits/sec. This is quite high when compared to the memory to CPU data rate of even the most powerful conventional computers.

These large computation and communication rates can lead to a very large performance-cost ratio given that integrated circuits of the type described may be produced in the near future for a cost less than or about equal to $100 per chip. An important characteristic of the array of processors just described is that, given some reasonable amount of memory per chip, a large entire problem can fit on this computer at one time. Thus, there would be no need either to break the solution process up into many sequential elements or to reload the array frequently. The resultant saving in serial computations and communications time and other overhead contributes substantially to the power of such a machine.

There are general problems in the design of any array computer utilizing LSI technology. The chip design must be as simple as possible, particularly the interconnections on the chip. If the on-chip wiring is highly complex, the number of bits per chip must decrease because a large fraction of the chip area is taken up by wiring and the cost of a chip is increased. If all N chips were interconnected to all other chips, on the order of N^2 wires would be required. If $N = 10^4$, the wiring density becomes extremely high and costly. In order to reduce cost, increase computational speed, maximize reliability and reduce wiring congestion, the number of interconnections must be kept to a minimum. Of course, it is possible to add a small number of additional interconnections, but at no point can many wires cross or lie parallel.

The Rand concept stresses a large degree of parallelism, while simultaneously emphasizing simplicity in processing element design, array design, and data streams. Another attribute is that it takes advantage of inexpensive components to minimize system cost.

The conclusions of Rand's initial study suggested that a special-purpose, parallel-processor machine capable of important simulations might be technically and economically feasible in the early 1980 time period. To explore further the practicality and timeliness of this concept, Rand held a workshop on March 9-10, 1977. Researchers from the computer technology, numerical analysis, and fluid dynamics communities participated in the critical evaluation of this idea.

The workshop participants generally agreed that appropriate technology now exists to consider a machine of the nature proposed capable of attacking problems of theoretical and practical importance. It would have a computing capacity at least an order of magnitude superior to the largest current computers and could be built within the next 3-5 years at a cost substantially less than current large-scale computers.

Both adequate algorithms and sufficient numerical experience are available for the solution of the relevant partial differential equations. The family of designs considered in the workshop offer the potential for efficient direct Navier-Stokes simulations of nonlinear laminar instabilities, boundary-layer transition and a simulation of large-scale turbulent flows at high Reynolds numbers using sub-grid modeling.

III. PARALLEL PROCESSOR COMPUTERS

The concept of an array computer is quite old (see, for example, the discussion and description of the Von Neumann array computer in Thurber, 1976). Far fewer array computers than sequential computers have been built in the past because of system complexity and high cost. Individual processors were expensive, and reproducing large numbers of components increased costs plus control overhead.

Recent advances in large-scale integrated circuit technology and reduction in manufacturing costs have removed the economic barrier to the use of large numbers of microprocessors in the array computer. Once the circuit design and setup costs are incurred, chip reproduction costs are small. By using current technology in the chip design, as developed for widespread use throughout the electronics industry, the development costs and technical risks associated with construction of the array machine are reduced without compromising the effectiveness of the design.

ILLIAC IV is the best known general-purpose array processor (Thurber, 1976). It can be viewed as an 8 × 8 array of cells (processing elements, PEs) arranged in a grid communicating with nearest neighbors and end around. Each PE is a rather sophisticated general-purpose computer and has 2K 64-bit words of memory.

Another example of the array processor concept is the ICL Distributed Array Processor (DAP) (Flanders, et al., 1977) which exhibits a large degree of design simplicity and processing efficiency. DAP was built quite recently and is a 32 × 32 array of computing cells, each with limited memory, and attached by a bus to a master memory. The design is, in some ways, quite conservative. All communication and computation is serial by bit and the technology used in construction is about 8 years old. Bit serial processing has storage advantages. But, ordinary arithmetic operations must be built up by software from single bit operations. This considerably slows arithmetic operations. A few test problems have been run on DAP. They were found to run from two to ten times faster than on an IBM 360/195 or CDC 7600. For one problem, DAP was reported to be about 3 times faster than a CRAY-I.

We compared the estimated performance of a 64 × 64 DAP with our 10,000 array processor computer for one problem: the solution of Laplace's equation on a 64 × 64 array using Jacobi relaxation. The DAP group compared the measured performance of ILLIAC IV and an IBM 360/195 with the estimated performance of a 64 × 64 DAP. The times required to calculate 10 iterations were estimated to be (Flanders, et al., 1977):

IBM 360/195	67.2 ms
ILLIAC IV	17.2 ms
DAP	3.3 ms

Using the type of analysis, add and multiply times, etc., described in Gritton, et al. (1977), we estimate that the corresponding time for 10 iterations on a 64 × 64 subset of our 100 × 100 array computer would be 0.08 ms. We believe that the success, to date, of the ICL-DAP is a strong argument for continued and more advanced development along these lines. DAP is an exploratory laboratory device and not an optimal design. Therefore, we anticipate that advancements in LSI technology and design techniques can lead to vast improvements over the already impressive DAP performance.

IV. PARALLEL PROCESSING APPLIED TO NAVIER-STOKES EQUATIONS

However, to exploit the full power of the array computer concept, the numerical algorithms must match the machine architecture. The challenge in applying the parallel processor concept is to couch the problem to be solved in terms which do not require long distance communication since this is fundamentally expensive from the computer architecture point of view. Also, serial computation should be avoided as much as possible since this does not make effective use of the parallel array hardware. The extent to which one is able to successfully develop algorithms will determine the magnitude of the computational gains.

During Rand's initial study a very simplified viscous flow problem was investigated to determine if there were any major obstacles in using parallel processing for flow problems. The model problem consisted of the three-dimensional solution to the Navier-Stokes equations for viscous incompressible flow of a constant density fluid in a boundary layer adjacent to a rigid wall. The Navier-Stokes equations in primitive variables and the Poisson equation for pressure were Fourier transformed in the span-wise direction. For the remaining two-space dimensions and time variable, the equations were approximated by using second-order-accurate finite-difference schemes. A low-order Adams-Bashforth method was used to solve the velocity equations and several point-wise relaxation schemes were considered for solving the Poisson equation.

The results of the analysis of the model problem as applied to a 100 x 100 array of cells with 128 Fourier components in the cross-stream direction showed that the problem could be solved at a rate of about 0.35 sec per time step. We estimate that the same problem run on a fast conventional computer (e.g., a CDC 7600) would take approximately 82 sec per time step.

We found that velocity calculations were amenable to parallel computation but that the pressure calculations (Poisson equation) were more difficult and would require more than 60 percent of the computing time. This implies that computational performance improvements are being paced by the pressure calculations and that experimentation may lead to substantial savings.

V. SUMMARY

The Rand study to date has not been exhaustive. However, it has been shown that the application of parallel processing to solve the Navier-Stokes equations is a viable concept. Our study has shown that certain standard algorithms can be employed in conjunction with nearest-neighbor communication to effect dramatic decreases in computational time. These results support the conclusion that the concept of a parallel array computer for solving the Navier-Stokes equations merits further investigation. Variations in flow geometries, flow parameters, and algorithms were not examined. Some significant topics that remain to be explored in any future study are:

1. COMPUTER ARCHITECTURE AND TECHNOLOGY
    a. Processing elements
        (1) Memory requirements
        (2) Multi-chip versus single-chip designs
        (3) Reliability
    b. Control
        (1) Error and status diagnostics
        (2) Input-output considerations
2. NUMERICAL ANALYSIS
    a. Efficiency of particular algorithms
    b. Relative merits of various techniques such as finite difference, pseudo-spectral, etc.

The experiences obtained with ILLIAC and DAP along with discussions at the Rand workshop have clearly defined some of the uncertainties in designing and developing a Navier-Stokes computer. We are advocating a multidisciplinary research and design program to eliminate these uncertainties and arrive at a technically sound design for a Navier-Stokes computer. The development of analytical and numerical methodologies should be sufficient to assess the effectiveness of parallel processing and to provide realistic estimates of reliability and performance.

REFERENCES

Case, K. M., F. J. Dyson, E. A. Frieman, C. F. Grosch, and F. W. Perkins, *Numerical Simulation of Turbulence*, Stanford Research Institute, Technical Report JSR-73-3, 1973.

Case, K. M., A. M. Despain, E. A. Frieman, F. W. Perkins, and J. F. Vesecky, *Problems in Laminar Flow-Turbulent Flow*, Stanford Research Institute, Technical Report JSK-74-2, 1975.

Flanders, P. M., D. J. Hunt, S. F. Reddaway, and D. Parkinson, "Efficient High Speed Computing with Distributed Array Processor," Symposium on High Speed Computer and Algorithm Organization, University of Illinois, 13-15 April 1977.

Gritton, E. C., W. S. King, I. Sutherland, R. S. Gaines, C. Gazley, Jr., C. Grosch, M. Juncosa, and H. Petersen, *Feasibility of a Special-Purpose Computer to Solve the Navier-Stokes Equations*, The Rand Corporation, R-2183-RC, June 1977.

Thurber, K. J., *Large Scale Computer Architecture: Parallel and Associative Processors*, Hayden Publications, Rochelle Park, N.J., 1976.

A Modular Minicomputer Based Navier Stokes Solver

John Steinhoff

Research Department
Grumman Aerospace Corporation
Bethpage, New York 11714

ABSTRACT

A modular computing system for solving time dependent incompressible Navier Stokes equations is described. The basic module consists of a minicomputer, low cost peripheral storage device (disk) and a modest number (8-12) of microcomputer modules. For the problems considered, the device, costing less than $100,000, will have approximately 40% of the computational speed of a large general purpose machine, such as the CDC 7600, at a cost of less than 1%. A specific algorithm is studied, which involves solving advective equations using fourth order finite difference approximations for spatial derivatives, and second order Adams-Bashforth time differencing. At each time step, a Poisson equation is solved for the pressure, using three dimensional Fourier transforms. For this algorithm, the device will be able to handle a fairly large (64^3-128^3) grid. A simple arrangement, where the microcomputers are connected to a single time multiplexed bus, only communicating to the host minicomputer, will be efficient for this problem. By running the machine in a dedicated mode for long periods of time, it will be possible to obtain a large number of solutions. As such, the device should be useful as a research tool.

A scheme is then outlined to assemble a number of these computing modules in parallel to decrease computing time. A simple ring configuration of ~8, using dual-ported disks, will give a speed increase of ~7. This configuration, using 8 minicomputers and 8 pairs of large (300 Mbyte) minicomputer disks would be able to compute 200 time steps for a $(512)^3$ problem in 6-12 days. Another scheme is outlined to further increase speed. This requires a more complex two dimensional planar array of computing modules, but should result in an additional large speedup.

Finally, we outline the advantages and disadvantages of using a number of these systems assembled in a loosely coupled configuration, each independently computing a separate flow, to give a very high throughput. Although we only study one type of problem, it seems likely that the same considerations apply to the larger set of flow problems involved in aerodynamic design and optimization.

It should be emphasized that only available systems, such as minicomputers, disks and microcomputers (or fairly simple modifications) are considered, as opposed to available components (I.C.'s), or available technology (gates). Thus, development costs and risks should be minimal.

1. Introduction

Studies have been done, over the last several years, of dedicated minicomputers as low cost means for solving large scientific problems. Among others, Kottler and McGill[1] used a minicomputer system to solve some problems, including a partial differential equation for pollutant dispersal. Also, in 1973, Steven Orszag[2] compared a minicomputer system to large general purpose machines for solving some time dependent hydrodynamics codes. The conclusions were that minicomputers could be more cost effective than the large machines. In particular, Orszag concluded that the minicomputer was more cost effective than a CDC 7600 but less than a CDC STAR for moderate resolution hydrodynamics codes. For these problems, the data bases are large and the system is essentially a disk based one, where data flows from a large data base on the disk to the computer. When these studies were done, disk speed was not a restriction and the limiting factor was the computational speed of the minicomputer.

In the last several years, improvements in LSI technology have made it possible to assemble reliable computational devices with a much lower cost per computation than minicomputers. These devices, ranging from moderately priced (~$100,000) fast systems with tightly coupled multiple functional units to very inexpensive simple units, can be very cost effective when used with a minicomputer as host. They have had a large impact on the signal processing field[3], in some cases proving to be faster than large machines such as the CDC 7600.

When coupled with the modest advances that have occurred in access time and throughput of disk systems, these improvements now make it possible to build a very effective, moderately priced (~$100,000) special purpose computer for large fluid flow problems.

We will first describe a special purpose computing system of the type that we are assembling in the Research Department. It consists of a minicomputer, moving head disk, and a peripheral processor with multiple functional units. The system is being designed specifically to handle, at low cost, the large data bases and data flow characteristics of a class of finite difference fluid flow algorithms, although there are some Monte Carlo algorithms that we have studied for which it will also be efficient. As an example, a particular time dependent incompressible Navier Stokes problem (Large Eddy Simulation) will then be described. The data flow requirements will be given and the operation of the system will be discussed.

This type of system can be assembled using available minicomputer, disk and microcomputer modules for less than 1% of the cost of a 7600. For the problem considered, however, it can have about 40% of the speed of the large machine (the actual system that we are currently assembling will be somewhat more modest, resulting in about 1/3 of this figure). Although it will be no faster than some large general purpose computers, by dedicating the system to a project for a long period of time, it will be able to economically solve quite large problems. In this mode, it will serve as a fluid flow research tool.

A scheme to increase the solution speed by configuring a number of the above research machines, or computing modules in parallel will then be described. Although the cost effectiveness will only increase by a modest amount (some of the equipment need not be replicated), the higher speed should make the system useful as an engineering tool, where results are needed in a shorter period of time. It should be mentioned that we are only considering algorithms for incompressible flow. Engineering calculations for aerodynamic design will, of course, have other requirements. The idea is mainly to give an example of a large, fast machine using our approach.

Finally, we will outline the advantages and disadvantages of using a number of these systems assembled in a loosely coupled configuration, each independently solving a separate problem, to give a very high throughput.

It should be emphasized that for each of the above configurations only commercially available systems, such as minicomputers, disks and microcomputer modules (or simple modifications) are considered, as opposed to available components (I.C.'s) or available technology (gates). Also, these systems are arranged in simple configurations with simple couplings. Hence, development costs and risks for the individual systems as well as for the assembly into the configurations described will be minimal.

2. Computing Module

The basic computing module consists of a minicomputer, peripheral storage device and peripheral processor (see Fig. 1). We are using a moving head disk for main storage. Eventually, solid state equivalents with much lower access times may be available at a comparable cost (10^{-2}-10^{-3} cents per bit), but probably not in the next several years. Also, we are using a set of independently programmable microcomputers for the peripheral processor. Other high performance processors, such as the FPS-120B "array" processor are also available for this function. These machines have tightly coupled multiple function units controlled by a single instruction stream and are very efficient for long vector operations. Our peripheral system, with independent functional units, will be more flexible and more effective for problems involving a significant number of scalar operations, and in addition will be almost as cost effective for vector operations.

Our initial system will consist of a NOVA 800 minicomputer, a Data General model 6045 10-megabyte (Mb.) moving head disk and a set of 6 microcomputer modules of our own design. For the problem considered, the disk, with a maximum throughput of 160,000 bytes/sec., will be the limiting factor. This system will have about 15% of the speed of a large machine, such as the CDC 7600. The use of a currently available faster disk, such as the Data General 90 Mb. or Cal-Comp Trident 300 Mb. together with about 8-12 modules should result in about three times this throughput. At this point the NOVA data channel will be about 40% saturated. Although faster systems could be assembled, the cost would be higher. At some point it becomes cheaper to replicate the system and operate a number of these systems in parallel.

A basic computing module, with a 300 Mb. disc, a NOVA 3 minicomputer (a new inexpensive version of the NOVA 800) and 12 microcomputer modules will cost approximately $70,000, not including I/O devices such as teletypes and CRT displays.

The basic microcomputer module we are using is based on an Intel 3000 series bipolar microprocessor chip set. Some details are given in Ref. (4). It does arithmetic in 16 bit fixed-point increments with overlapped instruction fetch, data fetch and arithmetic/logic operation, and has a 160 ns. full cycle time. It can also do a 16 x 16 multiply (in hardware) in 2-4 cycles. The control store consists of 512 32 bit words and the high speed data memory of 256 16 bit words, both bipolar 40 ns. random access memory. An additional 8K MOS memory is being added. The module operates independently, except for program load, and sends and receives data from the host minicomputer via the direct memory access (DMA) channel. The cost of additional modules, not including original development, will be about $3000 each. Twelve of these modules will have a total computing potential of approximately 3 (32 bit) multiplies, 16-24 (32 bit) additions, or 36-72 logical operations or (16 bit) additions per microsecond.

To make efficient use of our system, assembly language programming for the time-critical tasks seems to be essential(2), at least at the present time. Since long runs will be made in a dedicated mode, and it is anticipated that a relatively small number of algorithms will be used over a period of years, the relative cost of this software will not be as great as in a conventional, general purpose system where a large number of small jobs are run. The structure of our module with its separate fast data memory and writable control store, is very efficient for this programming mode.

An important feature of the software is that the minicomputer data flow part separates from the microcomputer data flow part. This in turn separates from the microcomputer arithmetic. Thus, each part of the data flow hierarchy can be separately written and checked out.

3.1 Dynamical Equations

We will be solving time dependent incompressible Navier-Stokes equations for turbulent flows, using the Large Eddy Simulation approach. The equations are spatially filtered, resulting in a damping of the high spatial frequency modes, and only the relatively slowly varying "large eddies" are treated explicitly. A sub-grid scale model is used to compute the stresses caused by the small eddies, which are not solved for. The resulting equations, when discretized, are tractable on current very large computers for laboratory sized spatial regions, even for high Reynolds numbers. The approach leads to good predictions for some quantities, such as the filtered energy spectrum.

Our initial approach will be similar to that of the Stanford Group (5,6,7). Most of this work has been done on a CDC 7600 and we should have a good basis for comparison.

The basic equations for the filtered variables are (see Ref. 6, Eq. (2.5))

$$\frac{\partial \bar{u}_i}{\partial t} = -\overline{\bar{u}_j\left(\frac{\partial \bar{u}_i}{\partial x_j} - \frac{\partial \bar{u}_j}{\partial x_i}\right)} - \frac{\partial \bar{P}}{\partial x_i} - \frac{\partial}{\partial x_j}\tau_{ij} + \nu\frac{\partial^2}{\partial x_j^2}\bar{u}_i \qquad (1)$$

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \qquad 1 \leqslant i,j \leqslant 3, \qquad (2)$$

where a bar denotes the filtering operation

$$\bar{u}_i(x,t) = \int G(x-x')\,\bar{u}_i(x',t)\,dx' .$$

The filtered product of two terms, $\overline{\overline{ab}}$, occurring in the left hand side of the above equation, can be computed using a Taylor expansion:

$$\overline{\bar{a}\,\bar{b}} \approx \bar{a}\,\bar{b} + \text{const.}\times\frac{\partial^2}{\partial x_j^2}(\bar{a}\,\bar{b}),$$

where the constant depends on G. The second term is known as the "Leonard term." Eddy viscosity models are commonly used for the subgrid scale term

$$\tau_{ij} = -2\nu_T \overline{S}_{ij}$$

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right),$$

with the Smagorinsky model

$$\nu_T = C_\Delta\left(2\,\overline{S}_{ij}\,\overline{S}_{ij}\right)^{1/2}$$

where C_Δ is the main adjustable constant.

A Poisson equation for the pressure is derived by taking the divergence of (1) and using (2):

$$\frac{\partial^2}{\partial x_j^2}\overline{P} = -\frac{\partial}{\partial x_i}\left\{\overline{\overline{u}_j\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i}\right)} + \frac{\partial}{\partial x_j}\tau_{ij} - \nu\frac{\partial^2}{\partial x_j^2}\overline{u}_i\right\}. \quad (3)$$

The basic approach is to use (1) to compute $\overline{u}_i$ at a new time step, with $\overline{p}$ computed using (3). This method predicts zero divergence for $\overline{u}_i$ at a new time step, if $\overline{u}_i$ has zero divergence at the current time step. The Adams-Bashforth method, which is second-order accurate in Δt is used to integrate $\overline{u}_i$:

$$\overline{u}_i^{(n+1)} = \overline{u}_i^{(n)} + \Delta t\left(\frac{3}{2}H_i^{(n)} - \frac{1}{2}H_i^{(n-1)}\right)$$

where

$$H_i = -\overline{\overline{u}_j\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i}\right)} - \frac{\partial \overline{P}}{\partial x_i} - \frac{\partial}{\partial x_j}\tau_{ij} + \nu\frac{\partial^2}{\partial x_j^2}\overline{u}_i\,.$$

Various finite difference methods have been used to approximate the spatial derivatives. Our example will involve second order equations for the eddy viscosity and Leonard terms as well as their derivatives, and fourth order terms for all other derivatives. Other difference formulae, involving a fixed number of neighboring points, can be computed on our system with schemes similar to the one we will describe.

Our initial problems will involve homogeneous flows. The periodic boundary conditions used in these flows are very simple to handle and represent a first study for our system. Other boundary conditions, such as no-slip ones, result in additional problems from the numerical analysis$^{(8)}$, as well as programming point of view. Especially when higher order methods are used, different operations must be done at grid points near boundaries. Unlike array processors, our system could do a number of these different tasks in parallel, but the programming would be more difficult. For the periodic boundary conditions used, the Poisson equation for the pressure can be solved directly using three dimensional Fourier transforms.

3.2 Implementation

The computations required at each time step can be divided into two parts: a semi-local phase where the finite difference approximations to the partial derivatives are computed and the velocities are updated; and a global phase where Poisson's equation is solved by a direct method (three dimensional Fourier transforms). The data flow requirements for the two phases are quite different.

3.2.1 Semi-local Phase

In the semi-local phase, the data, which consist of two three-vector arrays, w_i and z_i (related to the velocities at two time steps) and a scalar array (pressure) is read from the disk in a sequence of "computational steps." At each of these steps, derivatives of the pressure are computed and the velocities ($\bar{u}_i$) are updated for a set of points in the array. Then, new values of the subgrid scale terms are computed, and new sets of w_i and z_i and a new source term (ρ) for Poisson's equation (3) are computed and written back onto disk. The semilocal phase is characterized by the fact that the computations required for a given point in the array depend on data only from neighboring points, at most several grid points away. The data for one step is depicted in Fig. 2. In this example, blocks of new values corresponding to 4 units in the χ_1 direction, 64 units in the χ_2 direction (the full array width) and 8 units in the χ_3 direction are computed at each step. The variables required to compute these values are just those needed to compute the various partial derivatives at those points. It turns out that if we save some values from previous steps, the pressure (P) is needed in $(\delta\chi_1, \delta\chi_2, \delta\chi_3)$ = (4 x 64 x 16) blocks, z_i in (4 x 64 x 12) blocks and w_i in (4 x 64 x 8) blocks at each step. Taking account of extra disk accesses at boundary points, it turns out that ~10.7 x $(64)^3$ values have to be read from disk and ~7.3 x $(64)^3$ values computed and written back onto disk during the semilocal phase. (These figures are for a system with approximately 96K - 16 bit words of total random access memory (12 microcomputers with 8K each)).

At each step, as shown in Fig. 2, each microcomputer operates on values corresponding to a distinct range of χ_2 values. Thus each microcomputer module operates on a different χ_2 segment of the entire array. (It should be emphasized that the modules are independently programmable, and the actual points computed by each module can vary, as well as the actual computations performed. This feature should be particularly important for complex boundary computations).

We estimate that about 30-40% of the disk transfer time will be spent locating the data (latency) and 60-70% of the time transferring at contiguous-block rate. For the semilocal phase, the disk transfer time comes to $T_s^D \approx 30 \times (64)^3 \tau_w$, where τ_w is the time required to transfer a word at contiguous-block rate. Assuming 32 bit words, we have $\tau_w \approx 7 \mu$sec. for many currently available disks, giving $T_s^D \approx 55$ sec. We estimate that approximately 6 - 12 of our microcomputer modules will be able to do the computations and match this rate.

3.2.2 Global Phase

In the global phase, the ρ array is Fourier transformed. This transformation diagonalizes the Poisson equation. The resulting values are divided by a constant which depends on wave number and an inverse transform is performed on the resulting array, giving the pressure. The characteristic feature of the global phase is that, for each array point computed, data values are required from all other array points, and we cannot sequence through the data as in the previous phase. In particular, for the three dimensional Fourier transform, sets of one dimensional transforms are required in three different directions (χ_1, χ_2 and χ_3), and the data must be accessed in three different directions. It turns out that we can do the first transform (χ_2) on ρ in the semilocal phase before writing ρ onto the disk, and the last inverse transform (χ_2) on ρ after reading from the disk, so that in the global phase it is only necessary to do $\chi_1 - \chi_3$ transforms. These transforms require that data in χ_1-χ_3 planes be transferred to and from the disk. We effectively have a transpose problem, which can be solved fairly efficiently on a disk system with several addressable surfaces. For a disk with four surfaces and a system with total random access memory of 64K-16 bit words, we can read the $\chi_1 - \chi_3$ planes at the rate of about $4\tau_w \approx 28$ μsec per word, giving an effective latency of 75%. The resulting disk time; $T_G^D \approx 8 \times (64)^3 \tau_w$. For $\tau_w \approx 7 \mu$sec., we have $T_G^D \approx 15$ sec. Again, the computational speed of 6 - 12 microcomputers will approximately match this rate (in this phase, each module will do a separate one dimensional transform).

Assuming buffered operation for both phases, the total time for a time step is approximately 70 sec. (The total time for our initial prototype system should be about 200 sec.). For a $(128)^3$ problem, we have about 10 minutes per time step, or $\sim$32 hours for a flow requiring 200 time steps.

The host data channel will be about 40% saturated in the above calculations. A straightforward way to double the performance of the system would be to use a minicomputer with a slightly higher data channel rate, two 300 Mb. disks, and either 12-24 16 bit microcomputer modules, or 6 - 12 expanded 32 bit modules. Such a system would be able to solve $(256)^3$ problems, with 200 time steps taking $\sim$ 5 days; a reasonable time for a moderately priced, dedicated system.

4. Multi-Module Systems

Defining a module to be a minicomputer, set of microcomputers (16 or 32 bits) and one (or two) disks, we will describe two ways to increase solution speed by assembling a parallel system.

4.1 One Dimensional Array

The first is quite simple: we just have to dual-port each disk and form a ring configuration (see Fig. 3). For the $(64)^3$ problem, 8 sweeps (in the $\mathcal{X}_1$ direction) were required for a single module for the semilocal part of each time step. We could then use 8 modules-one for each sweep. With some minor exceptions, each module could read the same variables (during a sweep) as if it were doing the entire problem. These variables would be available on the two disks (or four disks) connected to each minicomputer. If the disk transfers were coordinated so that each left-hand disk were accessed at the same time by each minicomputer and then each right-hand disk, there would be no disk conflicts. This synchronization would require the testing of a flag only after the transfer of several hundred variables, and could be done using straightforward software techniques and standard interprocessor busses (IPB's). An advantage of this scheme is that one module by itself could use essentially the same program (for the semilocal phase) and solve the entire problem. Also, by changing K (the $(\mathcal{X}_3)$ width of the sweep of each module), and the number of sweeps each module must take, we could conveniently use any number of modules up to $\sim$8. This should simplify development of the full system and decrease time lost due to failures. The computational speed should increase roughly linearly (in the semilocal phase) with the number of modules.

We now describe a scheme to do the global part - the solution of Poisson's equation. For the single module, approximately 22% of the disk transfer time was associated with the Poisson solution, with only about 14% of the data involved, since we had to read smaller non-contiguous blocks at about 25% efficiency. With 8 modules, we could segment the data (ρ array) in the χ_2 direction into 8 groups (after doing the χ_2- one dimensional transforms), and rearrange the groups among the modules. Each module then would have to transfer 7 groups to the other 7 modules. By transferring around the ring-first in one direction then in the other, the rearrangement can be done in 7 steps. Each piece of data moves an average of 2 modules during the operation, and we finally have 8 entire (χ_1-χ_3) planes of data in each module (corresponding to different χ_2 values). The flow of the data (in one direction) is depicted in Fig. 4. Each solid line represents a transfer at the first step. The number of groups transferred by each module is 4, 3, 2 and 1 for the respective right-steps, and 3, 2 and 1 for the left-steps. (Most of these transfers can be overlapped with the semi-local phase). The first read for each value is similar to the read for the single module, except that larger blocks are read, with smaller latency ($\sim$50%). It appears as though the remaining transfers can be done with $\sim$30% latency. The entire transfer, which requires $2 \times (64)^3$ reads and $2 \times (64)^3$ writes, should take $\sim .8(64)^3\ \tau_w$, compared to $\sim 4(64)^3\ \tau_w$ required for the corresponding transfer in the single module. To complete the Poisson solution, we then have to do a two-dimensional ($\chi_1 - \chi_3$) transform in each module, a division, an inverse transform, and then an intermodule transfer again. The time required for the total solution (semilocal + global) is then $\sim 5.3\ (64)^3 \tau_w$, compared to $\sim 37\ (64)^3 \tau_w$ for one module, giving a speedup of ~ 7.

For a $(256)^3$ array, we have $\sim$ 640 and 320 seconds per time step, respectively, for single and dual disk systems. For 200 time steps, this gives $\sim$ 40 and 20 hours. Dual 300 Mbyte disks and 6 - 12 32 bit microcomputers would allow a $(512)^3$ solution (200-400 time steps) in 6-12 days.

There are some advantages to using the system that we have described. First, since it is a one dimensional configuration with simple connections, adding new modules or disconnecting failed ones should be straightforward, both in software and hardware. This also applies to each module, which is a linear configuration of microcomputers. Second, although we have 64-96 microcomputers operating in parallel, only groups of 8-12 microcomputers need be coordinated, and only a group of 8 modules need be coordinated. Third, each module can have its own operating system and do an entire problem by itself, with any number of microcomputers.

4.2 Two Dimensional Array

An obvious way to extend our system would be to form a two dimensional (8 x 8) array of modules. This system would have 512-768 microcomputers. The physical module arrangement would be similar to the ILLIAC IV, but the operation would be completely different.

Except for very large problems, a solid state storage device would most likely be sufficient for each module. Each minicomputer would then be connected to four of these storage devices, and would operate independently, only exchanging data and control primitives with four neighboring modules. Although we would lose most of the advantages of the simpler system, we would get a large speedup.

The $(64)^3$ problem would be segmented in 64 squares in the χ_1-χ_3 plane for the modules, and into 8-12 segments in the χ_2 direction for the microcomputers in each module. The semilocal operation would be a straightforward extension of the linear system's operation. There would be no access conflicts if all "North" memories were accessed at the same time, all "East" ones, etc. As in the last section, this synchronization would be simple in software.

The Poisson solution would also be a straightforward extension: For the data transfer there would be (say) a horizontal phase, exactly the same as the previous section, followed by a vertical phase - again the same. Although there would be twice as many data transfers as before (in the global phase), if a solid state memory were used with a small ($\lesssim$ 1 ms.) access time and 1-2 bytes/sec. data rate, the faster transfer times would more than compensate for this additional time. Hence, we should have roughly a factor of 64 speedup compared to one module.

Conclusion

We have described a series of machines, assembled from available high level components, that can have between 1/2 and 64 times the speed of a CDC 7600 for the incompressible Navier Stokes problems considered. Extensive use is made of buffering and software synchronization so that the inter-component connections are as simple as possible. Also, the systems are highly modular, and the more powerful machines consist of sets of simpler machines connected together. Thus, a program to develop machines along these lines and gradually increase complexity and computational speed should be fairly straightforward. As new hardware becomes available, during development, it could be incorporated at the subsystem level without major overall system changes.

It seems difficult to extend this simple, low risk design approach to much higher speed machines, without using more exotic hardware. However, if more performance is required and the above features prove to be important, an additional order-of-magnitude increase in throughput could be obtained by assembling a number of these machines in a very loosely coupled configuration. Each machine would then independently do an entire flow calculation, communicating to central archival store (if at all) only at the beginning and end of the problem.

This configuration would obviously have very little down time. Also, new hardware and new computational algorithms are continually being developed. Individual machines could be taken down and modified to incorporate these new developments with little impact on the overall system. In fact, certain specialized algorithms might have very particular data flow requirements for a certain class of problems. Some of the machines could then be made "specialized" and operate differently from the others.

Of course, there are disadvantages to the above approach. An aerodynamicist could most likely not use more than several machines concurrently, and would be able to get the most information out of a series of runs if they were done faster and sequentially (on a single machine). Thus, if 10 separate machines were used, each doing separate flow calculations, the attention of several aerodynamicists would be required, each working on a different overall flow problem. There is thus a trade-off between the need for additional aerodynamicists and the above benefits. Also, if automated optimization procedures, such as conjugate gradient techniques, are used for varying parameters and optimizing aircraft configurations, there might be very little decrease in efficiency involved in computing sets of solutions concurrently: There are parallel optimization algorithms (9) with rates of convergence that are comparable to the sequential ones.

The main reason that it is cost effective to build machines to concurrently do separate flow field calculations, for the problems considered, is that the large data bases reside on very cheap (per word) moving head disks. If fast random access memory were used for the data base, it would most likely be more cost effective to develop a single highly parallel machine, and avoid replicating the memory. Large random access memory, however, is not required for the flow calculations considered.

Although we have only considered a very specialized algorithm, it seems likely that the same considerations apply to larger classes of problems. This extrapolation should, of course, be tested by detailed studies of other algorithms.

References

1. C. Kottler, R. McGill, "The Feasibility of Using Minicomputers for Reducing Large Problem Solving Costs," Instruments and Control Systems, Vol. 46, p. 57 (1973).

2. S. A. Orszag, "Minicomputers vs. Supercomputers: A Study in Cost Effectiveness for Large Numerical Simulation Programs," Flow Research Note No. 38, Flow Research Inc., (1973).

3. J. Allen, "Computer Architecture for Signal Processing," IEEE Proceedings, p. 624, (1975).

4. R. McGill, J. Steinhoff, "A Multimicroprocessor Approach to Numerical Analysis: An Application to Gaming Problems," Proceedings Symposium on Computer Architecture, p. 46 (1975).

5. D. Kwak, W. C. Reynolds, J. H. Ferziger, "Three-Dimensional Time-Dependent Computation of Turbulent Flow," Report No. TF-5, Department of Mechanical Engineering, Stanford University, (1975). Also, S. Shaanan, J. H. Ferziger, W. C. Reynolds, "Numerical Simulation of Turbulence in the Presence of Shear," Report No. TF-6, Department of Mechanical Engineering, Stanford University (1975).

6. N. N. Mansour, P. Moin, W. C. Reynolds, J. H. Ferziger, "Improved Methods for Large-Eddy Simulations of Turbulence," Proceedings Symposium on Turbulent Shear Flows, University Park, Pa., p. 14.21, April 1977.

7. R. A. Clark, J. H. Ferziger, W. C. Reynolds, "Evaluation of Subgrid-Scale Turbulence Models Using a Fully Simulated Turbulent Flow," Report No. TF-9, Department of Mechanical Engineering, Stanford University (1977).

8. P. J. Roache, "Computational Fluid Dynamics," Hermosa Publishers, Albuquerque, N.M., 1972.

9. D. Chazan, W. L. Miranker, "A Nongradient and Parallel Algorithm for Unconstrained Minimization," SIAM J. Control, 8 p. 207 (1970).

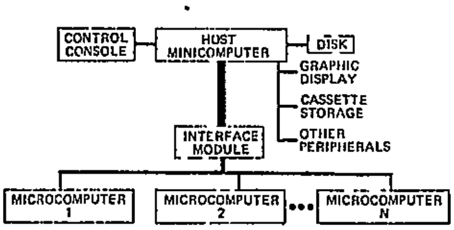
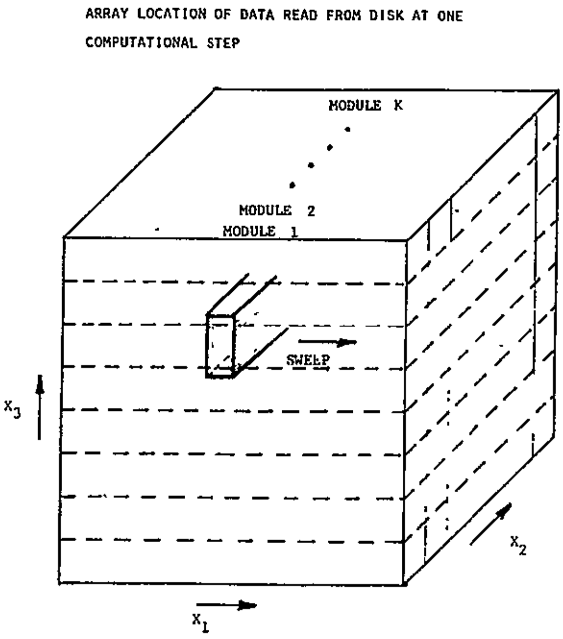


Fig. 1 Computing System Organization



ARRAY LOCATION OF DATA READ FROM DISK AT ONE COMPUTATIONAL STEP
MODULE K
MODULE 2
MODULE 1
SWEEP
x_3
x_2
x_1

Fig. 2 Data Flow -- Semilocal Phase

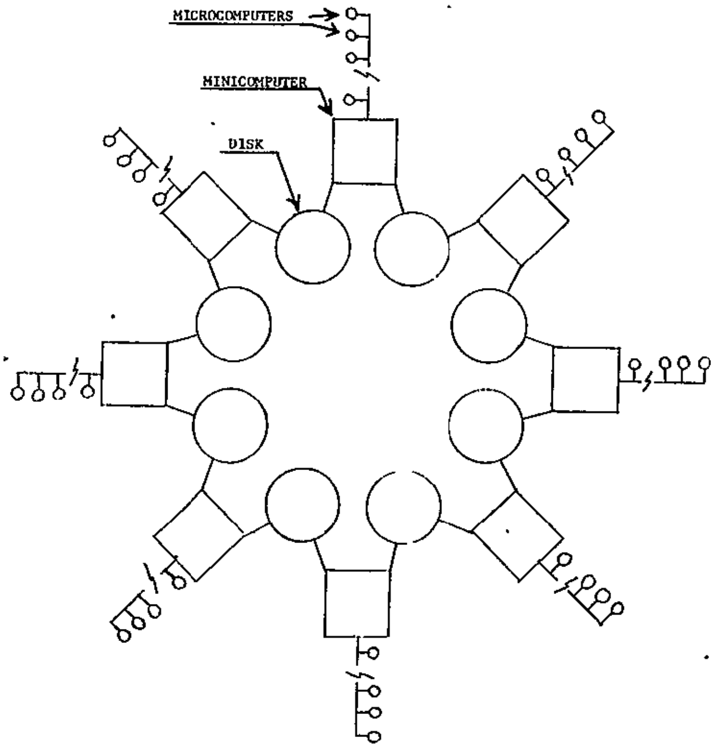


Fig. 3 Multi-Module System

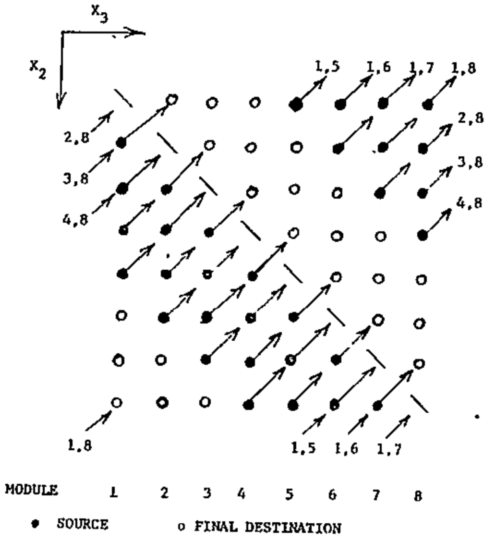


Fig. 4 Transpose -- First Step

COMPUTATIONAL ADVANCES IN FLUID DYNAMICS

T. D. Taylor
The Aerospace Corporation
Los Angeles, California

ABSTRACT

This presentation outlines advances that are possible in special purpose digital computers for solving fluid dynamics problems. The principal points of interest are 1) an office-size digital computer with capabilities exceeding a CDC 7600 can be constructed for less than 300K and 2) spectral methods offer a factor ten improvement over finite difference methods. Integration of these facts can produce two orders of magnitude reduction in cost of fluid dynamic computations and permit solution of 3-D problems which were not previously feasible.

INTRODUCTION

In recent years, significant advances have been made in numerical solution of fluid mechanics problems using digital computers. Many of these advances are documented in References 1 - 5. However, 3-D high Reynolds number viscous flow problems have resisted solution by standard numerical techniques on high speed computers. In particular, the prediction of the onset of transition to turbulence remains as one of the major unsolved applied problems of fluid mechanics. Two specific areas that now need such capability are (1) the design of low drag hydrodynamic vehicles and (2) the design of nose tips for reentry vehicles.

The difficulty associated with solving high Reynolds number flow problems on the computer is twofold. The first is the numerical technology which until recently has focused on finite difference methods. (These techniques yield limited resolution due to grid point restrictions induced by the high Reynolds numbers.) The second is the cost per computation on computers such as the CDC7600. This situation is now changing since new developments in both numerical methods and computer hardware are becoming available. In particular, "spectral methods" for solving high Reynolds number flow problems which are significantly faster than finite difference approaches are now available. In addition, computer processors with the speed of a CDC7600 are becoming available which offer hardware cost reduction in excess of a factor of 10*.

As a result, one can now visualize the possibility of large cost reductions (greater than a factor of 10) in the solution of fluid mechanics problems In addition, the low cost of hardware will make it possible to change the basic philosophy in computational fluid dynamics. For example, one can now consider a special purpose numerical processor which will operate at CDC7600 speeds or greater for a hardware procurement cost of less than 250K. Such a processor can be set up to solve fluid mechanics problems on a dedicated basis and can be run for long periods of time without concern for typical main frame hourly computer charges. The result is that one can consider extensive fluid mechanics computations at a controlled fixed cost which were not previously feasible. This change in computing philosophy can bring us closer to the realization of a computer simulation of operating systems and have a large impact on the solution of other engineering problems as well. It

* Floating Point Systems - AP 120B is an example

will make possible the use of large scale computer codes for design and process simulation that was not previously practical.

As this discussion indicates, the basic technology elements now exist for making significant advances in computational fluid mechanics. This paper outlines a plan for assembling these elements to simulate high Reynolds number viscous flows.

TECHNICAL DISCUSSION

The development of low cost fluid mechanics problem solutions requires both advanced numerical methods for solution of the Navier-Stokes equations at high Reynolds numbers and reduced computer cost per computational hour. The elements of technology which can be assembled to produce an advanced hydrodynamic numerical simulator will now be outlined.

Computation Methods

An extensive list of finite difference methods, which have been employed to attempt to solve the Navier-Stokes equations, can be compiled. For reference, the AGARDographs of Taylor[1] and Peyret and Viviand[2], along with the book of Roache[3], provide a rather complete discussion of the various approaches. As a result, an attempt will not be made here to discuss the details of the methods. Rather, the source of the difficulty in solving high Reynolds number, multidimensional problems will be illustrated. For this demonstration, it is sufficient to consider a model equation, the three-dimensional unsteady Burgers equation. In dimensionless form, this equation is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{R}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 y}{\partial z^2}\right) \qquad (1)$$

where R is the Reynolds number. The interest here is in solutions for very large values of the parameter R. (Flat plate boundary layer transition occurs at values of R in excess of 10^6.) This equation can be reduced to finite difference form by Taylor series expansions of the derivatives. When this is accomplished, one then must address questions of accuracy and stability of finite difference solutions. When all the finite difference terms are of order one, particularly

$$\frac{1}{R}\left(\frac{\Delta^2 u}{\Delta x^2} + \frac{\Delta^2 u}{\Delta y^2} + \frac{\Delta^2 u}{\Delta z^2}\right)$$

the accepted stable finite difference approximations usually do not encounter difficulty with accuracy or stability. As the Reynolds number increases, in order to keep the finite difference scheme stable it is necessary to lower the grid spacing so that

$$\frac{1}{R}\left(\frac{\Delta^2 u}{\Delta x^2} + \frac{\Delta^2 u}{\Delta y^2} + \frac{\Delta^2 u}{\Delta z^2}\right)$$

remains measurable in magnitude. Experience has shown that when $(R \bullet \Delta x)$, $(R \bullet \Delta y)$ or $(R \bullet \Delta z)$ exceeds about four[6], then it is necessary to have some additional artificial viscous term to stabilize the finite difference solution. This artificial dissipation tends to degrade the accuracy of the solution. For transition studies where accurate solutions are required at $R > 10^6$, it becomes evident that these restrictions introduce prohibitive step size and grid point distribution requirements. This should be expected since one then is encountering the classical singular perturbation problem associated with small second order terms in a differential equation. Experience[7] dictates that for methods without significant artificial dissipation the finite difference approximations can be implemented to obtain reasonably accurate results up to about $R = 10^4$ for flows with large gradients. For Reynolds numbers $>10^4$, artificial viscous terms appear to be necessary for such flows. For the investigation of transition which occurs typically for $R > 10^6$ and involves small disturbances, calculations with the minimum artificial dissipation are required for detailed understanding of the phenomenon. As a result, one must look for more efficient computational methods.

In recent studies, both finite differences [7, 8] and spectral methods have been investigated[9, 10] for solution of viscous flow problems. From the results, it can be concluded that a factor of 10 in computation speed can be gained over finite differences by applying spectral methods to solve viscous flow problems. This conclusion agrees qualitatively with a comparison of second and fourth order finite difference methods and a spectral method made by Orszag and Israeli.[11, 12] They state that in order to accurately (five percent accuracy) resolve a sinusoid, 20 finite difference points per wave length are required when using a second order method, 10 points per wave length with a fourth order method, and π modes per wave length with a spectral method. While these differences are not too impressive for one-dimensional problems, for the two- and three-dimensional problems of interest here the savings in computer storage can exceed two orders of magnitude (for example, $20^3 = 8000$, $10^3 = 1000$, $\pi^3 = 31$). This saving is directly reflected into computation time.

As a result of arguments such as those presented in the preceding paragraph, spectral methods should be emphasized for solving viscous flow problems. To date, the primary emphasis on spectral method application has been in solving the two-dimensional unsteady Navier-Stokes equations for incompressible flow. More effort is needed, however, in evaluating the usefulness of the method in compressible flow.

Computer Technology

The progress of large scale computer main frame development for solution of scientific problems is well known and no attempt will be made to elaborate on it in this discussion. Instead, the discussion will focus only on the most recent technology developments. The most recent large scale data processors which have been considered for scientific problem solutions are the CRAY 1, CYBER76, ASC, STAR 100 and ILLIAC IV. Each of these processors offers various features such as pipelining, parallel processing, microsecond clock times, etc. They all, however, have the problem of costing upward of $5-million and, hence, require major computational center investments. As a consequence, they basically limit scientific computation studies due to system cost per hour unless the computing task is of immediate value to a government project which can absorb the cost. Unfortunately, most scientific advances require development before they can be justified on an applied project. As a consequence, there is a demand for an alternate approach to scientific computational capability which may not necessarily develop along the "bigger is better" line of logic[*]. The basic building blocks for developing this new approach are now available and one of the proposed tasks is to utilize this new technology to develop a low cost fluid mechanics problem solver. It is important to note, however, that even though this proposal focuses on fluid mechanics, the basic philosophy can be applied to numerous other technology areas.

The basic elements of a new inexpensive computer are the powerful processor and memory chips which have revolutionized the desk calculator business and now are beginning to impact large scale computer designs. For example, Cyre, et al, at the University of Wisconsin, have proposed in a recent paper[13] to develop a special purpose finite difference or finite element computer using micro processors (with memory) at each grid point to compute the solution[**]. The processors would be coupled to six nearest neighbors for 3-D computations. The nearest neighbor concept becomes inefficient, however, if one needs to introduce implicit or higher order methods. This approach is optimum for an explicit three-point difference or finite element scheme (normally second order accurate) for solving problems. As pointed out earlier, high

[*] Initial steps in this direction have been indicated in publications by S. Orszag[19] and D. Auld and G. Bird[20].

[**] It is understood that Rand Corporation has made a similar proposal.

Reynolds number calculations by explicit methods imply extensive grid point distributions, and one is faced with construction of a 3-D grid point computer which will handle 10^5 and 10^6 grid points. Each unit will have to contain all necessary function subroutines for the algorithm being used, as well as storage, arithmetic processors, program control and error checks. As a result, the cost of a grid point unit will certainly exceed one or two dollars. When this cost is added to the cost of a control system and the software (ignoring hardware design and development cost), the overall system begins to look expensive for a special purpose computer.

A number of questions immediately arise if one considers expanding the concept to handle more general algorithms efficiently. These are:

1. How does one develop grid point connection architecture which will efficiently permit problem solutions by implicit and higher order algorithms such as spectral methods?
2. What development is required to have special grid point units per algorithm which encompass the necessary function subroutines for a computation?
3. How does one incorporate boundary conditions into a grid point computer without having them control the computing time?

From the discussion and questions, it becomes evident that possibly another approach which uses the new integrated circuit technology for high speed computing may have more to offer. This is emphasized even more when one considers that this encompasses basic computer hardware development.

An alternative to the micro processor per grid point approach which is consistent with current computer development is the use of high speed array processors as computer code subroutines. This approach permits overall flexibility to design a computer and code for a specific problem. The concept is to employ a mini or main frame as the host main program control and employ array processors to perform the subroutine calculation tasks. (Note that the subroutine can be the entire calculation of the program if desirable.) The array processor itself is coded to perform whatever subroutine calculation that one chooses.

The basic element of this new inexpensive computer is a low cost ~30K array processor that has become possible because of new large scale intgrated circuit technology. Such a unit is produced by Floating Point Systems

(FPS-AP-120B) and is readily available. The unit has been benchmarked by Professor R. Bucy at USC against the CDC7600 and the STAR 100[14,15]. Dr. Bucy found the unit to be roughly 2.5 times the speed of the 7600 and only slightly slower than the STAR[16]. In a recent paper on program and software requirements for high speed computers, Gary[17] compared the CRAY, CYBER 175 and FPS-AP-120B and indicated that, conservatively, the FPS box could be an order of magnitude better than the CRAY in (flops/dollar) for scalar operations and a factor of two better in vector operations. These two studies, along with the author's comparison given in Table 1, give definite substance to the postulate that this new technology should be able to reduce scientific computation cost by a factor of 10 and still maintain reliability.

The proposed concept is particularly appealing when one examines the advantages of cost, flexibility and development requirements. An installation employing these new processors can vary in cost from 60K to 300K depending on the configuration and needs. A possible configuration is shown in Figure 1. The systems needs a host computer which can be an existing main frame or an off-the-shelf minicomputer. For I/O, it can use the host system or be combined with a tape or disc. For these types of costs, one can consider a dedicated computational unit which can be run for long times on one problem at minimum cost. In addition, with a proper arrangement, this type of unit can remove large computation tasks from a main frame so that it can operate more optimally in job scheduling and time sharing mode.

The short term drawback to these new computational units is the need for a dedicated array processor programmer, but the experiences of Dr. Bucy at USC indicate that this is not a serious problem. In the long term, there will be a need for some compiler development. Such development is a computer systems task and should not be attempted by the applied user.

Other advantages of the concept are:

(1) Existing commercially tested computer elements and software can be utilized without significant development.

(2) Concept can be expanded from a single processor to processors operating in parallel as needed.

(3) The array processor is programmable and can be coded to solve all types of problems by either finite differences, finite element or spectral methods.

TABLE 1

Comparative Computer Statistics (Average)

	MIPS	Relative Speed	MFLOPS	MTTF	Relative Hardware Cost
FPS	60	2.5	12	3000 hrs	0.05
CDC7600	30	1	12	days	1.00
CRAY*	?	2-3	60 (25)**	7 hrs	0.60

MIPS - million instructions per second

MFLOPS - million floating point operations per second

MTTF - mean time to failure

* taken from LASL CRAY-1 evaluation

** ( ) currently obtained speeds

(4) Speeds for a single array processing unit can exceed CDC7600 and for an ideal situation for N units can be N times the speed of the CDC7600. (For most cases, however, this speed will be less because of the nature of the computation algorithm.)

(5) FPS processor speeds can be increased a factor of two without significant cost change and word lengths can also be extended without major hardware difficulty[18].

(6) This type of computer will permit engineering groups of all types to run complex codes for design at low cost.

CONCLUSIONS

The previous technical discussion outlined a significant advance that can now be made with regard to fluid mechanics simulation by a combination of both computational and hardware advances.

It is this author's view that plans for development of an advanced fluid dynamics computational facility should recognize the trends outlined in this discussion. In this context, it is proposed that the development of a super computer occur in modular concept so that as high speed arithmetic and storage units evolve they can be made available to industry to form small dedicated computers for research and engineering application. By introducing this planning, NASA can have a major impact on technology as they develop a large fluid dynamic simulator.

REFERENCES

1. Taylor, T. D., "Numerical Methods for Predicting Subsonic, Transonic and Supersonic Flow," AGARDograph No. 187, 1973.

2. Peyret, R. and H. Viviand, "Computation of Viscous Compressible Flows Based on the Navier-Stokes Equations," AGARDograph No. 212, 1974.

3. Roache, P. J. Computational Fluid Dynamics, Hermosa Publishers, Albuquerque, NM, 1972.

4. Holt, M., Numerical Methods in Fluid Mechanics, Springer-Verlag, New York, NY, 1977.

5. "Proceedings of International Conferences on Numerical Methods in Fluid Dynamics, 1-6," Lecture Notes in Physics, Springer-Verlag, New York, NY.

6. Taylor, T. D., E. Ndefo and B. S. Masson, "A Study of Numerical Methods for Solving Viscous and Inviscid Flow Problems," J. Comp. Phys., 9, 1972, pp. 99-119.

7. Widhopf, G. F. and K. J. Victoria, "On the Solution of the Unsteady Navier-Stokes Equations Including Multi-Component Finite Rate Chemistry," Computers and Fluids, No. 2, 1973, pp. 159-184.

8. Taylor, T. D., "An Evaluation of Cell Type Finite Difference Methods for Solving Viscous Flow Problems," Computers and Fluids, Vol. 1, 1973.

9. Murdock, J. W., "A Numerical Study of Nonlinear Effects on Boundary Layer Stability," AIAA 15th Aerospace Sciences Meeting, Paper No. 77-127, to be published in AIAA Journal.

10. Murdock, J. W. and T. D. Taylor, "Numerical Investigation of Nonlinear Wave Interaction in a Two-Dimensional Boundary Layer," Proc. of AGARD Symposium on Laminar Turbulent Transition, May 1977.

11. Orszag, S. A. and Israeli, "Numerical Simulation of Viscous Incompressible Flows," Annual Review of Fluid Mechanics, Vol. 6, 1974, pp. 281-318.

12. Orszag, S. A., "Turbulence and Transition: A Progress Report," 5th Int. Conf. on Numerical Methods in Fluid Mechanics, Springer-Verlag, New York, NY, 1976.

13. Cyre, W. R., C. J. Davis, A. A. Frank, L. Jedynak, M. J. Redmond and V. C. Rideouf, "A Parallel Array Computer for Solution of Field Problems," Proc. 1977 Array Numerical Analysis and Computer Conf. on Numerical Solution of Partial Differential Equations, Madison, WI, 1977.

14. Bucy, R. S., K. D. Senne, H. M. Youssef, "Pipeline Parallel and Serial Realization of Phase Demodulators," ICASE Report No. 76-31, NASA Langley Research Center, Hampton, VA, Nov. 1976.

15. Bucy, R. S. and K. D. Senne, "Non-Linear Filtering Algorithms for Parallel and Pipeline Machines," Proc. Conf. on Parallel Math and Computations, North Holland Press, Munich, April 1977.

16. Bucy, R. S., private communication, May 1977.

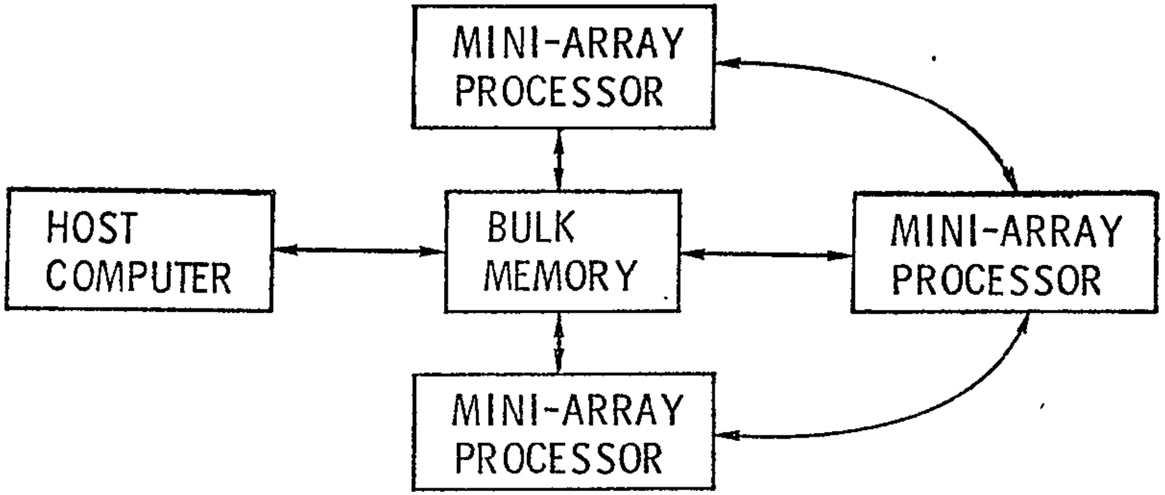
17. Gary, J. M., "Analysis of Applications Programs and Software Requirements for High Speed Computers," to be published.

18. O'Leary, G., private communication, April 1977.

19. Orszag, S., "Minicomputers vs. Supercomputers: A Study in Cost Effectiveness for Large Numerical Simulation Programs," Flow Research Report No. 38, Oct. 1973.

20. Auld, D. J. and G. A. Bird, "Monte Carlo Simulation of Regular and Mach Reflections," AIAA Journal, Vol. 15, No. 5, 1977, pp. 638-641.

Example from Existing Hardware



MINI-ARRAY PROCESSOR

HOST COMPUTER

BULK MEMORY

MINI-ARRAY PROCESSOR

MINI-ARRAY PROCESSOR

- HOST-COMPUTER SERVES AS I/O
- MINI-ARRAY COMPUTERS PERFORM CALCULATIONS AT SPEEDS OF CDC 7600 AND EACH HAVE 10K TO 100K HIGH SPEED MEMORY
- BULK MEMORY CAN BE UP TO 10^6 WORDS WITH ACCESS SPEEDS 600 MONO SECONDS

FIGURE 1

SESSION 12

Panel on SUPERCOMPUTER DEVELOPMENT EXPERIENCE

S. Fernbach, Chairman

omit

PANEL ON SUPERCOMPUTER DEVELOPMENT EXPERIENCE

INTRODUCTION

S. Fernbach, Panel Chairman
Lawrence Livermore Laboratory
Livermore, California

This panel discussion will be devoted to the experiences gained from Supercomputer Development of the recent past. Problems involved in management of computer projects, in development-type contracts, in special purpose computer systems, and special purpose systems which were not intended as such are some of the topics to be covered.

The initial user of supercomputers also have experienced problems in the contractual, acquisition, and implementation areas. Advanced computers may push the state of the art in either component development or architectural design, or both. When both are involved, failure of realization of one can impact the realization of the other.

In soliciting for a specification many prospective vendors become interested. Some may have hardware in fact, some in mind, others just gleams in their eyes. How does one evaluate paper machines? Price alone is of course meaningless; the contractor is willing to risk losses to get a development underway. Performance is speculative and often not met. It is difficult to specify a machine that will behave completely as intended. Today more thorough simulation is possible so that risk of failure is somewhat less than it was in the past. On the other hand, it may not be possible to get expected program performance even though the hardware is as specified. Simulation is again possible but costly and time-consuming.

Initial hardware performance has often left much to be desired; check-out time always seems to take much longer than expected. If mean time between failure is short, users are very, very unhappy. Even if the hardware performs well, usually the software is not well enough developed to operate satisfactorily. It is often difficult to ascertain whether to ascribe a failure to hardware or software.

On the whole, check-out time for a new computer can take years - a minimum of at least one. Preparing the operating system or checking it out is quite a chore. Having the appropriate application problems coded and ready to go at time of installation is another difficult job. Each software effort takes time to implement and check out. In instances where checkout of a system was accomplished with significant large jobs, it was later found that other jobs would not run until both hardware and software modifications were made.

Even today the construction, checkout and full implementation of a new supercomputer is an art rather than the science it should be.

OMIT

PEPE DEVELOPMENT EXPERIENCE

John A. Cornell
System Development Corporation
Huntsville, Alabama

PEPE (Parallel Element Processing Ensemble), currently the world's most powerful computer for a broad class of problems, is a classic example of a supercomputer system successfully designed, built, and operated to meet a general set of requirements that were not well understood at the start of the project. It was developed for research on, and ultimate use in, real-time ballistic missile defense systems. Its mission and user community are therefore considerably different from those of the other computers discussed in this workshop, but the experiences obtained and lessons learned during its development and operation are relevant to the development and use of any supercomputer.

PEPE can be regarded roughly as a large master computer, called a host, controlling many smaller slave processors, called elements. In the present design, the host is a CDC 7600, and there are 288 elements. Each element contains three processors sharing a common data memory. One of these processors, the correlation unit, is used for inputting data and has an instruction repertoire especially suited for the rapid correlation of new data with data already on file. The second processor, the arithmetic unit, has a repertoire similar to that encountered in conventional high-power general-purpose machines; i.e., fixed and floating point arithmetic operations, load and store, and logical operations. The third processor, the associative output unit, is used for finding and outputting data and is especially designed to perform complex, multidimensional file searches rapidly and efficiently. Each of the three processors is driven by its own control unit, which simultaneously drives all of the corresponding processors in the ensemble of elements. The three control units are also capable of executing their own sequential programs. They are combined into a control console, which drives the ensemble of elements in parallel and interfaces the ensemble with the host. The complete PEPE host system, then, is a multiprocessor employing seven processors in all (host, three sequential processors, and three parallel processors). All seven processors are capable of simultaneous, overlapped operation.

Support software for the PEPE includes the compilers and assemblers for the seven PEPE processors and a monitor system for binding programs into executable load modules. The entire machine can be programmed in a single language called PFOR, which is a superset of FORTRAN. PEPE software also includes an instruction-level simulator for PEPE, a general-purpose real-time operating system, and a general utilities package.

By almost any measure, the PEPE project was successful. From the viewpoint of its developers, it met or exceeded all schedule, cost, and performance goals. From the viewpoint of its users, it is reliable and easy to use and program. From the viewpoint of its sponsor, the U.S. Army Ballistic Missile Defense Advanced Technology Center, it is achieving the claims made for it.

In retrospect, the relative lack of technical problems on the PEPE project, not common in supercomputer development experience, can be traced to two factors, both unique to the PEPE project. First, the ballistic missile defense community approaches development projects somewhat differently. BMD systems must work the first time, even though their designers can never be certain how and in what environment they will be used. Moreover, they cannot be tested. BMD system designers therefore rely heavily on simulations, detailed and sometimes tedious design reviews, and extensive "what if" exercises to find and remove all conceivable objections. This approach environment, translated to the PEPE project, resulted in an uncommonly large amount of effort in testing architectural concepts via simulation before proceeding with detailed design work. Also, more than usual emphasis was placed on reliability and excess computing capacity to allow for growth.

The second reason for the success of the PEPE Project was the consolidation of all hardware and software development and initial user responsibility within one project organization. Thus, users had a strong, even predominating, influence on the architecture and the support software right from the start of the project.

Some lessons, of possible value to future supercomputer developers, were learned on the PEPE project. These follow:

1. Start problem programming early, even before the paper design is complete. Much can be learned about user-level system behavior just by writing programs without running them.

2. Employ discrete-event functional simulations early to uncover system bottlenecks and cases of over or under-utilization of machine resources. A combination of such simulations and problem coding can in effect provide fairly thorough user-level experience on the machine while paper design work is still in progress, and while changes can still be made easily.

3. Be conservative in predicting and announcing performance before the machine is operating and delivered. This rule was followed rigorously throughout the PEPE project; consequently, PEPE has exceeded just about every claim made for it. Needless to say, this both astounds and pleases users and sponsors.

4. Be conservative in hardware design, particularly in selecting technology. Advancing the state of the art in architecture, problem implementation, and hardware technology is too much for supercomputer developers to achieve simultaneously.

omit

MATCHING MACHINE AND LANGUAGE

Jackie Kessler
Burroughs Corp.
Paoli, Pennsylvania

A conscious design decision was made by Burroughs to design their early large scale machines, as the B5500, to the user's problem and to the primary high level language of the machine. This matching of the language and machine resulted in ease of use, programmability and high efficiency for the user. For Burroughs it meant a simple manageable interface, i.e., the compiler, between user and machine which would be written in this primary high level language and which could be easily maintained.

The success of this early decision led Burroughs and the design team on the Scientific Processor to adopt the same philosophy. This time, however, the target language was FORTRAN and the problems were of the large scientific number crunching variety. Extensive analysis was performed on production and research codes that spanned the expected user problem space. Loops were studied to determine such quantities as depth of nesting, types of loop parameters, structure and scope of these loops. Additionally the access patterns within loops, the data dependency between array values and the control structures in the loops were analyzed as well as the changes in nesting and loop parameters between loops.

What evolved from these studies were basic requirements and restrictions on the architecture, hardware and software for any general-purpose large scale scientific processor.

Perhaps the most important concept was the development of vector forms or templates which are executed easily on the machine and which are a direct translation of FORTRAN assignment statements. Again as in the B5500 it has been possible to match language and machine in such a fashion that the interface, the compiler, is straight forward and manageable. Additionally the user has direct access to the power of the machine in a high level language with which he is familiar. Because of this simplicity of the basic compiler recent advances in optimization and vectorization techniques can be added in a modular fashion.

RISK TAKING -- AND SUPERCOMPUTERS

Neil Lincoln
Research & Advanced Design Labs
Control Data Corporation
Arden Hills, Minnesota

The never-ending demand for greater and greater computational power to solve allegedly significant problems provides a challenge which lures a few hardy manufacturers and sparse but stalwart users in the implementation of yet another "super-computer". The very nature of this seemingly insatiable demand dictates that such super equipment will be designed and built with the latest technologies available (or almost available) and will be based on architectural concepts just a 'tad' bit beyond the programming state-of-the-art.

It is not clear that those co-developers in the past, while apparently assuming the risks, really ever understood the magnitude or impact of the various effects of living on the frontiers of hardware and software technology. For example, a manufacturer must make a decision about the type of circuit family to be used in a computer to be 'powered-on' in five years. To opt for utilization of an existing, mature circuitry, would obviously not provide the maximum speeds obtainable when the computer is put into operation. In an effort to produce the fastest 'whiz-bang' imaginable then, one has to engage in a guessing game about the probability that a particular logic system now undergoing development will be available in mass quantities of acceptable quality by the time the new 'super' is to be constructed. To be certain, the semiconductor industry is much more experienced and predictable than it was in the early days of super-computer development. However, the choice of building materials for such computational engines is not limited to circuitry alone. The high power densities implied by super-computing requires advances in power supplies, bussing and cooling as well as in circuit board technology. To achieve an aggressive performance goal then, the manufacturer may have to make a frontal assault on the art of producing all of the related technologies. The possibilities of missing performance, reliability and schedule objectives are obvious.

Can we produce the potential for missteps along the path to another computing behemoth? At the very least we can reduce the dollar impact of a hiccup in technology development, and eliminate the cost of architectural imperfections through the use of 'soft-prototypes'. There exists in several forms (the Control Data STAR-100 and 7600 computers being modest examples) the capability to fully simulate the behavior and circuit logic of a complete new supercomputer processor. Thus the manufacturer and user can 'fly before buy' using an accurate simulation of the mainframe on a critical code. Major capital investment can be postponed until after a complete design has been verified with actual production programs.

With the use of existing supercomputers to provide design and documentation assistance, coupled with a design validation tool, one aspect of the supercomputer production process yet remains in the hands of the human. The prediction of technology futures, the creation of supporting technologies and the decision to adopt a particular technological direction are essential to assuring that the resultant technology matches the logic family used in the simulation system. This requires a blend of unique and rare human skills involving semiconductor industrial exposure, packaging acumen, a bit of creative genius, and some luck. We will all still have to rely on the judgements of such people to guide us successfully through the maze of risks and tradeoffs to complete that 'future' machine. And then of course it would be extremely helpful if there was a 'smidgen' of good management...

OMIT

SUMMARY OF COMMENTS

J. E. Thornton
Network Systems Corp.
Brooklyn Center, Minnesota

My comments this afternoon are about people and organization. I believe it is useful to this group to examine how these huge machines are created. You might expect, for example, that an organization is assembled much like a symphony orchestra. Then after much tune-up, this assemblage of talent produces a wonderful performance. The development of a supercomputer is not a performance, however. It is much more like the composition and arrangement of music, usually done by one person.

Going on with this thought, one could compare the development to a relay race. Several runners make their individual efforts in sequence, handing the baton to the next. This comes a bit closer, since no one person could achieve the development of a modern supercomputer without taking so long that the basic technology would be obsolete. The problem with the relay race approach is that it is sequential and critically dependent on each individual runner.

No, I think the real analogy is mountain climbing. Here there is the team effort, the base camp, the sheer terror at times, and the inspiration of great achievement. There is occasional critical dependence on individual performance. Setbacks are progressively more serious as the team nears the summit. The penalty becomes longer and more costly.

In my experience, this matter of individual performance is the most difficult to cope with, to plan around, or to fix. In my current situation of a start-up company, my job is to get the money, get the staff, and then trust them to get it done.

Just as the mountain climbers are often asked, so the supercomputer people could also be asked, "Why do we do it?"

LIST OF ATTENDEES

LIST OF ATTENDEES

1.	Adams, J.C., Jr.	ARO, Inc.
2.	Ahtye, W.	NASA Ames
3.	Anderson, B.N.	NASA Lewis
4.	Arnold, J.O.	NASA Ames
5.	Ashley, H.	Stanford Univ.
6.	Bailey, F.R.	NASA Ames
7.	Baker, A.J.	Univ. of Tennessee
8.	Ballhaus, W.F.	Army Research and Tech. Labs.
9.	Barnes, G.H.	Burroughs Corp.
10.	Batchelder, R.A.	McDonnell Douglas Astronautics Co.
11.	Beam, R.M.	NASA Ames
12.	Berger, S.A.	Univ. of California, Berkeley
13.	Bergmann, M.Y.	NASA Ames
14.	Bertelrud, A.	NASA Ames
15.	Best, D.R.	Texas Instruments
16.	Bhateley, I.C.	General Dynamics Corp.
17.	Birch, S.F.	Boeing Military Airplane Dev.
18.	Black, R.R.	Air Force Flight Dynamics Lab.
19.	Blaine, R.	IBM Science Center
20.	Blottner, F.G.	Sandia Labs.
21.	Bomberger, A.C.	NASA Ames
22.	Bond, J.W.	The Aerospace Corp.
23.	Bower, W.W.	McDonnell Douglas Research Lab.
24.	Boyd, J.W.	NASA Ames
25.	Bradley, R.G.	General Dynamics Corp.
26.	Bright, L.G.	NASA Ames
27.	Brown, H.	Inst. for Advanced Computation
28.	Brown, R.M.	NASA Ames
29.	Brownell, D.H., Jr.	Systems, Science & Software
30.	Buning, P.G.	Univ. of Michigan
31.	Buzbee, B.	Los Alamos Scientific Lab.
32.	Calahan, D.A.	Univ. of Michigan
33.	Carmichael, R.	NASA Ames
34.	Carocci, B.	Floating Point Systems
35.	Castellano, C.	NASA Ames
36.	Cebeci, T.	Douglas Aircraft Corp.
37.	Chang, H.C.	Inst. for Advanced Computation
38.	Chapman, D.R.	NASA Ames
39.	Chapman, G.T.	NASA Ames
40.	Chase, J.B.	Lawrence Livermore Lab.
41.	Chatterjee, B.G.	Inst. for Advanced Computation
42.	Chaussee, D.	Nielsen Engr. and Research
43.	Chen, T.C.	IBM San Jose Research Lab.
44.	Cheng, H.K.	Univ. of Southern California
45.	Chin, J.	Army Research and Tech. Labs.
46.	Clark, J.H.	Univ. of California, Berkeley
47.	Cleary, J.W.	NASA Ames
48.	Coakley, T.	NASA Ames
49.	Coe, C.F.	NASA Ames
50.	Coles, D.	California Inst. of Tech.

51.	Cooper, M.	Office of Naval Research
52.	Cooper, R.E.	Lawrence Livermore Lab.
53.	Cornell, J.A.	System Development Corp.
54.	Crawford, W.L.	NASA Ames
55.	Davis, J.	Consultant
56.	Davy, W.C.	NASA Ames
57.	Deiwert, G.S.	NASA Ames
58.	DeMeritte, F.J.	NASA Hdqrs.
59.	Desideri, J.A.	Iowa State Univ.
60.	Dickey, M.	Cray Research, Inc.
61.	Dickson, L.J.	Boeing Aerospace Co.
62.	Dines, T.R.	NASA Ames
63.	Dix, J.P.	Informatics
64.	Dolkas, J.B.	Consultant
65.	Dongarra, J.	Los Alamos Scientific Lab.
66.	Dorr, F.W.	Los Alamos Scientific Lab.
67.	Downs, H.R.	SAI
68.	DuHamel, S.	NASA Ames
69.	Economidis, F.	Inst. for Advanced Computation
70.	Eddy, R.E.	NASA Ames
71.	Edmonds, S.	Pratt and Whitney Aircraft
72.	Erickson, L.	NASA Ames
73.	Evans, T.	Univ. of Southern California
74.	Feierbach, G.	Inst. for Advanced Computation
75.	Fernandez, G.	NASA Hdqrs.
76.	Fernbach, S.	Lawrence Livermore Lab.
77.	Ferziger, J.H.	Stanford Univ.
78.	Fidler, J.E.	Nielsen Engr. and Research
79.	Field, A.O., Jr.	Inst. for Advanced Computation
80.	Fineberg, M.S.	McDonnell Douglas Automation Co.
81.	Firth, D.	NASA Ames
82.	Flachsbart, B.	McDonnell Douglas Automation Co.
83.	Fornberg, B.	California Inst. of Tech.
84.	Frick, J.	Informatics
85.	Friedman, D.	McDonnell Douglas Corp.
86.	Fromm, J.E.	IBM Research
87.	Fung, L.W.M.	NASA Goddard
88.	Gardner, R.K.	Burroughs Corp.
89.	George, M.	Northrop Corp.
90.	Gessow, A.	NASA Hdqrs.
91.	Gilliland, M.C.	Denelcor
92.	Glatt, L.	The Aerospace Corp.
93.	Goodrich, A.R.	Inst. for Advanced Computation
94.	Goodrich, W.	NASA Johnson
95.	Goorjian, P.	Informatics
96.	Green, M.J.	NASA Ames
97.	Gregory, T.J.	NASA Ames
98.	Gritton, E.C.	The Rand Corp.
99.	Gunn, M.	Inst. for Advanced Computation
100.	Hall, W.E.	Burroughs Corp.
101.	Hankey, W.L.	Air Force Flight Dynamics Lab.
102.	Hansen, J.	NASA Goddard
103.	Harris, J.F.	NASA Langley
104.	Hartmann, M.J.	NASA Lewis
105.	Hathaway, W.	NASA Ames

106. Hausman, R.	Floating Point Systems
107. Hendrickson, C.P.	Lawrence Livermore Lab.
108. Hendrickson, R.	Cray Research, Inc.
109. Hicks, R.	NASA Ames
110. Hirsh, J.E.	Aeronautical Res. Assoc. of Princeton
111. Holst, T.L.	NASA Ames
112. Holt, M.	Univ. of California, Berkeley
113. Horstman, C.C.	NASA Ames
114. Hung, C.M.	DCW Industries
115. Hutchinson, W.H.	NASA Ames
116. Inouye, M.	NASA Ames
117. Ives, D.C.	Pratt and Whitney Aircraft
118. Janac, K.	EAI
119. Johnson, D.A.	NASA Ames
120. Jones, W.P.	NASA Ames
121. Kascic, M.J.	Control Data Corp.
122. Katsanis, T.	NASA Lewis
123. Kautz, W.H.	SRI International
124. Kessler, J.	Burroughs Corp.
125. King, W.S.	The Rand Corp.
126. Klineberg, J.M.	NASA Hdqrs.
127. Kogge, P.	IBM Federal Systems
128. Kohn, J.	Lawrence Livermore Lab.
129. Kolsky, H. G.	IBM Scientific Center
130. Kransky, V.	Lawrence Livermore Lab.
131. Langlois, W.E.	IBM Research
132. Leonard, A.	NASA Ames
133. Levesque, J.M.	R & D Associates
134. Lewis, C.H.	Virginia Poly.Inst. and State Univ.
135. Lewis, G.E.	Inst. for Advanced Computation
136. Lim, R.	NASA Ames
137. Lin, T.C.	The Aerospace Corp.
138. Lincoln, N.R.	Control Data Corp.
139. Linz, P.	Univ. of California, Davis
140. Locanthi, B.	California Inst. of Tech.
141. Lockman, W.K.	NASA Ames
142. Lomax, H.	NASA Ames
143. Lombard, C.K.	Lockheed Palo Alto Research Lab.
144. Long, G.	Lawrence Livermore Lab.
145. Lores, M.E.	Lockheed-Georgia Co.
146. Lund, C.M.	Lawrence Livermore Lab.
147. Lundell, J.	NASA Ames
148. Lundgren, J.	Informatics
149. Lyle, G.	NASA Ames
150. Lynch, J.T.	Burroughs Corp.
151. MacCormack, R.W.	NASA Ames
152. MacKay, J.S.	NASA Ames
153. Madden, J.F.	NASA Hdqrs.
154. Marschner, B.W.	Colorado State Univ.
155. Marshall, J.	Floating Point Systems
156. Martin, E.D.	NASA Ames
157. Marvin, J.G.	NASA Ames
158. McClary, J.F.	Los Alamos Scientific Lab.
159. McCluskey, E.J.	Stanford Univ.
160. McCoy, M.	Lawrence Livermore Lab.

161.	McDevitt, J. B.	NASA Ames
162.	McHugh, R.A.	Control Data Corp.
163.	McMahon, F.H.	Lawrence Livermore Lab.
164.	McMillan, O.J.	Nielsen Engr. and Research.
165.	McRae, D.S.	Air Force Flight Dynamics Lab.
166.	Melnik, R.E.	Grumman Aerospace Corp.
167.	Mendoza, J.P.	NASA Ames
168.	Merriam, M.L.	NASA Ames
169.	Morin, M.K.	NASA Langley
170.	Morris, W.H.	Control Data Corp.
171.	Murdock, J.W.	The Aerospace Corp.
172.	Murphy, D.	NASA Ames
173.	Nachtsheim, P.R.	NASA Ames
174.	Ndefo, E.	The Aerospace Corp.
175.	Nielsen, J.N.	Nielsen Engr. and Research
176.	Nixon, D.	NASA Ames
177.	Norin, R.S.	Floating Point Systems
178.	Olson, L.E.	NASA Ames
179.	Orbits, D.A.	Univ. of Michigan
180.	Owen, F.K.	Consultant
181.	Owens, J.L.	Lawrence Livermore Lab.
182.	Paul, G , Jr.	IBM Corp.
183.	Payne, F.R.	Univ. of Texas, Arlington
184.	Pease, M.C.	SRI International
185.	Pegot, E.	NASA Ames
186.	Perrott, R.H.	Inst. for Advanced Computation
187.	Petersen, R.H.	NASA Ames
188.	Peterson, V.L.	NASA Ames
189.	Potter, J.L.	ARO, Inc.
190.	Pratt, M.W.	Lawrence Livermore Lab.
191.	Presley, L.	NASA Ames
192.	Pritchett, P.	Univ. of California, Los Angeles
193.	Pulliam, T.	NASA Ames
194.	Rakich, J.	NASA Ames
195.	Redhed, D.D.	Boeing Computer Services
196.	Reklis, R.P.	Army Ballistic Research Lab.
197.	Roberts, L.	NASA Ames
198.	Roepke, R.G.	Air Force AEDC
199.	Rollwagen, J.A.	Cray Research, Inc.
200.	Rosen, R.	McDonnell Douglas Astronautics Co.
201.	Rossow, V.J.	NASA Ames
202.	Rubbert, P.E.	Boeing Aerospace Co.
203.	Rubesin, M.W.	NASA Ames
204.	Rudy, T.	Lawrence Livermore Lab.
205.	Runchal, A.K.	Dames and Moore
206.	Saunders, R.	Cray Research, Inc.
207.	Schiff, L.	NASA Ames
208.	Schneider, V.	The Aerospace Corp.
209.	Schulbach, C.	NASA Ames
210.	Schwenk, F.C.	NASA Hdqrs.
211.	Sedney, R.	Army Ballistic Research Lab.
212.	Sharbaugh, L.	Informatics
213.	Shavitt, I.	Battelle Columbus Lab.
214.	Sinz, K.	Lawrence Livermore Lab.
215.	Sloan, L.	Lawrence Livermore Lab.

216. Smith, B.F.	NASA Ames
217. Smith, M.C.	NASA Ames
218. Sorenson, R.	NASA Ames
219. Steger, J.L.	NASA Ames
220. Steinhoff, J.	Grumman Aerospace Corp.
221. Stevens, K.G., Jr.	NASA Ames
222. Stevenson, D.	Inst. for Advanced Computation
223. Stone, H.S.	Univ. of California, Berkeley
224. Sumner, F.H.	IBM Research Division
225. Syvertson, C.A.	NASA Ames
226. Tanner, J.G.	NASA Ames
227. Tavenner, M.S.	Air Force Systems Command Liaison Office
228. Taylor, T.D.	The Aerospace Corp.
229. Thames, F.C.	Vought Corp.
230. Thomas, P.D.	Lockheed Palo Alto Research Lab.
231. Thompkins, W.T.,Jr.	Massachusetts Inst. of Tech.
232. Thompson, J.F.	Mississippi State Univ.
233. Thornton, J.E.	Network Systems Corp.
234. Tindle, E.	NASA Ames
235. Toon, O.B.	NASA Ames
236. Tower, D.	Denelcor
237. Treon, S.L.	NASA Ames
238. Trimble, J.	Office of Naval Research
239. Tsuge, S.	Nielsen Engr. and Research
240. Viegas, J.	NASA Ames
241. Vigneron, Y.C.	Iowa State Univ.
242. Vinokur, M.	Univ. of Santa Clara
243. Voight, R.G.	ICASE
244. Wagenbreth, G.	R & D Associates
245. Wakefield, S.	Stanford Univ.
246. Waller, G.W.	R & D Associates
247. Wang, L.P.T.	Univ. of California, Los Angeles
248. Warming, R.F.	NASA Ames
249. Watson, V.	NASA Ames
250. White, H.S.	Lawrence Berkeley Lab.
251. White, J.S.	NASA Ames
252. Whitfield, J.D.	ARO, Inc.
253. Whiting, E.	NASA Ames
254. Widhopf, G.	The Aerospace Corp.
255. Winslow, A.M.	Lawrence Livermore Lab.
256. Wirsching, J.E.	Burroughs Corp.
257. Woodward, F.A.	Analytical Methods, Inc.
258. Wooler, P.T.	Northrop Corp.
259. Wu, J.C.	Georgia Inst. of Tech.
260. Yen, K.T.	Naval Air Development Center
261. Yen, S.M.	Univ. of Illinois
262. Yoshihara, H.	Boeing Co.
263. Zagotta, W.E.	Lawrence Livermore Lab.

1. Report No. NASA CP-2032	2. Government Accession No.	3. Recipient's Catalog No
4. Title and Subtitle FUTURE COMPUTER REQUIREMENTS FOR COMPUTATIONAL AERODYNAMICS*		5. Report Date
		6. Performing Organization Code
7 Author(s)		8. Performing Organization Report No. A-7291
9. Performing Organization Name and Address NASA Ames Research Center Moffett Field, California 94035		10. Work Unit No. 505-06-11
		11. Contract or Grant No.
		13. Type of Report and Period Covered Conference Proceedings
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes

*A workshop held at NASA Ames Research Center, Moffett Field, California, October 4-6, 1977.

16. Abstract

This report is a compilation of papers presented at the NASA Workshop on Future Computer Requirements for Computational Aerodynamics. The Workshop was held in conjunction with preliminary studies for a Numerical Aerodynamic Simulation Facility that will have the capability to solve the equations of fluid dynamics at speeds two to three orders of magnitude faster than presently possible with general purpose computers. Summaries are presented of two contracted efforts to define processor architectures for a facility to be operational in the early 1980's.

17. Key Words (Suggested by Author(s)) Numerical analysis Computer sciences	18. Distribution Statement Unlimited STAR Category - 59

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 517	22. Price* $12.75

*For sale by the National Technical Information Service, Springfield, Virginia 22161

U.S.GPO:1978-793-973/184