NASA Technical Memorandum 84604

Use of CYBER 203 and CYBER 205 Computers for Three-Dimensional Transonic Flow Calculations

N. Duane Melson and James D. Keller

APRIL 1983

25th Anniversary
1958-1983

NASA

NASA Technical Memorandum 84604

Use of CYBER 203 and CYBER 205 Computers for Three-Dimensional Transonic Flow Calculations

N. Duane Melson and James D. Keller
Langley Research Center
Hampton, Virginia

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1983

SUMMARY

Experiences are discussed for modifying two three-dimensional transonic flow computer programs (FLO 22 and FLO 27) for use on the CDC® CYBER 203 computer system at the Langley Research Center. Both programs discussed were originally written for use on serial machines. Several methods were attempted to optimize the execution of the two programs on the vector machine: leaving the program in a scalar form (i.e., serial computation) with compiler software used to optimize and vectorize the program, vectorizing parts of the existing algorithm in the program, and incorporating a new vectorizable algorithm (ZEBRA I or ZEBRA II) in the program. Comparison runs of the programs were made on CDC® CYBER 175, CYBER 203, and two-pipe CDC® CYBER 205 computer systems.

INTRODUCTION

Most research in the computation of transonic flows in recent years has been to improve the accuracy and geometric capability of computational tools. Most three-dimensional transonic codes, however, still use large amounts of computer resources and are expensive to use extensively, such as in parametric and optimization studies. Thus, there is a need for improving the computational efficiency (and reducing the cost) of these codes.

Two ways to reduce run costs and run times are through improvements in (1) convergence rate and (2) calculation rate. The way to increase convergence rate is by the use of improved iteration algorithms to solve the governing equations. To date, the true workhorse of potential flow calculations for transonic flow has been the successive line overrelaxation (SLOR) algorithm. SLOR is a very robust algorithm and is relatively easy to program. For these reasons, it has found widespread acceptance and use. Unfortunately, SLOR requires many iterations to obtain a converged solution; this can make three-dimensional calculations very expensive.

The most straightforward way to improve calculation rate is through the use of larger, faster computers. Three examples of these machines are the CDC® STAR-100, the CDC® CYBER 203, and the CDC® CYBER 205 computer systems. These machines are all known as vector processors. They obtain high calculation rates through unique architecture which performs operations on groups of operands (vectors). Unfortunately, the advantages gained by vector processing are not without some penalty. To use vector instructions, all the operands must be available before execution of the instruction is begun. In iterative algorithms where updated values at adjacent points are used to calculate values at a given point, this requirement is very restrictive. Quite often the direct application of partially implicit algorithms, such as SLOR, on a vector machine results in either the inefficient use of its vector-processing abilities or the necessity to resort to slower scalar operations. On the other hand, an explicit algorithm which is easily vectorized may have a much slower convergence rate. Hence, the efficient use of a vector-processing machine becomes a trade-off between calculation rate, convergence rate, and ease of programming.

In this paper, experiences are discussed for modifying two three-dimensional transonic flow computer programs (FLO 22 and FLO 27) for use on the CYBER 203

computer at the Langley Research Center. Both programs discussed herein were originally written for use on serial machines. Several methods were attempted to optimize the execution of these programs on the vector machine: (1) using the existing compiler software to optimize and vectorize the scalar code, (2) vectorizing parts of the existing algorithm in the program, and (3) incorporating a new vectorizable algorithm in the program (ZEBRA I or ZEBRA II). Comparison runs of the programs were made on CYBER 175, CYBER 203, and two-pipe CYBER 205 computers. All calculations discussed in this paper were for the ONERA M6 wing as described in reference 1. The free-stream conditions were a Mach number of 0.84 and an angle of attack of 3.06°.

Jon Hall at Control Data Corporation ran the various versions of the FLO 22 code on the CYBER 205 computer system.

VECTOR PROCESSORS

The STAR-100, the CYBER 203, and the CYBER 205 computers are all vector processors. In these machines, each operation is broken into many steps which are performed in series to obtain the result of the operation (ref. 2). This is a production-line type of process; once the first result is produced, the others follow very quickly. The time required to produce the first result is referred to as the start-up time. These vector processors contrast in operation to a scalar or serial processor where all the steps involved in a given operation are performed on one set of operands before anything is done with the next set. The crossover vector length, where a vector instruction is faster than a series of scalar instructions, is a function of the ratio of vector speed to scalar speed.

Vector length is an important factor in the calculation rate for these vector processors. The STAR-100, CYBER 203, and CYBER 205 are pipeline processors which can operate on vectors as long as 65 535. They operate more efficiently as the vector length increases and long vectors should be used whenever possible.

It is important to limit the use of scalar operations since such operations are generally slower than vector operations. The scalar speed of the STAR-100 was about one-seventh that of the CYBER 203, but the vector speeds were about the same. On the CYBER 203, the ratio of vector speed to scalar speed is not very high; therefore, some scalar code may be used without a large penalty. On the CYBER 205, where the vector speed is significantly faster and the scalar speed is only slightly faster than the CYBER 203, it is important to vectorize the code rather than leaving it in scalar form.

IMPLEMENTATION OF CODES

To examine ways to make the most efficient use of the vector-processing abilities of the CYBER 203, two programs were studied to determine the trade-offs necessary to best implement the programs. The two codes investigated were FLO 22 and FLO 27. FLO 22 is the Jameson-Caughey transonic wing-alone program (refs. 3 and 4) which solves the nonconservative full-potential equation in finite-difference form. FLO 27 is another Jameson-Caughey transonic program (ref. 5), but it is for a wing alone or a wing on a cylindrical fuselage of infinite length. FLO 27 solves the conservative full-potential equation in finite-volume form. Both codes were originally written for use on a conventional computer and use the SLOR iterative algorithm. Of the two programs, FLO 22 was studied in the most detail.

FLO 22 Study

FLO 22 solves the inviscid flow about swept or yawed wings by using a sheared parabolic coordinate system in chordwise planes. Over the past several years, it has been the most widely used program in the aircraft industry for transonic wing-alone calculations and was therefore chosen for implementation on the CYBER 203.

In the present study and another study conducted at Langley, four different approaches were taken to implement FLO 22 on the CYBER 203. The simplest improvement was to put the program on the machine in scalar form and to use the large central memory and virtual memory architecture of the CYBER 203 to replace the buffered input/output features in the original code. Various levels of optimization provided by the compiler were used, including a provision for automatic vectorization. The second approach was to explicitly vectorize as much of the program as possible without changing the original order of calculations or the iterative algorithm. The original algorithm and order of calculations in FLO 22 allowed only limited vectorization and short vector lengths. In the third approach, a highly vectorizable algorithm, ZEBRA II (ref. 6), was incorporated into the program. This incorporation was done in conjunction with a reorganization of the storage for efficient execution and the resulting program was then written with vector instructions. It was found that this vectorized version of ZEBRA II obtained an excellent calculation rate but the convergence rate suffered to such a degree that a net negative effect was obtained with the incorporation of ZEBRA II. To improve the convergence rate, another algorithm, ZEBRA I (ref. 6), was incorporated into the program in the fourth part of the FLO 22 study. ZEBRA I gave a convergence rate comparable to the original SLOR algorithm but this was in conjunction with a penalty in calculation rate due to factors discussed subsequently.

Scalar FLO 22.- The scalar version of FLO 22 which was available at the beginning of this study was a serial code adapted for use on a standard CYBER 175 type machine, that is, no extended core capability. To operate on a CYBER 175, it was necessary to use special input/output commands to move the potential array in and out of central memory a plane at a time since the storage was not sufficient for the entire array in central memory. Only four planes at a time were kept in central memory.

Execution of the serial FLO 22 program on a grid with 192 points in the x or tangential direction, 24 points in the y or normal direction, and 32 points in the z or span direction (192 by 24 by 32) on a CYBER 175 with the highest level of compiler optimization (OPT = 2) gave a calculation rate of 7400 grid points per second (pps). (See table I.)

The scalar CYBER 175 version of FLO 22 was modified so that it would run in central memory on a CYBER 203. This modification involved the removal of all special input/output statements and the correction of the z index for each reference to the potential function. The scalar version of FLO 22 was executed on the CYBER 203 on a 192 by 32 by 32 grid and a calculation rate of 16 650 pps was obtained. The program was then recompiled with the optimizing version of the FORTRAN compiler (OPT = BO) which eliminates redundant code, optimizes DO loops, and does instruction scheduling. A calculation rate of 48 800 pps was then obtained. Thus, use of the optimization feature of the compiler can make a scalar program run significantly faster and is recommended. With automatic vectorization also included in the compilation (OPT = BOV), the same calculation rate was obtained. This is the first of several examples presented in this paper which show that the automatic vectorization option rarely is able to vectorize code in this type of program. Since the amount of work

required to get FLO 22 running on the CYBER 203 in scalar form was small, the speedup from 7400 to 48 800 pps made the effort very worthwhile.

Vector FLO 22.- During the period of time when Langley had a STAR-100 computer, it was desirable to vectorize this code since the scalar performance of that computer was poor relative to its vector performance. The initial vectorization effort described herein occurred during that period. This initial vectorization of the code was performed by a group of researchers at Langley and is reported in detail in reference 7. Since this work is so closely related to the present study, a brief description of the results is given in the following discussion.

In Jameson's original FLO 22 code, calculation of updated values of the potential are obtained by successive line overrelaxation (SLOR) for combinations of normal and tangential lines in chordwise planes, one plane at a time, starting at the root and going out the span. In each plane, line overrelaxation is performed by tangential lines in the region in front of the nose of the airfoil section to infinity and by normal lines from this region to infinity off the trailing edge for both the upper and lower surfaces. (See fig. 1.) The extent of the tangential implicit lines in the central strip off the leading edge can be varied by an input parameter from the case where all relaxation is performed by using line overrelaxation along tangential lines to the case where all relaxation is performed by using normal lines.

To make vectorization easier and to maximize their vector length, Smith, Pitts, and Lambiotte (ref. 7) vectorized only the tangential overrelaxation routines. This choice allowed them to use vector lengths equal to the number of grid points in the x-direction (192 for a 192 by 32 by 32 grid as used in the test case considered in this paper). This vector length was used for all the calculations necessary to generate the residual at each point along a given tangential line. (The residual is herein defined as the result of the finite-difference operator operating on the values of the potential function.) It is very important to have a long vector length for the residual calculation since this calculation is about 90 percent of the computational work in this potential flow code.

Since the vector length used in a calculation greatly affects the calculation rate, it is obvious that the number of grid points in the tangential direction will affect the calculation rate of the vectorized FLO 22 code. There is a crossover length below which scalar instructions are faster than vector instructions. Because of the superior scalar speed of the CYBER 203, the crossover is higher than on the STAR-100 for which the vectorized code was originally developed. In fact, for some of the coarser grids the scalar CYBER 203 code was found to be faster than the vectorized version.

There is another consideration involving vector length for this code. FLO 22 uses central differences at subsonic points and backward differences at supersonic points. This causes some difficulty in vectorizing the code. The vectorized version of FLO 22 calculates the residual at all the points on a line using central differences. This residual is not correct for the supersonic points on that line and must be recalculated. One alternative in this recalculation is to use vector instructions to calculate the residual at all points on the line by using backward differences (as if all the points were supersonic) and use the results of this calculation only at the supersonic points. This is effective if there are a large number of supersonic points in the line. If the number of supersonic points is below an experimentally obtained value, it is faster to calculate the supersonic residuals at only the supersonic points by using scalar instructions. Lambiotte found that for this code on the

CYBER 203, it was better to use scalar instructions if there were less than 40 supersonic points on a given line.

The vectorized FLO 22 code, working on a grid size of 192 by 32 by 32, operates at 50 100 pps with both the BO and BOV levels of optimization. Although this was a considerable improvement over the STAR-100 scalar code, it is only modestly better than the scalar code which can now be run on the CYBER 203. Interestingly enough, the improved vector performance of the CYBER 205 will undoubtedly reverse this comparison again. It is possible to predict the performance of the scalar and vector versions of the original FLO 22 code on the CYBER 205 by using performance estimates discussed in the following section. These results show that the scalar code runs approximately 20 percent faster on the CYBER 205 than on the CYBER 203. Hence, the scalar version of FLO 22 should run at about 58 600 pps on the CYBER 205. Vector code tends to run 2 1/2 to 3 times faster on the CYBER 205. Therefore, the vectorized version of FLO 22 should run at 125 000 to 150 000 pps. Thus, the vector version of the original FLO 22 code should be significantly faster than the scalar version on the CYBER 205.

FLO 22 with ZEBRA II algorithm.- In an effort to improve the calculation rate of the vector FLO 22 code, an explicit vectorizable algorithm was incorporated. This algorithm, called ZEBRA II, was developed by South, Keller, and Hafez (ref. 6).

Basically, ZEBRA II is a two-color point relaxation scheme where each cross plane is a checkerboard pattern. (See fig. 2.) The cross planes are aligned in such a way that each color traces out tangential lines. This alignment means that all the points of one color in a cross plane can be updated before points of the other color are updated.

The incorporation of ZEBRA II in FLO 22 was initially done with scalar instructions. First, it was necessary to rewrite parts of the code to allow sweeps through planes in the cross flow direction rather than the chordwise direction. The residual calculation was also changed to the calculation of a steady-state residual with explicit updates added. (The steady-state residual is the result of the finite-difference operator operating on the values of the potential function generated by the previous global iteration.)

The calculation rate for the scalar version of FLO 22 with ZEBRA II was 44 700 pps for both the BO and BOV levels of optimization on the CYBER 203. With no optimization, a calculation rate of only 13 400 pps was obtained. This speedup due to optimization is not uncommon for scalar codes such as the FLO 22 with ZEBRA II.

The FLO 22 code with the ZEBRA II algorithm was then rewritten with vector instructions. It is possible to use vector instructions to calculate the steady-state residual for an entire cross plane at a time. Then, updates are performed on the points of the first color, black, and then on points of the second color, white, of the cross-plane checkerboard. The residuals at the white points are updated by using corrections at the adjacent black points. Calculation then proceeds to the next downstream cross plane. Thus, the length of the vectors used in calculating the residual is equal to the number of points in a cross plane. Because of the requirement that vector elements be contiguous in storage, it was necessary for the potential to be stored in a two-dimensional array rather than in a three-dimensional array. In this two-dimensional array, the first index refers to the x-direction and the second index points to all the elements in the given cross plane. (An option of

the compiler was used to invoke a nonstandard array-addressing algorithm so that the last index of the array varied the quickest in contiguous storage to allow vector instructions.)

In order to calculate the residual with full-plane vectors, it was necessary to make all intermediate results temporary vectors the length of the cross plane; this added considerable storage to the program. Again, there was a problem relating to the fact that supersonic points are not treated in the same way as subsonic points. The residual was first calculated at all points in a cross plane by using the subsonic formula. If there were any supersonic points in the plane, corrections to the residual were calculated (again by using full-plane vectors) and applied at only the supersonic points.

With ZEBRA II incorporated into FLO 22 in vector form, the CYBER 203 calculation rate was 59 000 pps with both the BO and BOV optimization levels. For the nonoptimized level of compilation, the computation rate was 57 500 pps. The small difference is because the program is highly vectorized and the optimization only works on scalar code.

To compare the relative speeds of the CYBER 203 and CYBER 205 for transonic flow calculations, both the scalar and vectorized versions of FLO 22 with ZEBRA II were run, unchanged, on a two-pipe CYBER 205. The CYBER 205 gave a calculation rate of 54 100 pps on the scalar version - a 20-percent improvement in scalar calculation rate over the CYBER 203. For the vectorized FLO 22 with ZEBRA II, a rate of 173 800 pps was obtained - a 195-percent improvement in vector calculation rate over the CYBER 203.

The sustained calculation rate for the vectorized residual and update portions of FLO 22 with ZEBRA II was approximately 26 million floating-point operations per second on the CYBER 203 and 76 million floating-point operations per second on the CYBER 205.

Since the developmental work on ZEBRA II in reference 6 was for a conservative formulation of the full-potential equation, it contained no cross derivatives. FLO 22 is nonconservative and, therefore, does contain cross-derivative terms. Hence, it was necessary to experiment with these terms in FLO 22 with ZEBRA II to determine the best mix of old and new values to optimize convergence. This experimentation was done with the constraint of Jameson's rule of balanced coefficients for supersonic points (ref. 8).

The convergence rate of ZEBRA II was not as good in FLO 22 as was anticipated from the work in reference 6, where it was found that ZEBRA II gave a convergence rate comparable to SLOR. In FLO 22, the ZEBRA II algorithm required more than twice as many iterations to obtain the same level of convergence as with the SLOR algorithm. These extra iterations more than canceled the 20-percent increase in calculation rate obtained with the ZEBRA II.

FLO 22 with ZEBRA I algorithm.- To improve the poor convergence rate found with the ZEBRA II algorithm, it was decided to implement an algorithm known as ZEBRA I (ref. 6). This algorithm uses the same arrangement of black and white points as the ZEBRA II algorithm, but the tangential lines are solved implicitly using tridiagonal systems of equations. It is still vectorizable because it solves many independent tridiagonal systems at the same time.

For simplicity, the vector ZEBRA II version of the program was used as the starting point for the ZEBRA I incorporation. The full-plane residual calculation was retained but the whole update portion of the program was changed.

For ZEBRA I, the two back-substitution coefficients used to solve the tridiagonal system of equations generated along tangential lines are calculated for a cross plane and saved as three-dimensional arrays. Since only one color is updated at a time, only one-half the coefficients for each plane are saved. The back-substitution coefficients are calculated one plane at a time until the coefficients for all the black points in the flow field are calculated. As mentioned, the storage of these coefficients requires two arrays, each one-half the size of the entire flow field. The back substitution is then performed by using vector lengths equal to one-half the length of the cross planes. Updates of the potential are performed as the corrections are calculated. Once the black point backward sweep is completed, the residual is recalculated using full cross-plane length instructions and the new values of the potential function at the black points. Then the forward and backward substitutions are done for the white points. An obvious problem with this scheme is the calculation of the residual at each point twice for each iteration. The elimination of the extra calculation of the residual at each point would require reworking nearly all the storage in the program which was not done in this study.

The convergence rate of the ZEBRA I algorithm was found to be as good as the original SLOR algorithm. The calculation rate was only 38 800 pps on the CYBER 203, a direct result of the extra work used to calculate the residual twice at each point. On the CYBER 205, a calculation rate of 124 100 pps was obtained.

It is possible to predict the calculation rate obtainable if the extra residual calculation at each point is eliminated in FLO 22 with ZEBRA I. This requires taking into account the percentage of total work that the residual calculation involves and the reduced vector lengths which would be used for the calculations. For the CYBER 203, a calculation rate of 61 000 pps is projected. If the same rate is used to predict the CYBER 205 performance, a conservative estimate of 195 000 pps is obtained. Thus, the efficient implementation of the ZEBRA I algorithm would result in a significant increase in calculation rate without any degradation in convergence rate.

FLO 27 Study

FLO 27 is a computer code written to analyze the transonic flow over a wing alone or a wing on a cylindrical fuselage. It uses a finite-volume formulation to solve the full-potential equation in conservative form. In the construction of the computational coordinate system, a Joukowski transformation is used to transform the cylindrical fuselage to a vertical slit and then a sheared parabolic transformation is used in planes containing the airfoil sections.

The work performed on the FLO 27 code was limited to the area of programming techniques. No changes were made to the iteration algorithm used in the program (SLOR).

The starting point for this study was a version of FLO 27 which was written for use on the CYBER 175 computer. Its main three-dimensional array of potential functions was stored on disk, and special input/output statements were used to bring planes of data in to central memory and to store updated planes of data back on the disk. This buffering of data was eliminated and the code was modified so that a

large array containing all the values of the potential function was used. This modified code was able to fit in central memory on the CYBER 203.

On the CYBER 175 (with OPT = 2), the original scalar code ran at a rate of 3200 pps on a 160 by 16 by 32 grid. This same version of the code ran at about 3560 pps on the CYBER 203 (with OPT = BO), a speedup of 11 percent.

The authors then investigated a suggestion from Raymond Blanc of Control Data Corporation. He obtained a surprising speedup in FLO 27 by changing two small loops from scalar code to vector code.

A timing study was performed on the subroutine which accounts for the majority of the execution time for the program. This subroutine, YSWEEP, contains about 350 lines of code. In the main portion of YSWEEP, many arithmetic operations are performed for each point in the computational grid. A breakdown of the number of operations is as follows:

Operation	Occurrences
Additions and subtractions	249
Multiplications	142
Divisions	7
Square roots	5
Sine or cosine	4
ATAN2 function	2

Although there were only two evaluations of the ATAN2 function, the timing study gave the surprising result that these two operations took 77 percent of the time used in this portion of the code! The two ATAN2 functions happen to be in a loop which could be changed easily to allow the use of the vectorized version of the ATAN2 function (VATAN2). With a vector length of 160, VATAN2 is about 32 times as fast as ATAN2. This minor coding change caused the calculation rate for the main iteration loop to increase from 3560 pps to 13 320 pps, an increase of 247 percent.

Because of the conservative, finite-volume formulation, the main portion of the FLO 27 code has some simple loops which can be recognized by the compiler as vectorizable. Use of the automatic vectorizing compiler option (OPT = BOV) further increases the calculation rate to 19 460 pps. Thus, with only minor modifications, the calculation rate of FLO 27 was increased from 3200 pps on the CYBER 175 to 19 460 pps on the CYBER 203 - a net increase of over 500 percent.

CONCLUDING REMARKS

Two three-dimensional transonic flow programs (FLO 22 and FLO 27) have been modified for use on CYBER vector processing machines.

From the study of the FLO 22 code, it was found that on the STAR-100 machine the most efficient version of FLO 22 was the Smith-Pitts-Lambiotte code (NASA Technical Memorandum 78665) which used vector instructions with the original SLOR algorithm. On the CYBER 203, it was found that the best version of FLO 22, in terms of the

trade-off between work required to change the code, convergence rate, and calculation rate, was the scalar version of the original code. On the CYBER 205, the long vectors and good convergence rate of the efficiently implemented ZEBRA I version of FLO 22 make it the fastest of the versions of FLO 22 considered in this study.

From the FLO 27 study, it was found that limited vectorization and the replacement of an inefficiently implemented scalar trigonometric function with a system-supplied vectorized routine produced significant improvements in calculation rate over a serial machine. These changes required a minimum of work and were thus deemed to be quite effective.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
February 22, 1983

REFERENCES

1. Monnerie, B.; and Charpin, F.: Buffeting Tests With a Swept Wing in the Transonic Range. NASA TT F-15803, 1974.

2. Knight, John C.: The Current Status of Super Computers. Comput. & Struct., vol. 10, no. 1/2, Apr. 1979, pp. 401-409.

3. Jameson, Antony: Numerical Calculation of the Three Dimensional Transonic Flow Over a Yawed Wing. Proceedings AIAA Computational Fluid Dynamics Conference, July 1973, pp. 18-26.

4. Jameson, Antony; and Caughey, D. A.: Numerical Calculation of the Transonic Flow Past a Swept Wing. COO-3077-140 (Contract EY-76-C-02-3077*000 and NASA Grants NGR-33-016-167 and NGR-33-016-201), Courant Inst. Math. Sci., New York Univ., June 1977. (Available as NASA CR-153297.)

5. Jameson, Antony; and Caughey, D. A.: A Finite Volume Method for Transonic Potential Flow Calculations. A Collection of Technical Papers - AIAA 3rd Computational Fluid Dynamics Conference, June 1977, pp. 35-54. (Available as AIAA Paper 77-635.)

6. South, Jerry C., Jr.; Keller, James D.; and Hafez, Mohamed M.: Vector Processor Algorithms for Transonic Flow Calculations. A Collection of Technical Papers - AIAA Computational Fluid Dynamics Conference, July 1979, pp. 247-255. (Available as AIAA Paper 79-1457.)

7. Smith, Robert E.; Pitts, Joan I.; and Lambiotte, Jules J.: A Vectorization of the Jameson-Caughey NYU Transonic Swept-Wing Computer Program FLO-22-VI for the STAR-100 Computer. NASA TM-78665, 1978.

8. Jameson, Antony: Iterative Solution of Transonic Flows Over Airfoils and Wings, Including Flows at Mach 1. Commun. Pure & Appl. Math., vol. XXVII, no. 3, May 1974, pp. 283-309.

TABLE I.- FLO 22 TIMING RESULTS

Program	Grid	Machine	Optimization level	Calculation rate, pps
Original FLO 22 (scalar)	192 by 24 by 32	CYBER 175	OPT = 2	7 400
	192 by 32 by 32	CYBER 203	No optimization	16 700
			OPT = BO	48 800
			OPT = BOV	48 800
		CYBER 205	OPT = BO	[a]58 600
Original FLO 22 (vector)	192 by 32 by 32	CYBER 203	OPT = BO	50 100
			OPT = BOV	50 100
		CYBER 205	OPT = BO	[a]125 000 to 150 000
FLO 22 with ZEBRA II (scalar)	192 by 32 by 32	CYBER 203	No optimization	13 400
			OPT = BO	44 700
			OPT = BOV	44 700
		CYBER 205	OPT = BO	54 100
FLO 22 with ZEBRA II (vector)	192 by 32 by 32	CYBER 203	No optimization	57 500
			OPT = BO	59 000
			OPT = BOV	59 000
		CYBER 205	OPT = BO	173 800
FLO 22 with ZEBRA I (residual calculated twice (vector))	192 by 32 by 32	CYBER 203	OPT = BO	38 800
		CYBER 205	OPT = BO	124 100
FLO 22 with ZEBRA I (residual calculated once (vector))	192 by 32 by 32	CYBER 203	OPT = BO	[a]61 000
		CYBER 205	OPT = BO	[a]195 000

[a]Projected value.

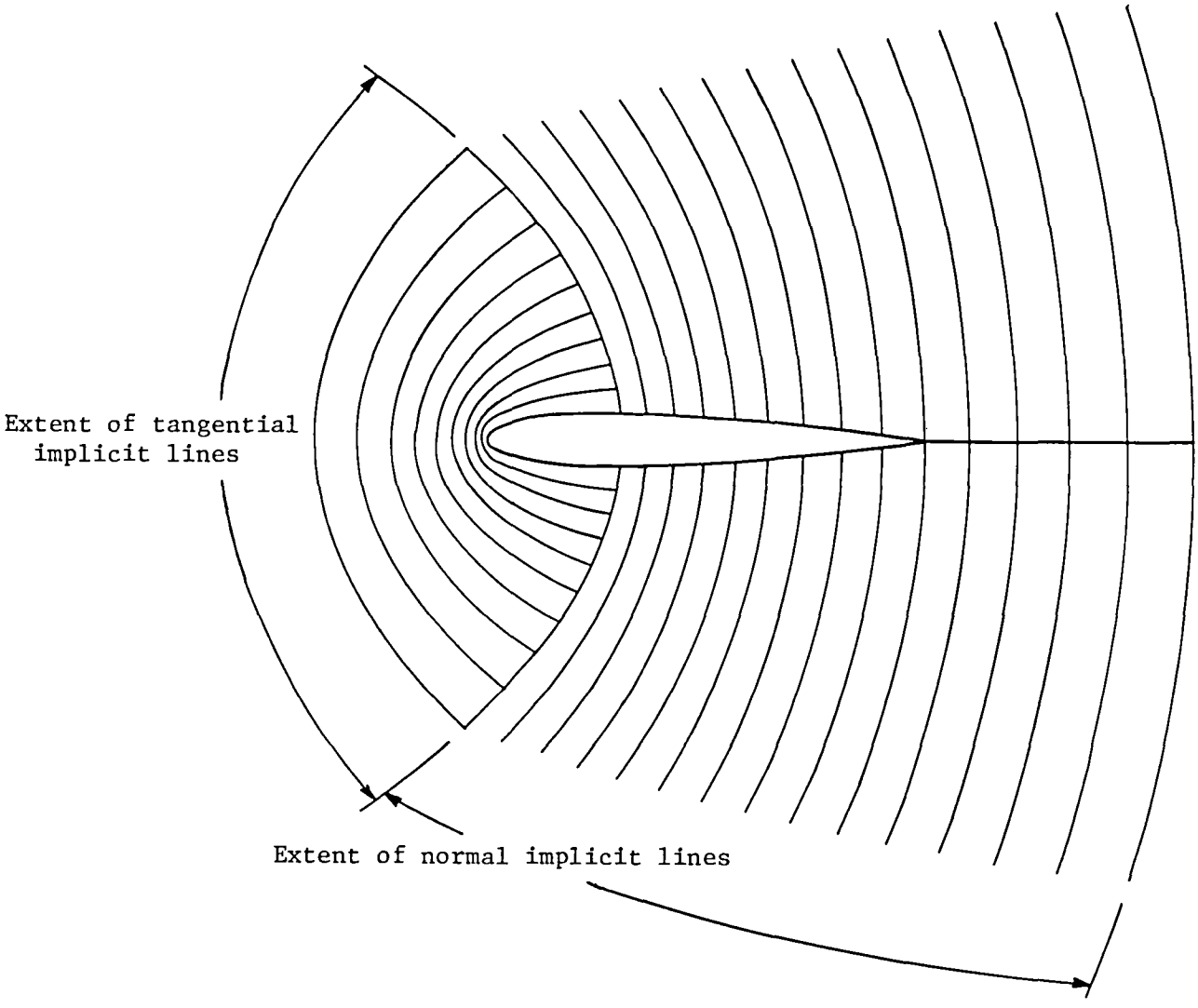

Figure 1.- Schematic of chordwise plane iteration scheme in original FLO 22.

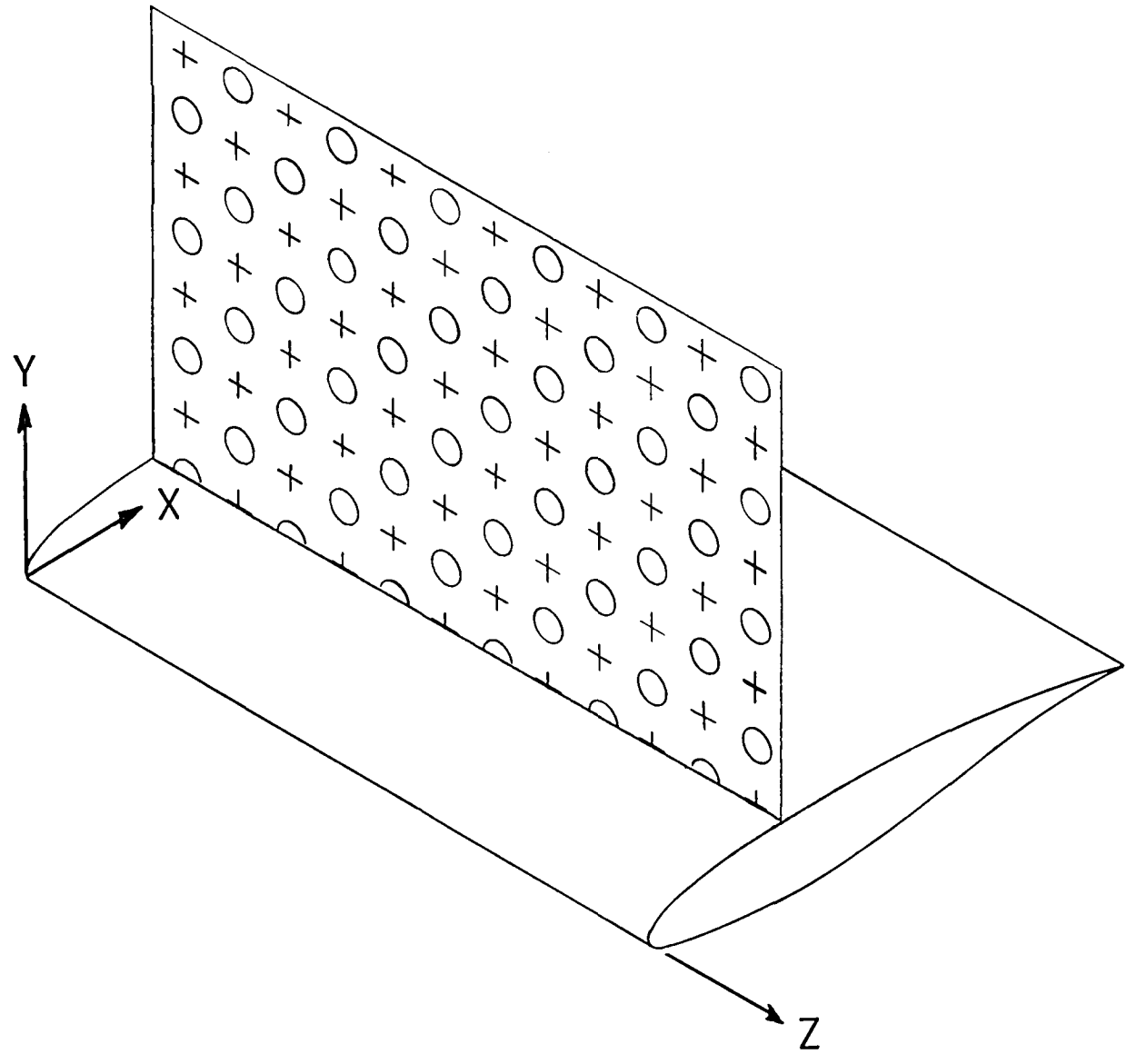

Figure 2.- Schematic of cross-plane checkerboard pattern for ZEBRA algorithm.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-84604		

4. Title and Subtitle	5. Report Date
USE OF CYBER 203 AND CYBER 205 COMPUTERS FOR THREE-DIMENSIONAL TRANSONIC FLOW CALCULATIONS	April 1983
	6. Performing Organization Code: 505-31-03-01
7. Author(s): N. Duane Melson and James D. Keller	8. Performing Organization Report No.: L-15553
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	13. Type of Report and Period Covered: Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Experiences are discussed for modifying two three-dimensional transonic flow computer programs (FLO 22 and FLO 27) for use on the CDC CYBER 203 computer system at the Langley Research Center. Both programs were originally written for use on serial machines. Several methods were attempted to optimize the execution of the two programs on the vector machine: leaving the program in a scalar form (i.e., serial computation) with compiler software used to optimize and vectorize the program, vectorizing parts of the existing algorithm in the program, and incorporating a new vectorizable algorithm (ZEBRA I or ZEBRA II) in the program. Comparison runs of the programs were made on CDC CYBER 175, CYBER 203, and two-pipe CDC CYBER 205 computer systems.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Transonic flow Three-dimensional flow Vector processing FLO 22 FLO 27 ZEBRA I ZEBRA II	Unclassified - Unlimited Subject Category 34, 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	15	A02

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

NASA-Langley, 1983

National Aeronautics and Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

THIRD-CLASS BULK RATE

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

NASA

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return