NASA-CR-197745

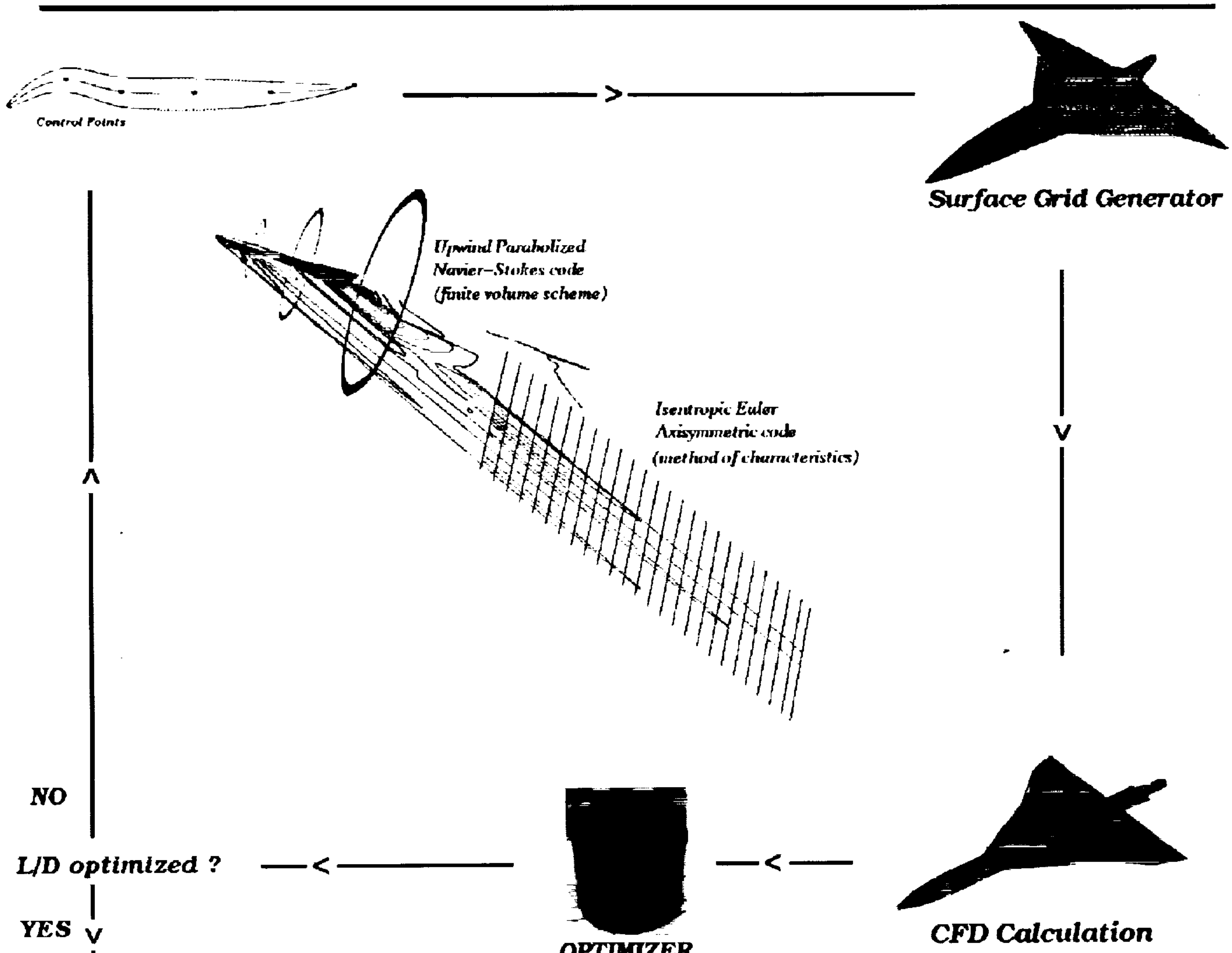
MCAT Institute
Final Report
95-7

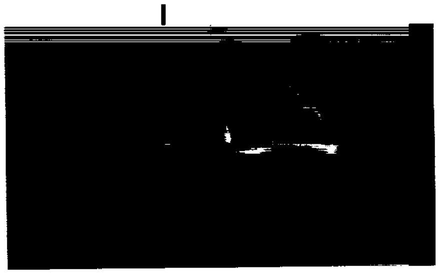
MCAT Institute
3933 Blue Gum Drive
San Jose, CA 95127

# Supersonic Civil Airplane Study and Design: Performance and Sonic Boom

Samson Cheung

January 1995
NCC2-617



(NASA-CR-197745) SUPERSONIC CIVIL AIRPLANE STUDY AND DESIGN: PERFORMANCE AND SONIC BOOM Final Report, Jul. 1989 - Jan. 1995 (MCAT Inst.) 117 p

N95-26813

Unclas

G3/05 0048506

# Supersonic Civil Airplane Study and Design: Performance and Sonic Boom

**Samson Cheung**

This final report summarizes the work performed from July 1989 to Jan. 1995. The work is supported by NASA Co-operative Agreement NCC2-617. This report consists of four parts. The first part is the introduction of the research effort. The second part describes the work and results from July 1989 to June 1993. The third part describes the work and results from July 1993 to January 1995. A summary is given at the end of this report.

## 1 INTRODUCTION

The present supersonic civil airplane, the Concorde, is a technological break-through in aviation history. However, it is an economical disaster for two main reasons. The first is her low aerodynamic performance, that allows only 100 passengers to be carried for a short-range flight with expansive airfare. Another reason is that the shock waves, generated at supersonic cruise, coalesce and form a classical N-wave on the ground, forming a double bang noise termed sonic boom, which is environmentally unacceptable. To enhance the U.S. market share in supersonic civil transport, an airframer's market risk for a low-boom airplane has to be reduced.

Since aircraft configuration plays an important role on aerodynamic performance and sonic boom shape, the configuration of the next generation supersonic civil transport has to be tailored to meet high aerodynamic performance and low sonic boom requirements. Computational fluid dynamics (CFD) can used to design airplanes to meet these dual objectives. The work and results in this report are used to support NASA's High Speed Research Program (HSRP).

In this five years of study and research, CFD tools and techniques have been developed for general usages of sonic boom propagation study and aerodynamic design. In the beginning of the 90's, sonic boom extrapolation technique was still relied on the linear theory developed in the 60's for the nonlinear techniques were computationally expensive. A fast and accurate sonic boom extrapolation methodology (Section 3.2), solving the Euler equations for axisymmetric flow, has brought the sonic boom extrapolation technique up to the 90's standard.

Parallel to the research effort on sonic boom extrapolation, CFD flow solvers have been coupled with a numeric optimization tool to form a design package for aircraft configura-

tion. This CFD optimization package has been applied to configuration design on a low-boom concept (Section 2.3) and an Oblique All-Wing concept (Section 2.4) prior to the wind-tunnel models are built and tested at Ames. The tunnel test results have validated the CFD technique and design tools.

Moving to the world of parallel computing, the aerospace industry needs a numeric optimization tool suitable for parallel computers. A nonlinear unconstrained optimizer for Parallel Virtual Machine has been developed for aerodynamic design and study. Study in Section 3.3 demonstrates the capability of this optimizer on aerodynamic design.

# 2 PREVIOUS WORK/RESULTS

The work and results described in this section was begun in July 1989. The first project was to use CFD tools and existing linear theory to predict waveform signatures at some distances from flight vehicles. The aim of this study was to demonstrate and develop the technique of sonic boom prediction by CFD. The next step was to apply this developed technique to low-boom configurations.

The second project, which was the continuation of the first one, was to develop a CFD optimization package for design process on meeting the dual objectives of high aerodynamic performance and low sonic boom loudness. This optimization package was applied to three different High Speed Civil Transport (HSCT) baseline configurations and a generic body of revolution.

A wind-tunnel model (Ames Model 3) was built based on one of the modified HSCT baseline configuration. This model was tested in June 1993. The test results were used to validate the design method. Publication of the result was limited due to the sensitive nature of the project.

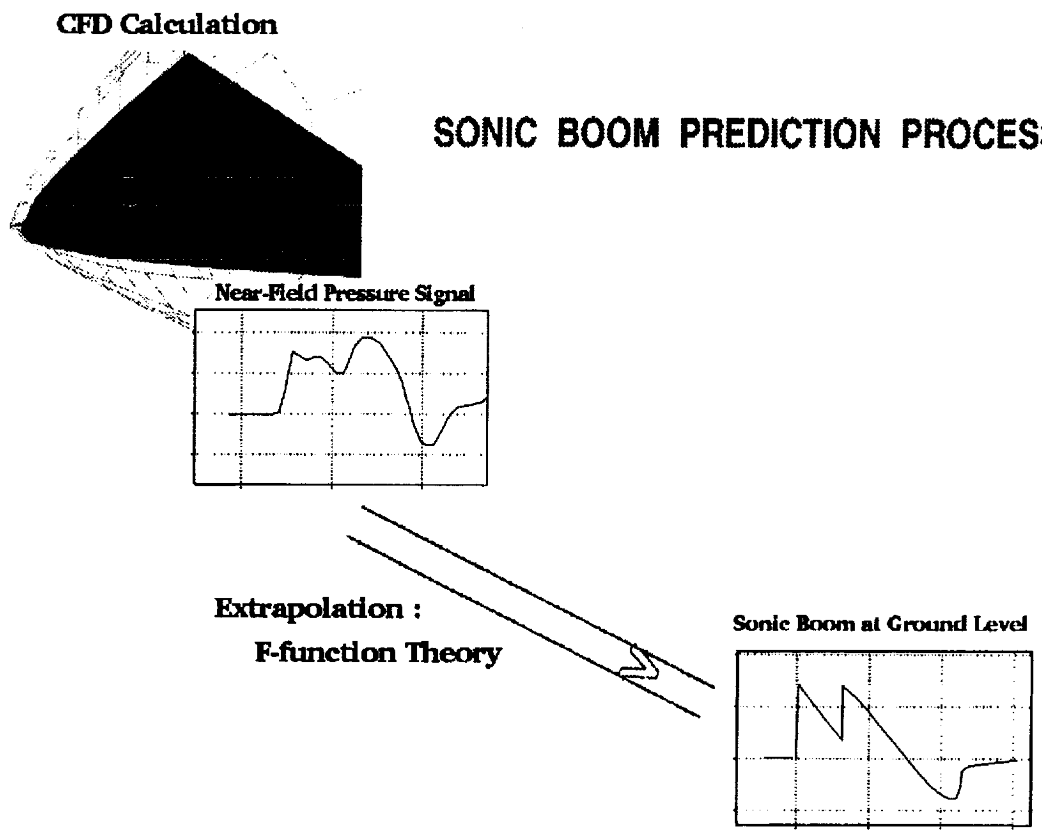
A counterpart of the conventional HSCT concept was the Oblique All-Wing (OAW) concept. CFD computational supports, as well as optimization calculations, were provided to the OAW design team consisting personnels from NASA Ames Research Center, industry, and university. The aim of the project was to design a realistic configuration for wind-tunnel test. The model was built and tested at Ames in June 1994.

## 2.1 Sonic Boom Prediction Technique

In the early stage of sonic boom prediction activity, two major things were involved. The very first thing was to identity the capability of CFD in sonic boom prediction. The second thing was to apply these CFD tools to predict sonic boom signals of varies configurations after necessary code modification, grid refinement study, and comparison with supersonic linear theory.

### 2.1.1 Method Validation

A three-dimensional parabolized Navier-Stokes code, UPS3D,[1] developed at Ames was used as the flow-solver. It is a space-marching code with finite-volume approach. The near field solution of a simple wing/body configuration was calculated by UPS3D, and the overpressure signal at some desired distances were obtained either by the axisymmetric option of UPS3D or a quasi-linear extrapolation code, based on Whitham's F-function theory[2]. Later I realized that using Lighthill integral[3] to calculate the F-function for non-axisymmetric aircraft was more accurate, I wrote a Fortran code, LHF, for sonic boom prediction based on Lighthill integral. This code is available from Ames Software Library. A copy of LHF is attached in Appendix A. The figure below is a brief summary of the sonic boom extrapolation process.




A series of studies on grid refinement, including solution adaptive grid, and on sensitivity of initial distance of extrapolation were conducted. It was found that viscous calculation was unnecessary for sonic boom prediction. However, the grid must be sufficiently fine in the regions of shock and expansion waves. In order to capture all the nonlinear effects in a three-dimensional flow, the near-field overpressure should be captured at about one span length below the flight track before extrapolating to the far field. The detail results were published in AIAA Journal of Aircraft[4] and NASA Technical note[5].

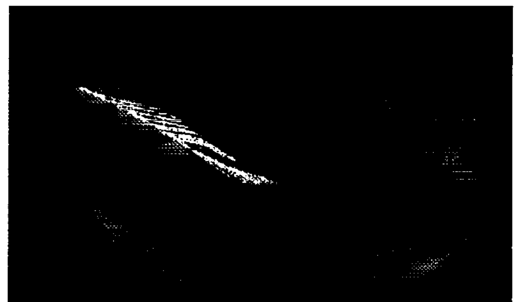
In summary, the tools for sonic boom prediction had been identify and validated in the above study. The combination of CFD and Whitham's method gave a relative efficient tools for sonic boom prediction. Nevertheless, the CFD codes was still computationally expensive for design optimization runs.

### 2.1.2 Boom Prediction for Low-Boom Configurations

With the experience on grid refinement study and the extrapolation procedure, the prediction tools were being used to predict the sonic boom of two low-boom configurations designed by Boeing aircraft company and Langley research center.

Each of the two configurations consisted of two separated parts, namely, the wing and the fuselage. The wing was defined by data in spanwise cuts, whereas the fuselage was defined by data in streamwise cuts. In order to create a single wing-fuselage surface gird for UPS3D code, a grid generator (SAMGRID) was written to defined the wing in streamwise cuts and aggregated the wing to the fuselage. Computation results of the two configurations are shown below.


Langley's Mach 2 Configuration


Boeing's Low-Boom Configuration

The sonic boom signals calculated from the CFD prediction tools were compared to the wind-tunnel data of the Langley's configuration. The computational results of the Boeing's configuration was used to validate the linear design method used by Boeing.

## 2.2 Supersonic Airplane Design

The need for simultaneous sonic boom and aerodynamic optimization was highlighted when it became clear that designed to a strict sonic boom constraint suffered an unacceptable performance penalty. Therefore, low-boom design studies must carefully balance the trade-off between sonic boom loudness and aerodynamic performance. A CFD optimization package was developed to demonstrate the methodology for the optimization of supersonic airplane designs to meet the dual objectives of low sonic boom and high aerodynamic performance.

In this project, an optimizer with linear and nonlinear constraints was first identified, and then an efficient CFD flow solver was chosen. This CFD code had to be sufficiently fast because more than 90% of the computational time were used in CFD calculations. Before this optimization was used to design low-boom wind-tunnel model (Section 2.3), it was tested and exercised by improving aerodynamic performance of a low-boom wing/body configuration and a body of revolution.

ORIGINAL PAGE
COLOR PHOTOGRAPH

### 2.2.1 CFD Optimization Package

Several computational tools interconnect in the optimization procedure are listed below:

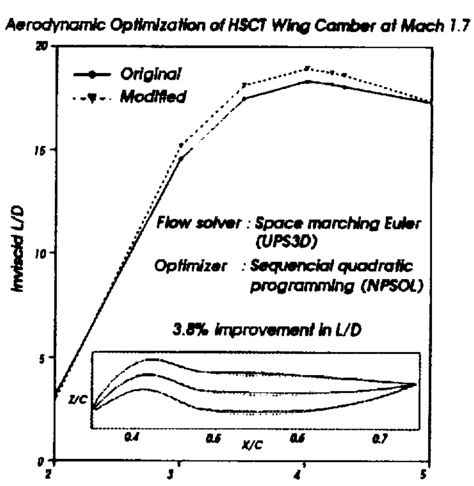
- UPS3D: 3-D parabolized Navier-Stokes code; inviscid calculation only (Ref. 1)
- NPSOL: numerical optimization code[6]; a sequential quadratic programming algorithm in which the search direction is the solution of a quadratic programming subproblem
- HYPGEN: hyperbolic grid generator[7]; a sufficiently fast and robust to operate within an automated optimization environment.
- LHF: sonic boom extrapolation code (Appendix A); a routine based on Whitham's F-function and the equal-area rule[8]
- SAMGRID: wing/body surface grid generator (Appendix B); a sufficiently fast and robust to operate within an automated optimization environment
- DB: sonic boom loudness calculation; a code gives perceived loudness (PLdB) of the sonic boom can be determined by Stevens' Mark VII method[9] which involves Fast Fourier Transform on the energy spectrum of the sonic boom

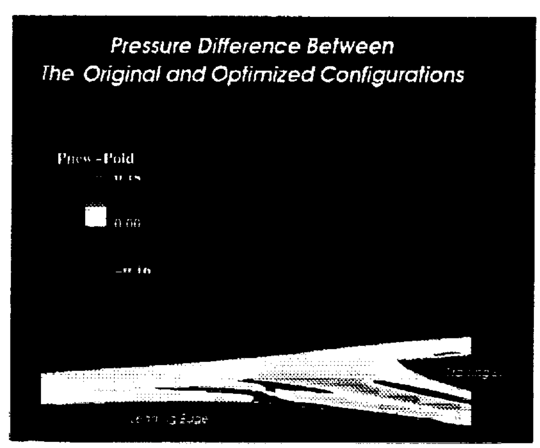
This CFD optimization package is robust and efficient on Cray-YMP. The application of this package will be described in the following sections.

### 2.2.2 Aerodynamic and Sonic Boom Optimization

The optimization design package was exercised using a recently-developed low-boom wing-body configuration, Boeing 1080-991 (also called Haglund model), designed by George Haglund. This optimization technique was applied separately to the two objectives of high aerodynamic performance and low sonic-boom loudness.

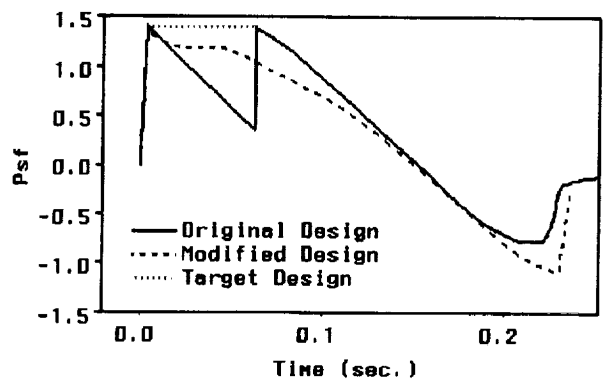
For aerodynamic enhancement, control points are set on the cambers of the wing, with the thickness kept fixed. The left figure below shows the differences on a inboard airfoil section of the original and the modified. The polar plot shows the improvement of L/D of the modified configuration over the original by 3.8%. The right figure below shows that the modified wing had less wave drag than the original one at the leading edge. This means



that the leading thrust is improved by the optimization process. The whole process takes about 4 CPU hours on Cray-YMP.

For sonic boom improvement, F-function was employed as an entity to define the equivalent area distribution and sonic boom shape. The original Haglund model was supposed to give a flat-top pressure waveform at the ground. However, calculations showed that the waveform had an intermediate shock followed right after the bow shock; whereas the flat-top waveform would have no intermediate shock. The design code redistributed the equivalent area of the fuselage (without changing the wings), and re-captured the flat-top characteristic of the pressure waveform. The figure below compares the sonic boom signatures among the original, optimized, and target flat-top. Due to the sensitive nature of the con-




figuration, the change of the configuration will not be shown here. The details of this optimization methodology and results were considered as sensitive materials and were presented in the 2nd Annual Sonic Boom Workshop.[10]

### 2.2.3 Drag Minimization on Haack-Adams Body

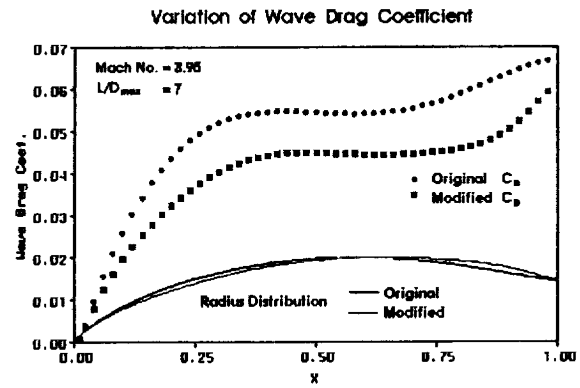
The purpose of this study was threefold:

- to search for a design method to minimize the drag of a supersonic projectile
- to demonstrate the capability of the CFD optimization package described above
- to search for computational grid density effect on optimization performance

The baseline configuration chosen for this study was called Haack-Adams body[11], a body of revolution with a pointed nose and a base of finite area. This body was thought to be the minimum-drag body under the slender body theory. Wind-tunnel data were available for CFD validation. The method of optimization made use of the Fourier Sine expansion, which had three main advantages over the traditional techniques based on shape functions and control points:

- The volume of the body was fixed without putting external constraints. External constraints cost more computational time. For some cases, fixed volume is not feasible.
- Global minimum was search.
- Number of design variables was substantially reduced.

The figure below summaries the result of this study. The nose of the body was trimmed to

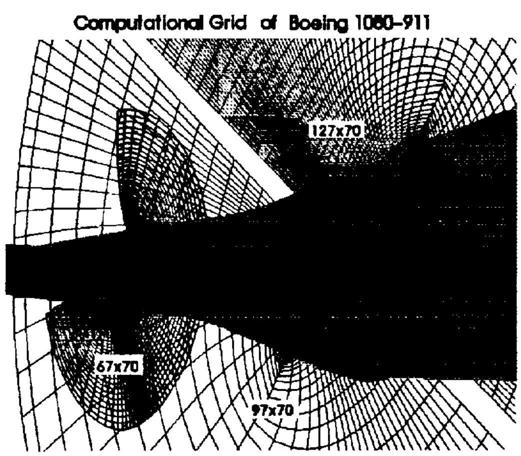


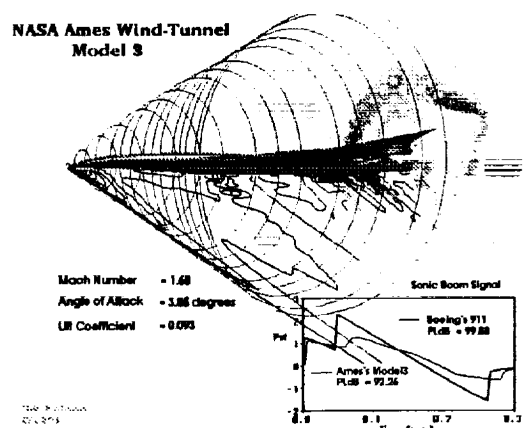

ORIGINAL PAGE
COLOR PHOTOGRAPH

reduce the wave drag. Since the total volume was constrained, volume was added near the end of body. Total wave drag reduction was by 6%. The results were presented in a AIAA meeting[12] and published in Journal of Aircraft Vol. 32, No. 1, Jan/Feb. 1995.

## 2.3 Low-Boom Wind-Tunnel Configuration (Ames Model 3)

Efforts were made to design a new wing/body/nacelle configuration, which had a lower sonic boom relative to the baseline, 1080-911 from Boeing Company, of low boom HSCT concept. The CFD optimization package described in Section 2.2.1 were employed to modify this baseline configuration. The result of the optimization was used to build a wind-tunnel model, Ames Model 3, tested at Ames 9'x7' wind tunnel in June 1993. Due to the sensitive nature of the configuration, no planform shapes will be shown here. However, the left and right figures below show the computational grid and the optimization result, respectively. The plot at the lower right-hand corner of the right figure shows the

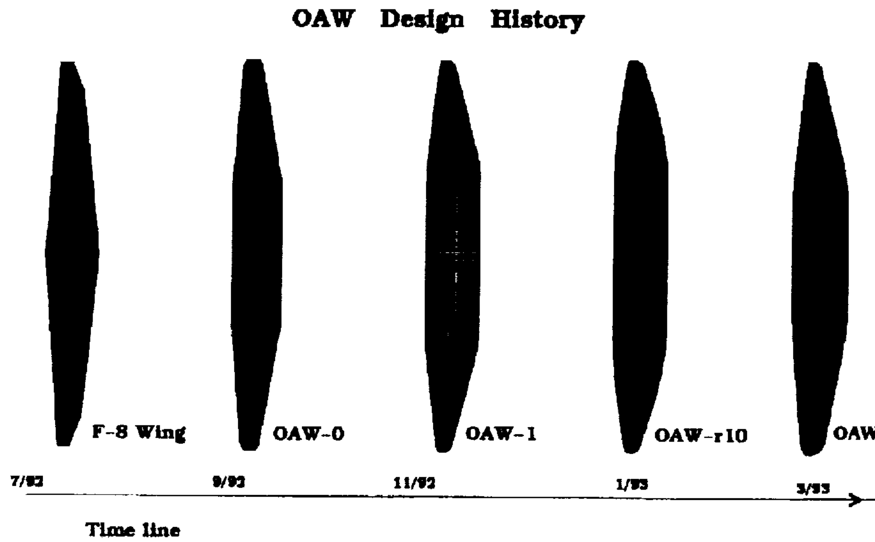


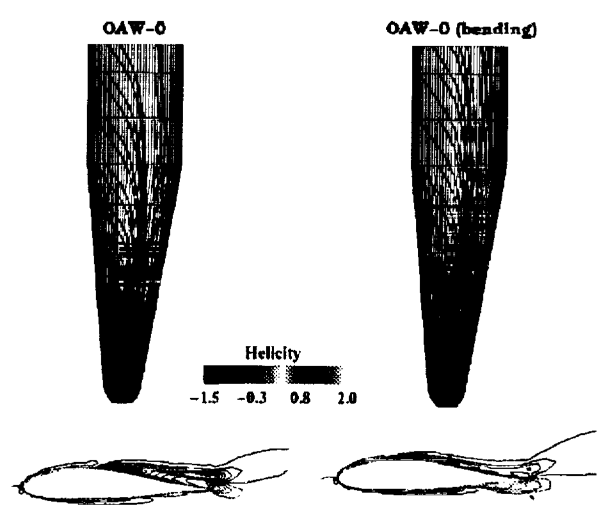

sonic booms of the baseline and Model 3 respectively. The baseline configuration has a loudness level about 100 PLdB; whereas Model 3 has about 92 PLdB. The results of this research were presented in the 3rd Annual Sonic Boom Workshop.[13]

## 2.4 Oblique-All Wing (OAW) Computation and Design

Oblique flying-wing[14] is an alternative supersonic aircraft concept. Ames, Boeing, Douglas, and Stanford University joined and formed a design team in 1992 to investigate the feasibility of OAW for commercial use. The study included aerodynamic performance, stability, structure, landing gear, airplane exits, and airport regulations. The design team decided to build a wind-tunnel model for wind-tunnel testing in June 1994. My job was to provide Navier-Stokes CFD supports and, if possible, optimization results. The figure below shows some of the wings that were analyzed since the beginning of this study.




The flow solver being used was Overflow code, a 3-D Navier-Stokes code using the diagonal with ARC3D algorithm[15]. One of the most challenging works of this project was to reduce the separation on the left wing (trailing wing). The separation on the upper surface of the wing and the corresponding vortices are shown in the left side of the figure below. It




was found that bending of the wing could abate the separation, as well as improve the lift-to-drag ratio. The right side of the figure shows a weaker separation pattern on the ended

wing. Due to the sensitive nature of this study, the results can only be presented in the weekly group meetings at Ames and a controlled distributed NASA Contractor Report.

# 3 CURRENT WORK/RESULTS

Currently, research effort was concentrated on one theme that is sharpening the tools for HSCT design. Three research topics are focused: near-field CFD calculation and sonic boom softening of Boeing Reference-H, improvement of sonic boom extrapolation, and aerodynamic design on parallel computer.

In order to study and design a real complex aircraft, a relatively fast CFD technique has to be developed for optimization environment. Coupling a fast space-marching code and a time iterative code with overset gird concept can take the advantage of marching code at the fuselage/wing region and solve the complex flow field near the wing/nacelle region at the same time.

A very efficient wave propagation code for mid-field sonic boom prediction has been developed based on the method of characteristics. This code solves the Euler equations for 1.2 minutes on Cray-YMP; whereas, the axisymmetric CFD method described in Section 2.1.1 takes 40 minutes on the same computer.

Number crunching problems, like CFD calculations, on parallel machines can be efficiently done in today's computing environment. This may lead to the future of aerodynamic research and design. In order to exercise HSCT design on parallel computers, a nonlinear optimization routine has been developed for a network based parallel computer system in which a cluster of engineering workstations serves as a virtual parallel machine.

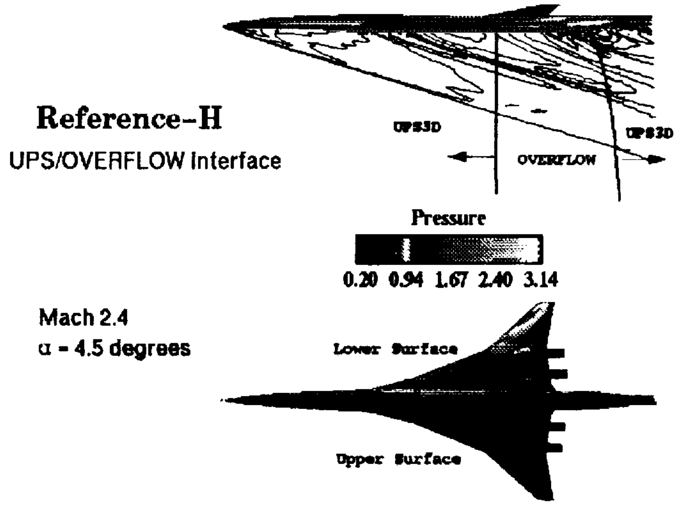
## 3.1 Sonic Boom and Performance Study of Reference-H

Research effort on *low-boom configuration* concept has been invested for the past four years. A new proposed route structure for HSCT's incorporating supersonic corridors over land and water has relaxed the sonic boom constraint somewhat. The objective of this study is twofold. First is to exercise the methodology of combining two different CFD codes to solve the near-field solution of a realistic HSCT configuration in an efficient and accurate manner. Second is to reduce the sonic boom loudness of a *performance configuration* concept, Reference-H, without jeopardizing the aerodynamic performance. The basic components of Reference-H are a fuselage, a pair of swept wings, and four nacelles.

### 3.1.1 Reference-H Near-Field Study

The CFD codes used in this study are the UPS3D code and the OVERFLOW code. Both CFD codes has been described in Section 2.1.1 and 2.4, respectively. The former is an efficient space-marching code. However, it fails in the region where subsonic pocket exists; especially in the region of the wing/nacelle integration. The latter is a time-iterative code with Chimera overset grid concept, which makes the code more viable in solving the

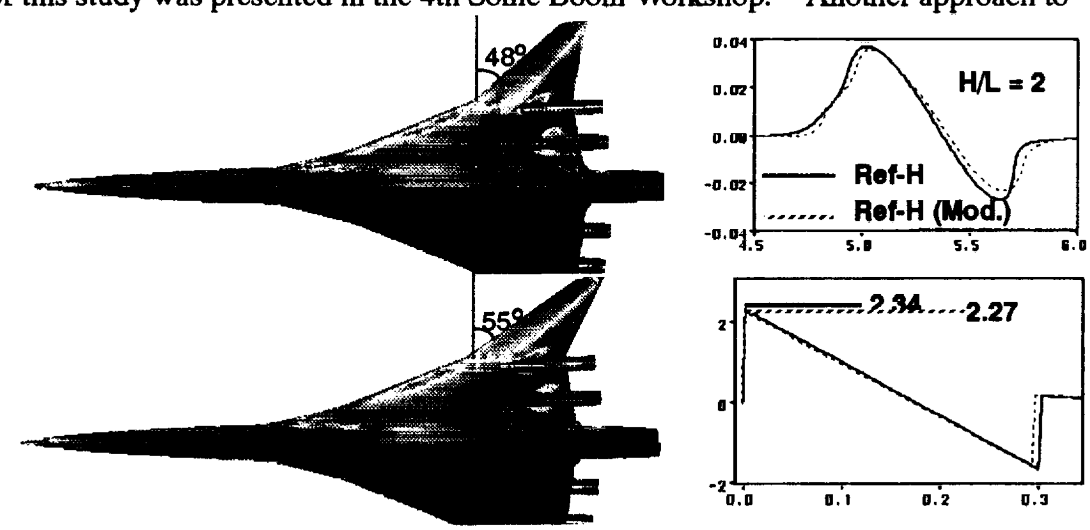
region of wing/nacelle integration. In this study, only inviscid flow is considered. Figure below summarizes the result of the CFD calculations.




The near-field solution is studied for the case of Mach number 2.4 and angle of attack 4.5 degrees. Wind-tunnel data of the Reference-H validate the CFD method. Study shows that flow particles turn significantly over the outer nacelle compared with the inner nacelle. It indicates that the effect of the nacelle orientation might improve the aerodynamic performance.

### 3.1.2 Sonic Boom Softening

The sonic boom of the Reference-H configuration is also obtained. The calculation shows that the boom is an N-wave of 104 PLdB with 2.5 psf. bow shock on the ground. Details of the sonic boom prediction technique can be found in Ref. 10. Boom modification for performance aircraft is very much different from the low-boom aircraft for cruise Mach number and lift are higher. Therefore, the technique developed previously can not be strictly applied to Reference-H. However, changing the equivalent area can be helpful. The result of this study was presented in the 4th Sonic Boom Workshop.[17] Another approach to



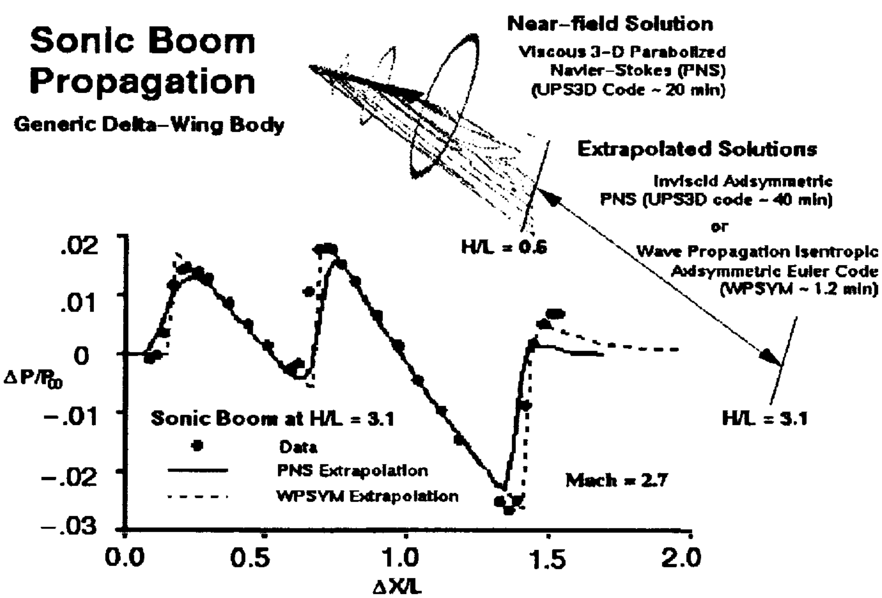

reduce the boom is by experimenting the sweep angle. The figure above show one of the exercises done on the Ref-H. This exercise successfully shows Boeing how much boom

reduction can be achieved by redistributing the lift. An closer on-going technology communication with airframe industry is needed in order to achieve the goal of sonic boom softening on performance aircraft. A team consisting myself and other personnels from Boeing and NASA Langley has been formed to achieve the goal.

## 3.2 Sonic Boom Mid-Field Extrapolation (WPSYM)

In the beginning of 90's, sonic boom extrapolation technique was still relied on the linear theory developed in the 60's for the nonlinear techniques were computationally expensive. Today, a fast and accurate sonic boom extrapolation methodology is needed to bring the sonic boom extrapolation technique up to the 90's standard for HSCT design. The objective of this study is to develop an efficient and accurate higher-order computational method, solving the Euler equations, for supersonic aero-acoustic wave propagation.

An axisymmetric wave propagation code (WPSYM) has been developed for mid-field sonic boom extrapolation. This propagation code has been demonstrated as an efficient and accurate tool over the previous CFD method, described in Section 2.1.1 and Ref. 4, on a generic wing-body configuration. The figure below shows that a 3-D near-field solution

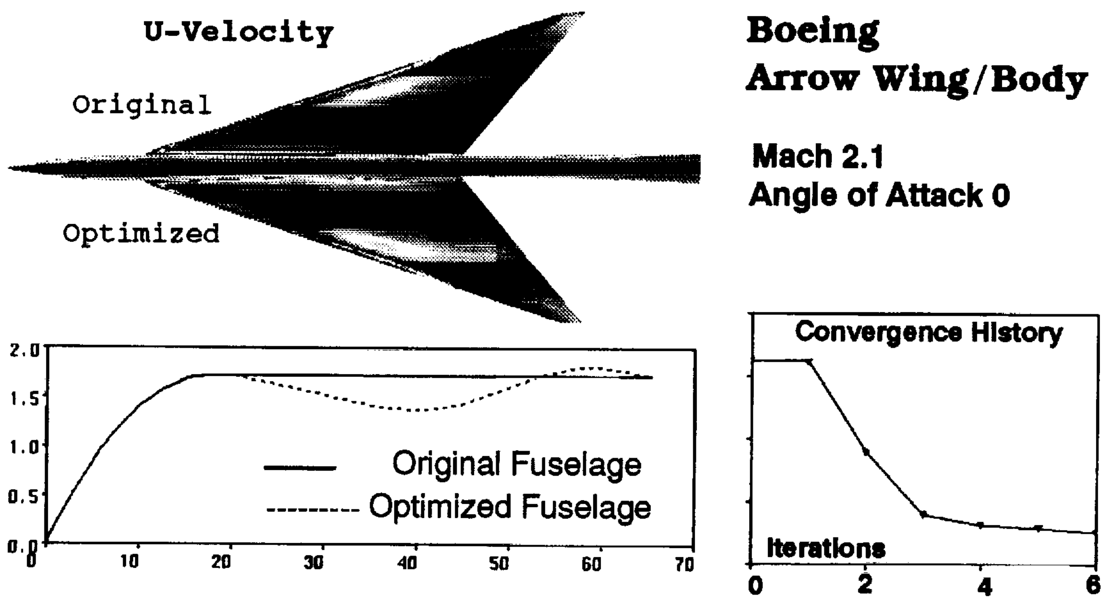



is obtained from UPS3D code; the result is then interfaced to two axisymmetric sonic boom extrapolation codes, namely, the axisymmetric version of UPS3D and the recent wave propagation code (WPSYM). The former takes 40 minutes on Cray-YMP, and the latter takes 1.2 minutes on the same machine. The x-y plot in the figure compares the numerical extrapolation results to wind-tunnel data. The result has been shown in NASA Technical Highlight and the methodology has been presented in the 4th Annual Sonic Boom Workshop at NASA Langley in June 1994.[16]

## 3.3 Optimizer on PVM (*II*OWA)

Moving to the world of parallel computing, the aerospace industry needs a numeric optimization tool in the parallel environment. One of the promising parallel computing concept is the network-based distributed computing. The Parallel Virtual Machine (PVM) is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. PVM allows users to link up engineering workstations to work as a single distributed-memory (parallel) machine. Merritt Smith and I wrote a manual on PVM for beginning users. A copy of the manual is attached in Appendix C.

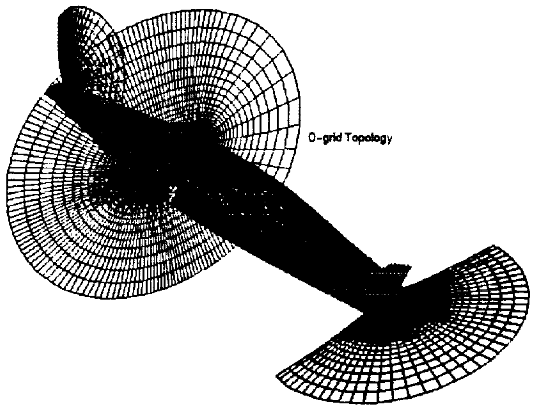
A parallel optimizer based on nonlinear Quasi-Newton method has been developed and coupled with an efficient CFD code for basic aerodynamic design and study. This optimizer is called *II*OWA (parallel Optimizer With Aerodynamics). The figure below is a demonstration of *II*OWA. A Boeing arrow wing/body configuration is chosen in this

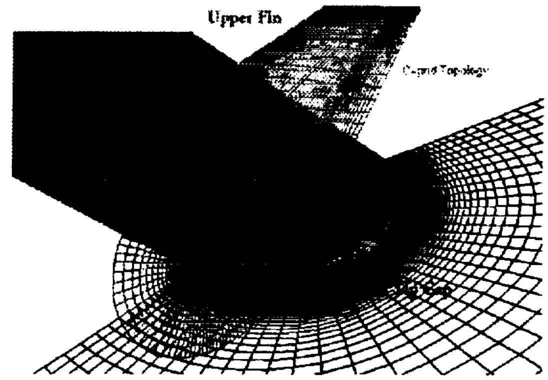



study. The fuselage radius is changed so that the wave drag is minimized. The parallel CFD optimization process takes 24 wall-clock hours on 4 SGI workstations to reduce the wave drag by 6.5%. The optimized result is a "coke bottle" shape fuselage, as expected by supersonic area rule. The convergence history of the optimization process is also shown in the figure. The optimizer is also coupled with a parallel CFD code, MEDUSA, to perform viscous 2-D multizone airfoil optimization supported by overset grid concept. The results will be presented at NASA CAS conference in March 1995.

## 3.4 Oblique All-Wing (OAW): CFD support

The OAW design team has asked for CFD support on the latest configuration OAW-3 from which a wind-tunnel model has been built and tested at Ames in June 1994. The figure below shows the chimera grid topology on the OAW-3 with fin. The design team want to compare the CFD result with the result from pressure sensitive paint (PSP). Therefore,




CFD calculations have to be done prior to the wind-tunnel test because color map from CFD result is need for PSP calibration.

# 4 SUMMARY

The computational tools for sonic boom prediction, aerodynamic calculation, and configuration design of the current HSCT concept have been validated and applied to build wind-tunnel model for further testing and validation. The techniques developed in this five-year research and their applications, such as sonic boom prediction technique (Section 2.1), design of Ames Model 3 (Section 2.3) by CFD optimization (Section 2.2), and sonic boom softening for performance configuration (Section 3.1), have clearly shown support to the HSRP as it moved to its phase two period.

An accurate sonic boom extrapolation tool has always been an issue. It is because the flow phenomena in the atmosphere are nonlinear, but the common technique for extrapolation is linear acoustic theory developed in the 60's. On the other hand, CFD technique is too computationally expensive. Recently, a fast and accurate sonic boom extrapolation methodology (Section 3.2), solving the Euler equations for axisymmetric flow, has brought the sonic boom extrapolation technique up to the 90's standard.

Parallel computing is a fast growing subject in the field of computer science because of the promising speed in number crunching computations. A new optimizer (Section 3.3) for parallel computing concept has been developed and tested for aerodynamic drag minimization. This optimimizer is also coupled with a parallel CFD code so the whole optimization process is parallel. This is a promising method for CFD optimization making use of the computational resources of workstations, which unlike supercomputers spend most of their time idle.

Finally, the OAW concept is so attractive because of its overall performance in theory. In order to fully understand the concept, a wind-tunnel model is built. CFD Navier-Stokes calculations helps to identify the problem of the flow separation (Section 2.4), and also help to design the wing deflection for roll trim and alleviating the flow separation.

# 5 References


1. Lawrence, S., Chaussee, D., and Tannehill, J., "Application of an Upwind Algorithm to the 3-D Parabolized Navier-Stokes Equations," AIAA paper 87-1112, June 1987.
2. Whitham, G. B., "The flow Pattern of a Supersonic Projectile," Comm. Pure & Appl. Math., Vol. V, No. 3, 1952.
3. Lighthill, M. J., "Higher Approximation", General Theory of High Speed Aerodynamics, Vol. VI of High Speed Aerodynamics and Propulsion, Princeton University Press, 1954.
4. Cheung, S., Edwards, T., and Lawrence, S., "Application of CFD to Sonic Boom Near and Mid Flow-Field Prediction," J. of Aircraft, Vol. 29, No. 5, 1992.
5. Cheung, S., Edwards, T., and Lawrence, S., NASA TM 102867, August 1990.
6. Gill, P., Murray, W., Saunders, M., and Wright, M., Users Guide for NPSOL, Technical Report SOL 86-2, Stanford University, CA 94305, 1986.
7. Chan, W., and Steger, J., "A Generalized Scheme for Three-Dimension Hyperbolic Grid Generator," AIAA Paper 91-1588, 1991.
8. Heaslet, M. and Lomax, H., "Supersonic and Transonic Small Perturbation Theory," General Theory of High Speed Aerodynamics, Vol. VI of High Speed Aerodynamics and Propulsion, Princeton University Press, 1954.
9. Stevens, S. S., "Perceived Level of Noise by Mark VII and Decibels (E)," Jour. Acoust. Soc. Am., Vol. 51(2), 1972.
10. Cheung, S. and Edwards, T., "Supersonic Design Optimization Method for Aerodynamic Performance and Low Sonic Boom," High Speed Research: Sonic Boom, Vol. II, NACA CP-3173, 1992.
11. Haack, W., "Projectile Shapes for Smallest Wave Drag," Translation No. A9-T-3, Contract W33-038-ac-15004, ATI No. 27736, Air Material Command, US Air Force, Brown Univ., 1948.
12. Cheung, S., Aaronson, P., and Edwards, T., "CFD Optimization of a Theoretical Minimum Drag Body," AIAA paper 93-3421, August 1993.
13. Cheung, S., "Design Process of Ames Wind-Tunnel Model 3," High Speed Research: Sonic Boom, Vol. II, NACA CP-10133, 1993.
14. Water, M., Ardema, H., and Kroo, I., "Structure and Aerodynamic Considerations for an Oblique All-Wing Aircraft," AIAA paper 92-4220, August 1992.
15. Pulliam, T. and Chaussee, D., "A diagonal Form of an Implicit Approximation Factorization Algorithm," J. of Comp. Physics, Vol. 39, 1981, pp. 347-363.
16. Cheung, S., "Viscous CFD Analysis and Optimization of an Oblique All-Wing Transport," NASA CDCR-20005, November 1994.
17. Cheung, S., "Sonic Boom Softening of Reference-H," High Speed Research: Sonic Boom, NASA Langley, June 1994. (NASA CP will be published).

# Appendix A

**LHF (Fortran Listing)**

LINE # SOURCE TEXT

```
C
      PROGRAM LHF
C
C-----------------------------------------------------------------------
C
C     This program calculates
C     1) the Lighthill F-function on body surface.
C        - a) input data
C        - b) design parameters
C     2) the overpressure signature at given distapce R1.
C     3) the loudness level of the sonic boom at R1
C
C-----------------------------------------------------------------------
C
C     INPUT  -
C     Lhf.in(3)     :Input parameter
C     area.in(2)    :Equivalent area distribution. (INAREA=0)
C     f.dat(4)      :F-function distribution. (INAREA=2)
C     p.r0(8)       :Pressure signature at near field distance R0. (R0>0.)
C     coef.dat(33)  :B-function due to lift. (LIFT=1)
C     grid.in(11)   :Surface grid--PLOT3D planar format. (INAREA=1)
C     Default case  :Wing-body (INAREA=-1)
C
C
C     OUTPUT -
C     area.out(12)  :Equivalent area distribution and its derivative.
C     ffn.out(13)   :F-functions on the body surface and at distance R1.
C     p.out(14)     :Pressure signature at distance R1.
C     icurve_F(34)  :Integral curve of the shifted F-function.
C
C-----------------------------------------------------------------------
C
C     Dr. Samson Cheung       [Tel: (415)-604-4462 ]
C     MCAT Institute
C     NASA Ames Research Center
C     M/S 258-01
C     Moffett Field, CA 94035
C     Date : 1992/3/4      Version 2.1
C
C-----------------------------------------------------------------------
C
      PARAMETER (KMAX=220,LMAX=1,JMAX=351,NMAX=900)
C
      DIMENSION X(KMAX,LMAX,JMAX),Y(KMAX,LMAX,JMAX),Z(KMAX,LMAX,JMAX)
      REAL R(NMAX), S(NMAX), SP(NMAX), TAU(NMAX), FTAU(NMAX)
      COMMON/PAR/ FMACH,PFAC
      LOGICAL WBDY
      WBDY = .FALSE.
C
      OPEN(UNIT=3, FILE='Lhf.in', STATUS='OLD')
C
C     Read the input parameters
      NAMELIST /PARA/ FMACH,PFAC,R0,R1,INAREA,LIFT,TORX
      READ(3,PARA)
      WRITE(6,PARA)

C     Input free-stream Mach number = FMACH
C     If TORX > 0, sonic boom verses time, else verses distance
C
C     Is the surface grid contains the whole configuration, or
C     only half-plane or only quater-plane ?
C      PFAC = 1.      ! Whole plane
C      PFAC = 2.      ! Half-plane
C      PFAC = 4.      ! Quater-plane
C
C     R1 will be the distance where the signature is captured.
C
C     If read in area distribution, INAREA = 0
C        read the grid            , INAREA = 1
C        the wing-body case       , INAREA =-1
C        read in F-function       , INAREA = 2
C        read the B-function      , LIFT   = 1
C        read a signiture at R0   , R0     > 0.0
C

      PI = 4.*ATAN(1.)
      JDIM = JMAX
c      JDIM = 301

C
C     If R0 > 0, we read the pressure signuture at R0 and extrapolate
      IF(R0.GT.0.) GOTO 790

C     Find the area distribution of body configuration (sample case).
      IF(INAREA.LT.0) THEN
       CALL WBODY(JDIM,S,TAU)
c       WBDY = .TRUE.
c       CALL CONE(JDIM,S,TAU)
c       CALL WING(JDIM,S,TAU)
c       CALL SEARS(JDIM,S,TAU)
c       CALL BULLET(JDIM,S,TAU)
       GOTO 270
      ENDIF

C     Read in the given area distribution
      IF(INAREA.EQ.0) THEN
       OPEN(UNIT=2, FILE='area.in')
       DO 50 J=1,NMAX
        READ(2,*,END=75) TAU(J),S(J)
        S(J) = S(J)*PFAC
 50    CONTINUE
 75    CONTINUE
       CLOSE(2)
       JDIM = J-1
C       OPEN(UNIT=2, FILE='area.in')
C       DO 80 J=1,JDIM
C        WRITE(2,*) TAU(J),S(J)
C 80    CONTINUE
       GOTO 270
      ENDIF

C     Read in the F-function or define a F-function by calling FUNC
C     and integrate out the equivalent area by calling EAREA
      IF(INAREA.EQ.2) THEN
c       CALL FUNC(TAU,FTAU,JDIM)
       OPEN(UNIT=4, FILE='f.dat')
       DO 100 J=1,NMAX
        READ(4,*,END=110) TAU(J),FTAU(J)
 100   CONTINUE
 110   CONTINUE
```

LINE # | SOURCE TEXT

```
c          JDIM = J-1
           CALL DISTARC(TAU,FTAU,J-1,TAU,FTAU,JDIM,10.,0)
           CALL EAREA(S,FTAU,TAU,JDIM)
           GOTO 270
         ENDIF

c        Read the PLOT3D surface grid file (Planar format)
c        and find the equivalent area distribution
         OPEN(UNIT=11, FILE='grid.in',FORM='UNFORMATTED')
         READ(11) KDIM,LDIM,JDIM
         DO 200 J=1,JDIM
           READ(11) ((X(K,L,J), K=1,KDIM),L=1,LDIM),
     &              ((Y(K,L,J), K=1,KDIM),L=1,LDIM),
     &              ((Z(K,L,J), K=1,KDIM),L=1,LDIM)
  200    CONTINUE
         CLOSE(11)
c
         CALL EQUAREA(KDIM,LDIM,JDIM,X,Y,Z,KMAX,LMAX,JMAX,S)
c
         DO 220 J=1,JDIM
           S(J) = PFAC*S(J)
           TAU(J)=X(1,1,J)
  220    CONTINUE
c
  270    CONTINUE

c        Obtain the derivative of the area distribution
c
         IF(LIFT.EQ.1)     CALL BFUNC(JDIM,S,TAU)
c
         JDIMM1 = JDIM-1
         DO 300 J=2,JDIMM1
           A1=(TAU(J)-TAU(J+1))/((TAU(J)-TAU(J-1))*(TAU(J+1)-TAU(J-1)))
           A2=(TAU(J)-TAU(J-1))/((TAU(J+1)-TAU(J))*(TAU(J+1)-TAU(J-1)))
           SP(J) = A1*(S(J-1)-S(J))+A2*(S(J+1)-S(J))
  300    CONTINUE
         A2 = (TAU(3)-TAU(1))/((TAU(2)-TAU(1))*(TAU(3)-TAU(2)))
         A3 = (TAU(2)-TAU(1))/((TAU(3)-TAU(2))*(TAU(3)-TAU(1)))
         A1 = A2-A3
         SP(1) = -A1*S(1)+A2*S(2)-A3*S(3)
         A1 = (TAU(JDIM)-TAU(JDIM-2))
     &    / ((TAU(JDIM)-TAU(JDIM-1))*(TAU(JDIM-1)-TAU(JDIM-2)))
         A2 = (TAU(JDIM)-TAU(JDIM-1))
     &    / ((TAU(JDIM-1)-TAU(JDIM-2))*(TAU(JDIM)-TAU(JDIM-2)))
         A0 = A1 - A2
         SP(JDIM) = A2*S(JDIM-2)-A1*S(JDIM-1)+A0*S(JDIM)
c  1st order     SP(1) = (S(2)-S(1))/(TAU(2)-TAU(1))
c  1st order     SP(JDIM) = (S(JDIM)-S(JDIMM1))/(TAU(JDIM)-TAU(JDIMM1))
c

c        Redistribute the S in equal spacing
         CALL DISTARC(TAU,S,JDIM,TAU,S,JDIM,10.,0)
         CALL DISTARC(TAU,SP,JDIM,TAU,SP,JDIM,10.,0)

         DO 340 J=1,JDIM
           R(J) = SQRT(S(J)/PI)
  340    CONTINUE
c
         OPEN(UNIT=12,FILE='area.out')
         WRITE(12,400)
  400    FORMAT(37H#This is equivalent area distribution)
         DO 450 J=1,JDIM
           WRITE(12,580) TAU(J),S(J)
  450    CONTINUE
c          WRITE(12,*)' '
c          WRITE(12,451)
c 451      FORMAT(48H#This is the derivative of the area distribution)
c          DO 455 J=1,JDIM
c            WRITE(12,580) TAU(J),SP(J)
c 455      CONTINUE
         CLOSE(12)

c        The interference between wing and body, sample case only.
         IF(WBDY) CALL WB(JDIM,SP,TAU)

c
c        Obtain the Lighthill F-function
         CALL LIGHT1(TAU,R,SP,JDIM,FMACH,FTAU)

c
c        Write out the Lighthill F-function at the body surface.
         OPEN(UNIT=13,FILE='ffn.out')
         WRITE(13,556)
  556    FORMAT(49H#This is the Lighthill F-function at body surface)
         DO 560 J=1,JDIM
           WRITE(13,580) TAU(J),FTAU(J)
  560    CONTINUE
  580    FORMAT(2X,E16.8,1X,E16.8)
         CLOSE(13)
  790    CONTINUE

c
c        Obtained the pressure signature at distance R1 from the body
         CALL FFN(FTAU,TAU,FMACH,JDIM,R0,R1,TORX)
c
         CLOSE(3)
         STOP
         END
```

LINE # SOURCE TEXT

```
      SUBROUTINE LIGHT1(TAU,R,SP,N,FMACH,FTAU)
      DIMENSION R(N),SP(N),TAU(N),FTAU(N)
C
      PI= 4.*ATAN(1.)
      BETA=SQRT(FMACH**2-1.0)
      TAU(1)=0.
      FTAU(1)=0.
C
      DO 95 M=1,N
 95   FTAU(M)=0.

      DO 100 J=1,N
        DO 102 I=2,N
          IF(ABS(R(I)).LE.1.E-10) THEN
            Z1 = 1.E+10
            F4 = 0.
            GOTO 98
          ENDIF
          AB=2.0/(BETA*R(I))
          AB1=ABS(AB)
          F1=SQRT(AB1)
          F2=SP(I)-SP(I-1)
          F3=F1*F2
          F4=F3/(2.0*PI)
          Z1=(TAU(J)-TAU(I))/(BETA*R(I))
          XLO=-1.0
          IF (Z1.LT.XLO) GO TO 96
          IF (Z1.LT.4.0) GO TO 97
          IF (Z1.GE.4.0) GO TO 98
 96       HZ1=0.
          FTAU(J)=FTAU(J)+HZ1*F4
          GO TO 99
 97       HZ1=.02937*Z1*Z1-.2175*Z1+.7531
          FTAU(J)=FTAU(J)+HZ1*F4
          GO TO 99
 98       BB=1.0/(2.0*Z1)
          BB1=ABS(BB)
          HZ1=SQRT(BB)
          FTAU(J)=FTAU(J)+HZ1*F4
 99     CONTINUE
102     CONTINUE
100   CONTINUE
      RETURN
      END
```

SOURCE TEXT

LINE #

```
      SUBROUTINE EQUAREA(KDIM,LDIM,JDIM,X,Y,Z,KMAX,LMAX,JMAX,S)
C
C     This subroutine fines the cross-section area of a surface grid
C     which has symmetry plane at Y-axis.  For each marching station
C     (for each x) the area is found.  The area is approximated by
C     tropisodial rule.
C
      DIMENSION X(KMAX,LMAX,JMAX),Y(KMAX,LMAX,JMAX),Z(KMAX,LMAX,JMAX)
      REAL S(JDIM)
C
      DO 10 J=1,JDIM
        DAREA = 0.
        DO 5 K=2,KDIM
          H     = Y(K,LDIM,J)-Y(K-1,LDIM,J)
          ADD = Z(K,LDIM,J)+Z(K-1,LDIM,J)
          DAREA = DAREA + 0.5*ADD*H
    5   CONTINUE
C
C       The unwanted base area.
        H = Y(1,LDIM,J) - Y(KDIM,LDIM,J)
        ADD = Z(1,LDIM,J)+Z(KDIM,LDIM,J)
        BASE = ABS(0.5*ADD*H)
C
C       The area surounded by half of the plane
        S(J) = ABS(DAREA)-BASE
   10 CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE FFN(F,X,FMACH,NP,R0,R1,TORX)
c
c     This program uses F-function theory to predict the pressure
c     signature at far field when an initial pressure signature is given
c
      PARAMETER (NMAX=1400)
      DIMENSION F(NP),X(NP),Y(NMAX),P(NMAX)
      DIMENSION YSTP(NMAX)
      DIMENSION DBLVL(3)

      OPEN(UNIT=14,FILE='p.out')
c
c     Input of initial parameters and pressure signal
c
c     FMACH  = Free-stream Mach number
c     R0     = Initial distance from the body (altitude)
c     R1     = Final distance from the body (altitude)
c     NP     = Number of data (NP < NMAX)
c     NMAX   = This should be large enough to resolve the signature
c     TORX   > 0, sonic boom versus time, else versus distance
c
      NAMELIST /XPSCALE/ XSCALE,PSCALE,Ag,Pg,Pa,IRISE,ALT
c
c     Read the input parameters
      READ(3,XPSCALE)
      WRITE(6,XPSCALE)

c     Define the parameters used in the F-function theory
c
      GAMMA = 1.4
      B = SQRT(FMACH**2 - 1.)
      CAP = (GAMMA+1.)*FMACH**4/(SQRT(2.)*B**1.5)
      SRR1 = SQRT(R1)

c
c     If R0 > 0, extrapolate from R0 to R1.  First calculate the F-fn.
      IF(R0.GT.0.) THEN
        OPEN(UNIT=8, FILE='p.r0')
        DO 15 I=1,NMAX
          READ(8,*,END=30)X(I),P(I)
 15     CONTINUE
 30     CONTINUE
        CLOSE(8)
        NP = I-1
        SRR0 = SQRT(R0)
        DO 35 I=1,NP
          F(I) = SQRT(2.*B*R0)*P(I)/(GAMMA*FMACH*FMACH)
          X(I) = X(I) - B*R0 + CAP*SRR0*F(I)
 35     CONTINUE
      ENDIF
c
c     Y is transposed coordinate
c
c      WRITE(13,49)
      DO 50 I=1,NP
        Y(I) = X(I) - CAP*SRR1*F(I)
c         WRITE(13,*) X(I),F(I)
c 49     FORMAT(43H$This is a transposed F-function at surface)
 50   CONTINUE

c     Find the largest and smallest values of Y
c
      YMAX = -1.E+8
      YMIN =  1.E+8
      DO 55 I=1,NP
        IF(Y(I) .GE. YMAX) YMAX=Y(I)
        IF(Y(I) .LE. YMIN) YMIN=Y(I)
 55   CONTINUE

c
c     Print out the integral curve of the shifted F-function
c      CALL INTF(NP,Y,F)

c     Need to march in Y-direction, define the step
c
      YSTP(1) = YMIN
      YDIS = YMAX-YMIN
      DY = YDIS/FLOAT(NMAX-1)
      DO 80 J=2,NMAX
        YSTP(J) = YSTP(J-1) + DY
 80   CONTINUE
c
c     March through the shifted F-function, check area-balance and
c     place the shock.
      CALL MARCH(NMAX,NP,Y,YSTP,F)
c
c     Obtain the solution
c     NOTE: If TORX>0, the sonic boom is in the form (P-Pinf) vs time
c           or it is  in the form (P-Pinf)/Pinf vs distance.
      DO 150 I=1,NP
        P(I) = GAMMA*FMACH*FMACH*F(I)/SQRT(2.*B*R1)
        X(I) = Y(I) + B*R1
 150  CONTINUE

c
c     Make the data points in evenly distributed manner and
c     scale the sonic if desired
      DO 180 I=1,NP
        X(I) = X(I)*XSCALE
        P(I) = P(I)*PSCALE
 180  CONTINUE

c     Atmospheric aspect
c     ALT = Altitude
c     Ag = speed of sound at ground in ft/sec
c     P0 = reference pressure lb/ft^2 = SQRT(Pa*Pg)
c     Pg = pressure at the ground
c     Pa = pressure at flight altitude
      P0 = SQRT(Pg*Pa)
      VEL = FMACH*Ag
      TREF = X(1)/VEL
      IF(TORX.GT.0.) THEN
        DO 260 I=1,NP
          X(I) = X(I)/VEL - TREF
          P(I) = P(I)*P0
 260    CONTINUE

c
c       The signal (DP vs Time) is calculated, use a empirical program to
c       calculate the rise time, and embed the rise time into the signature.
```

LINE # SOURCE TEXT

```
C          Note: Unit used is still the stupid English unit!
           CALL RISETIME(FMACH,P,X,NP,ALT,IRISE)
C
C          Obtain the noise level
           CALL NOISE(DBLVL,X,P,NP)
C
C          Write the dB(PL) value out
           WRITE(14,500)DBLVL(1),DBLVL(2),DBLVL(3)
  500      FORMAT
     &     ('#Noise level ',F10.4,'PLdB',3X,F10.4,'dB(A)',3X,F10.4,'dB(C)')
        ENDIF

C       Write the sonic boom
        WRITE(14,555)R1
  555   FORMAT(28H#The pressure signal at R1= ,F10.4)
        DO 670 I=1,NP
          WRITE(14,700) X(I),P(I)
  670   CONTINUE
  700   FORMAT(3X,E20.8,2X,E16.6)
        CLOSE(14)
C
        RETURN
        END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE INTF(NP,Y,F)
C     This program print out the integral curve of the shifted F-function
      DIMENSION F(NP),Y(NP)
      OPEN(UNIT=34,FILE='icurve_F',FORM='FORMATTED')
C
      SUMF = 0
      WRITE(34,120)
      DO 100 J=2,NP
        DY = Y(J)-Y(J-1)
        SUMF = SUMF + 0.5*DY*(F(J)+F(J-1))
        WRITE(34,130)Y(J),SUMF
 100  CONTINUE
 120  FORMAT(42H# Integral curve of the shifted F-function)
 130  FORMAT(2E16.6)
C
      CLOSE(34)
      RETURN
      END
```

SOURCE TEXT

```
      SUBROUTINE SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,III)
      DIMENSION F(NP),Y(NP)
      DIMENSION YSTP(NMAX)
      DIMENSION INDEX(40),FS(40)
      COMMON/SHOCK/ INSCT

      YEND = Y(NP)
      FIRST = 1.
      DO 500 J=2,NMAX
        YS = YSTP(J)
C
C       Get the points on the curve for integration, start searching from
C       IS to IE
C
        CALL POINT(NP,Y,F,INDEX,FS,YS,INSCT)
C
C       After obtain the integration points, we can integrate and
C       find the area
C
        IF(INSCT.GT.2) THEN
          IF(III.EQ.3) INSCT = 3
          IS=INDEX(1)
          IE=INDEX(INSCT)
          CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLAT2)
        ELSE
C         The tail shock is already formed, leave program
          IF(INSCT.LE.1 .AND. YS.GT.YEND*1.05) RETURN
          FIRST=1.
          GOTO 500
        ENDIF
C
        IF(FIRST.GT.0.) IFLAT1 = IFLAT2
        IF(IFLAT2.EQ.0) RETURN
        FIRST = -1.
C
C       If IFLAT = 0, YS is the point that have area balanced.
C       If IFLAT2 and IFLAT1 are in different sign, i.e.,
C       the correct point should be between I and I-1.
C       Use bisection method to fine the correct point Y(ISTART)
C
        IF(IFLAT1*IFLAT2 .LT. 0.) THEN
          Y1 = YSTP(J-1)
          Y2 = YSTP(J)
          NC = 500
          DO 200 IC=1,NC
            YS = 0.5*(Y2+Y1)
            CALL POINT(NP,Y,F,INDEX,FS,YS,INSCT)
            IF(III.EQ.3) INSCT = 3
            IS=INDEX(1)
            IE=INDEX(INSCT)
            CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLAT0)
            IF(IFLAT0.EQ.0) RETURN
            IF(IFLAT0*IFLAT1 .LT. 0) THEN
              Y2 = YS
              IFLAT2 = IFLAT0
            ELSE
              Y1 = YS
              IFLAT1 = IFLAT0
            ENDIF
 200      CONTINUE
          WRITE(*,*) 'After ',NC,' steps of bisection'
          RETURN
        ELSE
          IFLAT1 = IFLAT2
          GOTO 500
        ENDIF
C
        FIRST = 1.
 500  CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE POINT(NP,Y,F,INDEX,FS,YS,INSCT)
      DIMENSION Y(NP),F(NP)
      DIMENSION INDEX(40),FS(40)
c
c     Find the points FS on the F-function when YS is given
c     INDEX = the index runs from 1 to NP
c     INSCT = # of points being intersect, at least3 point to do integration
c
      INSCT = 0
      IF(YS .LT. Y(1)) THEN
        INSCT = INSCT + 1
        FS(INSCT) = 0.
        INDEX(INSCT) = 1
      ENDIF
c
      DO 100 I=2,NP
         FAC1 = YS - Y(I)
         FAC2 = YS - Y(I-1)
         IF(FAC1*FAC2 .LE. 0.) THEN
           IF(ABS(Y(I)-Y(I-1)).LE.1.E-14) THEN
c              write(*,*)'ZEROOOOOO,YS,Y(I),I'
c              write(*,*)'ZEROOOOOO',YS,Y(I),I
             INSCT = 0
             RETURN
           ENDIF
           INSCT = INSCT + 1
           SL = (F(I)-F(I-1))/(Y(I)-Y(I-1))
           FS(INSCT) = F(I-1)+SL*(YS-Y(I-1))
           INDEX(INSCT) = I
         ENDIF
 100  CONTINUE
c
      IF(YS .GT. Y(NP)) THEN
        INSCT = INSCT + 1
        FS(INSCT) = F(NP)
        Y(NP) = YS
        INDEX(INSCT) = NP
      ENDIF
c
      RETURN
      END
```

SOURCE TEXT

```
      SUBROUTINE AREA(NP,Y,F,YS,FS,IS,IE,IFLAT)
      DIMENSION Y(NP),F(NP),FS(40)
      COMMON/SHOCK/ INSCT
C
C     Fine the integral of F by trapezoidal rule
C     Integrating from I=IS to IE, E1 is area that from YS to Y(IS)
C     and E2 is the area that from Y(IE) to YS.  Thus E2 should be
C     substracted out and E1 should be added in
C
      E1 = 0.5*(Y(IS)-YS)*(F(IS)+FS(1))
      IF(FS(1) .EQ. 0.) E1 = 0.
      E2 = 0.5*(Y(IE)-YS)*(F(IE)+FS(INSCT))
      AREA1 = E1
      IE=IE-1
      DO 10 I = IS,IE
        SLAP = 0.5*(Y(I+1)-Y(I))*(F(I+1)+F(I))
        AREA1 = AREA1 + SLAP
 10   CONTINUE
C
      A = AREA1 - E2
      IF(A.GT.0) IFLAT=1
      IF(A.LT.0) IFLAT=-1
      IF(ABS(A).LT.1.E-7) IFLAT=0
C
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE WBODY(JDIM,S,TAU)
C     This subroutine find the area distribution of
C     the wing-body configuration.
      DIMENSION S(JDIM),TAU(JDIM)
C
      PI=4.*ATAN(1.)
      ANG=21.*PI/180.
      ANG1=80.*PI/180.
      DX=25.52/FLOAT(JDIM-1)
C
      S(1) = 0.
      TAU(1) = 0.
      DO 2 J=2,JDIM
        TAU(J) = TAU(J-1)+DX
        TTT = TAU(J)-7.01
        IF(TTT.GT.0.) TTT=0.
        RR=0.54-0.011*TTT**2
        S(J) = PI*RR*RR
        IF(TAU(J).GT.8.21 .AND. TAU(J).LT.12.25) THEN
          AA = 4.*0.5*0.05*TAN(ANG)*(TAU(J)-8.21)**2
          S(J) = S(J) + AA
        ENDIF
        IF(TAU(J).GT.12.25 .AND. TAU(J).LT.15.77688849) THEN
          H2 = 0.05*(16.29-TAU(J))
          B2 = 2.91*((TAU(J)-12.25)/(15.77688849-12.25))
          B1 = (TAU(J)-8.21)*TAN(ANG)-B2
          H1 = 0.05*B1/TAN(ANG)
          AA = 4.*(0.5*B1*H1+0.5*(H1+H2)*B2)
          S(J) = S(J) + AA
        ENDIF
        IF(TAU(J).GT.15.77688849 .AND. TAU(J).LT.16.29) THEN
          AA = 4.*(0.5*0.05*TAN(ANG1)*(16.29-TAU(J))**2)
          S(J) = S(J) + AA
        ENDIF
        IF(TAU(J).GT.18.93 .AND. TAU(J).LT.17.52) THEN
          SLOP=(0.15-0.54)/(17.93-17.52)
          RRR=0.54+SLOP*(TAU(J)-17.52)
          S(J) = PI*RRR**2
        ENDIF
        IF(TAU(J).GT.17.93) S(J)=PI*0.15*0.15

 2    CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE CONE(JDIM,S,TAU)
C     This subroutine find the area distribution of the cone-cylinder
C     with half-angle 3.24 degree and 8.6 units of length.
      DIMENSION S(JDIM),TAU(JDIM)
C
      PI=4.*ATAN(1.)
      ANG = 3.24*PI/180.
      DX = 16./FLOAT(JDIM-1)
      S(1) = 0.
      TAU(1) = 0.
      DO 2 J=2,JDIM
        TAU(J)=TAU(J-1)+DX
        IF(TAU(J).LE.8.6) THEN
          R = TAU(J)*TAN(ANG)
        ELSE
          R = 8.6*TAN(ANG)
        ENDIF
        S(J) = PI*R*R
 2    CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE SEARS(JDIM,S,TAU)
C     This subroutine find the area distribution of the Sears-Haack body
C     with fineness ratio 23.5
      DIMENSION S(JDIM),TAU(JDIM)
C
C     # of point on body + # of point on sting = JDIM
      JBDY = JDIM*(2./3.)
      JSNG = JDIM - JBDY
C
      PI=4.*ATAN(1.)
      BL = 1.
      TOTL = 1.9* BL
      F  = 23.5
      DTHETA = PI/FLOAT(JBDY-1)
      DX = BL/FLOAT(JBDY-1)
      RMAX = BL/(2.*F)
      S(1) = 0.
      TAU(1) = 0.
C
C     Constants of Sears-Adams body
      write(*,*)'Input Abase/Amax'
      read(*,*) AR
      write(*,*)'Input XMAX'
      read(*,*) AA
      XMAX = AA*BL
      CONST = AR/PI
      C1 = 1./(2.*(2.*XMAX/BL - 1.))
C
      DO 2 J=2,JBDY
c        THETA = PI-DTHETA*FLOAT(J-1)
c        TAU(J) = (1.+COS(THETA))*BL/2.
        TAU(J) = FLOAT(J-1)*DX
        THETA = ACOS(2.*TAU(J)/BL - 1.)
C       Sears-Haack body
c        POS = (SIN(THETA))**3
C       Haack-Adams body
        POS = CONST*( PI-THETA+0.5*SIN(2.*THETA) +
     &                (4./3.)*C1*(SIN(THETA))**3 )

        R = RMAX*SQRT( ABS(POS) )
        S(J) = PI*R*R
 2    CONTINUE
C
C     Add a sting
      DX = (TOTL-BL)/FLOAT(JSNG)
      DO 5 J=JBDY+1,JDIM
        TAU(J) = TAU(J-1) + DX
        S(J) = S(J-1)
 5    CONTINUE
      RETURN
      END
```

SOURCE TEXT

LINE #

```
      SUBROUTINE BULLET(JDIM,S,TAU)
C     This subroutine find the area distribution of a bullet with a form
C     R = AX^gama
      DIMENSION S(JDIM),TAU(JDIM)
C
      PI=4.*ATAN(1.)
      BL = 4.
      GAMA = 0.65
      RBASE = 0.25
      A = RBASE/(BL**GAMA)
      TOTLEN = BL + 2.*BL
      DX = TOTLEN/FLOAT(JDIM-1)
      S(1) = 0.
      TAU(1) = 0.
      DO 2 J=2,JDIM
        TAU(J)=TAU(J-1)+DX
        IF(TAU(J).GE.BL) THEN
          R = RBASE
        ELSE
          R = A*TAU(J)**GAMA
        ENDIF
        S(J) = PI*R*R
 2    CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE WING(JDIM,S,TAU)
C     This subroutine find the area distribution of the low-aspect-ratio wing
      DIMENSION S(JDIM),TAU(JDIM)
C
      PI=4.*ATAN(1.)
      STING = 0.
      DX = 3./FLOAT(JDIM-1)
      S(1) = 0.
      TAU(1) = 0.
      DO 2 J=2,JDIM
        TAU(J)=TAU(J-1)+DX
        IF(TAU(J).LT.2.) THEN
          Z = (PI/12.5)*(TAU(J)-0.5*TAU(J)*TAU(J))
          S(J) = Z
          IF(TAU(J).GT.1.70897) THEN
             STING=PI*0.0625*0.0625
             S(J) = Z + STING - Z*0.125
          ENDIF
        ELSE
          STING=PI*0.0625*0.0625
          S(J) = STING
        ENDIF
 2    CONTINUE
      RETURN
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE BFUNC(JDIM,S,TAU)
C     This subroutine obtains the B-function from fort.10 and add it
C     into the equalivant area.
      PARAMETER (NMAX=800)
      DIMENSION S(JDIM),TAU(JDIM),B(NMAX),X(NMAX)
      COMMON/PAR/ FMACH,PFAC
C
      OPEN(UNIT=33,FILE='coef.dat',FORM='FORMATTED')
C
      READ(33,12)
      READ(33,12)
      READ(33,12)
      READ(33,12)
      READ(33,12)
      DO 10 I=1,NMAX
c       READ(33,15,END=17) X(I),CL,CD,SLOD,B(I),CM
       READ(33,*,END=17) X(I),B(I)
       B(I) = B(I)*PFAC
   10 CONTINUE
   11 CONTINUE
   12 FORMAT(1X)
   15 FORMAT(6E13.5)
   17 CONTINUE
      CLOSE(33)
      NPOINT = I-1
      OPEN(UNIT=33, FILE='bfn.dat')
      DO 20 I=1,NPOINT
       WRITE(33,*) X(I),B(I)
 20   CONTINUE
C
      ISTART=1
      DO 50 J=1,JDIM
        DO 30 I=ISTART,NPOINT
          IF(ABS(X(I)-TAU(J)).LE.1.E-10) THEN
            S(J)=S(J)+B(I)
            ISTART=I
            GOTO 40
          ENDIF
          IF(X(I).GT.TAU(J)) THEN
            IF(I.EQ.1) THEN
              BF=0.
              XF=0.
            ELSE
              BF=B(I-1)
              XF=X(I-1)
            ENDIF
            SLOPE=(B(I)-BF)/(X(I)-XF)
            BT = B(I) + SLOPE*(TAU(J)-X(I))
            S(J) = S(J) + BT
            ISTART=I-1
            IF(I.EQ.1) ISTART=1
            GOTO 40
          ELSE
            IF(I.LT.NPOINT) GOTO 30
            S(J) =S(J) + B(NPOINT)
            ISTART=I
            GOTO 40
          ENDIF
   30   CONTINUE
   40   CONTINUE
   50 CONTINUE
      RETURN
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE WB(JDIM,SP,TAU)
C     This subroutine obtains the wing-body interferencw correction
C     and add it into the derivative of equalivant area.
C     This is a test for wing-body case
      DIMENSION SP(JDIM),TAU(JDIM)
      COMMON/PAR/ FMACH,PFAC
C
      DO 10 J=1,JDIM
         IF(TAU(J).GE.8.21 .AND. TAU(J).LE.12.25)
     &   SP(J)=SP(J)+4.*.05*.54
         IF(TAU(J).GT.12.25 .AND. TAU(J).LE.16.29)
     &   SP(J)=SP(J)-4.*.05*.54
 10   CONTINUE
      RETURN
      END
```

SOURCE TEXT

```
      SUBROUTINE FUNC(TAU,FTAU,JDIM)
      DIMENSION TAU(JDIM),FTAU(JDIM)
      NAMELIST /FFUNC/ YF,ELAM,C,H,B,D,E,BL,YR,DEL
C
C     Read the input parameters
      READ(3,FFUNC)
      WRITE(6,FFUNC)

      TAU(1)=0.
      FTAU(1)=0.
      DY=YR/FLOAT(JDIM-1)
      DO 10 J=2,JDIM
        TAU(J)=TAU(J-1)+DY
        IF(YF.EQ.0.) GOTO 6
        IF(TAU(J).LE.YF/2.) FTAU(J)=2.*TAU(J)*H/YF
        IF(TAU(J).GE.YF/2.0 .AND. TAU(J).LE.YF)
     &  FTAU(J)=C*(2*TAU(J)/YF-1.) - H*(2.*TAU(J)/YF-2.)
C
c 6      IF(TAU(J).GE.YF .AND. TAU(J).LE.ELAM)
c    &   FTAU(J)=B*(TAU(J)-YF)+C
 6      IF(TAU(J).GE.YF .AND. TAU(J).LE.DEL) FTAU(J)=C
        IF(TAU(J).GE.DEL .AND. TAU(J).LE.ELAM)
     &  FTAU(J)=B*(TAU(J)-DEL)+C
C
        IF(TAU(J).GE.ELAM .AND. TAU(J).LE.BL)
     &  FTAU(J)=B*(TAU(J)-ELAM)+(ELAM*B-D)
        IF(TAU(J).GE.BL)
c    &   FTAU(J)=(ELAM*B-D+B*(BL-ELAM))*(TAU(J)-YR)/(BL-YR)
     &   FTAU(J)=-E/(TAU(J)-(BL-ABS(BL-ELAM)/10.))
 10   CONTINUE
C
      CALL DISTARC(TAU,FTAU,JDIM,TAU,FTAU,JDIM,10.,0)

      WRITE(13,75)
      DO 20 J=1,JDIM
        WRITE(13,80) TAU(J),FTAU(J)
 20   CONTINUE
 75   FORMAT(22H#F-function from input)
 80   FORMAT(2X,F8.4,1X,E16.8)
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE EAREA(S,FTAU,TAU,JDIM)
      DIMENSION S(JDIM),TAU(JDIM),FTAU(JDIM)
      DIMENSION F(900)
c
c     Obtain the equivalent area from F-function via Abel Transform
c
c              _x      _y
c     A(x) =  |   (   |  F(t)/SQRT(y-t) dt )  dy
c            _|0     _|0
c
      S(1) = 0.
      TAU(1)=0.
      DO 10 J=2,JDIM
       SS = 0.
       DO 7 I=1,J-1
        DY=TAU(I+1)-TAU(I)
        FINGRL = 0.
        DO 5 K=1,I-1
          DT = TAU(K+1)-TAU(K)
          FINGRL = FINGRL + DT*FTAU(K)/SQRT(TAU(I)-TAU(K))
 5      CONTINUE
        SS = SS + 2.* FINGRL*DY
c         SS = SS + 4.*SQRT(TAU(J)-TAU(I))*FTAU(I)*DY
 7     CONTINUE
       S(J) = SS
 10   CONTINUE
      TANN=S(3)/TAU(3)
      S(2)=TANN*TAU(2)
      WRITE(12,15)
 15   FORMAT(27H#Area from given F-function)
      DO 20 J=1,JDIM
        WRITE(12,80) TAU(J),S(J)
 20   CONTINUE
 80   FORMAT(2X,F8.4,1X,E16.8)
      RETURN
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE RISETIME(FMACH,P,T,NP,ALT,IRISE)
C     An empirical method to calculate the rise time of a sonic boom
C     Rise time derived from regression analysis of Air Force sonic boom
C     flight test data.  Good for N-wave type of signal, may be somewhat
C     conservative (shorter rise time).
C     All unit used are English unit !!!
C
C     FMACH = Free-stream Mach number
C     P(T)  = sonic boom
C     PSH   = Shock strength*
C     ALT   = Altitude (ft)
C     P0    = Free-stream pressure (lb/ft2)
C     RT    = Rise time (sec)
C     TEMP  = Temperature R=F+459.67=(9/5)K
      DIMENSION P(NP),T(NP)
      P0 = 2116.2
      TEMP = 518.69

      IKOUNT = 0
 12   CONTINUE
      IKOUNT = IKOUNT + 1
C
C     Find out the shock strength
      PSH = 0.
      ISH0 = 0.
      DO 30 I=1,NP
        IF(T(I).EQ.T(I+1)) THEN
          IF(ISH0.EQ.0) ISH0=I
          PSH = ABS(P(I+1)-P(ISH0))
        ELSE
          IF(PSH.EQ.0.) THEN
            GOTO 30
          ELSE
            GOTO 40
          ENDIF
        ENDIF
 30   CONTINUE
 40   CONTINUE
      ISH = I
      IF(PSH.EQ.0.) RETURN
C
      IF(IRISE.EQ.1) THEN
C       Now calculate the rise time using Air Force data base
        Y1 = 2.92*FMACH - 7.38
        Y2 = Y1 + ((8.5*FMACH**2 - 45.9*FMACH + 62.9))**(.5)
        AK1 = Y2 * 1000.
        AK = AK1 / (((ALT/1000.) - 5.)*2117.)
        VIS = 100. + 0.5*(TEMP-410.)
        RT = (AK*VIS)*P0/(PSH*TEMP)
        RT = RT/1000.
      ELSEIF(IRISE.EQ.2) THEN
C       Now calculate the rise time assumming 1psf has 3msec rise time
        RT = 0.003/PSH
      ENDIF
C
      WRITE(14,80)RT
 80   FORMAT(38H# Rise time (sec) of the bow shock is,F10.5)
C
C     Originally, T(ISH0)=T(ISH) with infinite shock strength, now create a
C     signal with the rise time, between the index ISH0 to ISH
C     Also extend the signal by the amount of rise time.
C
      DRT = RT/FLOAT(ISH-ISH0)
      DO 200 I=ISH0,ISH-1
        T(I+1) = T(I) + DRT
 200  CONTINUE
      DO 300 I=ISH+1,NP
        T(I) = T(I) + RT
 300  CONTINUE
C
      IF(IKOUNT.LT.10) GOTO 12
      RETURN
      END
```

LINE # SOURCE TEXT

```
*************************************************************************
      SUBROUTINE DISTARC(X,Y,N,XNEW,YNEW,NNEW,FAC,IFLAT)
c
      DIMENSION X(N),Y(N),XNEW(NNEW),YNEW(NNEW)
c
c     This program redistribute the points (X,Y) by subroutine DISTRI
c     based on the arc length.  FAC is the first grid spacing. Note that
c     the end points of the two sets are the same.
c     IFLAT=0, grid points will custer near the first point, =1 near the end.
c     Input  array is (X(i),Y(i)), i=1,...,N,
c     Output array is (XNEW(i),YNEW(i)), i=1,...,NNEW
c
      PARAMETER (MAX=2000)
      DIMENSION S(MAX),TOTARC(MAX),XN(MAX),YN(MAX)
c
c     Maximum number of points allowed is MAX
      IF(MAX.LE.N .OR. MAX.LE.NNEW) THEN
        WRITE(*,*) 'SUB DISTARC : MAX is less than N or NNEW'
        STOP
      ENDIF
c
c     Look for total arc length
      TOTARC(1) = 0.
      DO 10 K=2,N
        ARC = SQRT( (X(K)-X(K-1))**2 + (Y(K)-Y(K-1))**2 )
        TOTARC(K) = TOTARC(K-1) + ARC
 10   CONTINUE
c
c     Apply subroutine DISTRI to obtain the stratching function S
      IF(FAC.LT.1.) THEN
        DELT=FAC*(TOTARC(N)/FLOAT(NNEW-1))
        CALL DISTRI(DELT,NNEW,S,IFLAT)
      ELSE
        S(1) = 0.
        DO 25 K=2,NNEW
          S(K) = S(K-1) + 1./FLOAT(NNEW-1)
 25     CONTINUE
      ENDIF
c
c     Redistribution, put new array in a temperary arrays XN and YN
      XN(1)=X(1)
      YN(1)=Y(1)
      XN(NNEW)=X(N)
      YN(NNEW)=Y(N)
c
      DO 60 J = 2,NNEW
        ARCNEW =  S(J)*TOTARC(N)
        DO 55 K = 2,N
          IF(TOTARC(K).EQ.ARCNEW) THEN
            XN(J) = X(K)
            YN(J) = Y(K)
            GOTO 60
          ENDIF
          IF(TOTARC(K).GT.ARCNEW) THEN
            X1 = X(K-1)
            X2 = X(K)
            Y1 = Y(K-1)
            Y2 = Y(K)
            XX = X1 + (X(K)-X(K-1))*
     &           (ARCNEW-TOTARC(K-1))/(TOTARC(K)-TOTARC(K-1))
            CALL LININT(X1,X2,Y1,Y2,XX,YY)
            XN(J) = XX
            YN(J) = YY
            GOTO 60
          ENDIF
 55     CONTINUE
 60   CONTINUE
c
c     Write the temperary arrays into the output XNEW, YNEW
      DO 70 J=1,NNEW
        XNEW(J) = XN(J)
        YNEW(J) = YN(J)
 70   CONTINUE
      RETURN
      END
```

SOURCE TEXT

LINE #

```
************************************************************************
      SUBROUTINE DISTRI(FANG,KFCS,S,IFINE)
      PARAMETER (MAX=500)
      DIMENSION S(MAX),DUM(MAX)
C
C.....Calculating the stretching function S when given
C      the first spacing, FANG, and the number of points KFCS
C.....If IFINE=1, distribution is cusltering at outer grid
C
      IF(MAX.LE.KFCS) THEN
        WRITE(*,*)'SUB DISTRI : MAX is less than KFCS'
        STOP
      ENDIF

      IF(KFCS.EQ.1) THEN
        S(1) = 0.
        GOTO 40
      ENDIF
C
      DZ1 = FANG
      KFM = KFCS-1
      DZETA = 1./FLOAT(KFM)
      RDBETA = 1.5
      CALL GRBET(DZ1,KFM,0.0001,100,RDBETA)
      CALL FZ1(KFCS,RDBETA,DZETA,S)
C
      IF (IFINE.EQ.1) THEN
          DO 37 K=1,KFCS
            DUM(KFCS-K+1) = S(K)
   37     CONTINUE
          DO 38 K=1,KFCS
            S(K) = 1.-DUM(K)
   38     CONTINUE
      ENDIF
   40 CONTINUE
      RETURN
      END
```

LINE #

SOURCE TEXT

```
*************************************************************************
      SUBROUTINE FZ1(L1,TBETA,DET,Z)
C
C     COMPUTES NORMALIZED NORMAL DISTANCE, Z(L)
C
      DIMENSION Z(250)
      IF(TBETA.EQ.1.) THEN
      DO 10 L=1,L1
      Z(L)=0.
   10 CONTINUE
      ELSE
      DO 20 L=1,L1
      ETA=(L-1)*DET
      RR=(TBETA+1.)/(TBETA-1.)
      EEE=1.-ETA
      RBETA=RR**EEE
      Z(L)=(TBETA-1.)*(RR-RBETA)/(RBETA+1.)
   20 CONTINUE
      END IF
      RETURN
      END
```

SOURCE TEXT

```
      SUBROUTINE GRBET(DFM,NPT,FPCC,ICC,BETA)
C
C     BISECTION METHOD USED TO DETERMINE STRETCHING PARAMETER, BETA,
C     WHICH GIVES DESIRED dY AT THE WALL
C
      DIMENSION Z(250)
      ICCL=ICC
      FPCCL=FPCC*DFM
      BETA1=BETA
      Z1=DFM
      DET=1./NPT
      BR=1.
      FR=-Z1
      IICC=ICC/10
      DO 10 I=1,IICC
      BF=BETA1
      BETA=0.5*(BETA1+1.)
      CALL FZ1(2,BF,DET,Z)
      FF=Z(2)-Z1
      IF(FF.GT.0.) GO TO 15
      BETA1=2.*BETA1-1.
   10 CONTINUE
   15 CONTINUE
      DO 5 NIT=1,ICCL
      CALL FZ1(2,BETA,DET,Z)
      F=Z(2)-Z1
      IF(F.GT.0.) THEN
      FF=F
      BF=BETA
      ELSE
      FR=F
      BR=BETA
      END IF
      BETA=0.5*(BF+BR)
      IF(ABS(F).LT.FPCCL) GO TO 4
    5 CONTINUE
      WRITE(6,100) BETA,F
  100 FORMAT(1H0,36H EXCEEDED MAX. NO. OF ITS....BETA,F ,3G13.6)
    4 CONTINUE
C       CALL FZ1(2,BETA,DET,Z)
C       F=Z(2)-Z1
      BM1=BETA-1.
      BFM1=BF-1.
      BRM1=BR-1.
      RETURN
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE LININT(X1,X2,Y1,Y2,XLOCAL,YLOCAL)
C     This subroutine linearly interpolate YLOCAL when given (X1,Y1) & (X2,Y2)
      IF(X1.EQ.X2) THEN
        YLOCAL=(Y2-Y1)/2.
        GOTO 100
      ENDIF
      SLOPE = (Y2-Y1)/(X2-X1)
      YLOCAL = SLOPE*(XLOCAL-X2) + Y2
100   CONTINUE
      RETURN
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE MARCH(NMAX,NP,Y,YSTP,F)
      DIMENSION F(NP),Y(NP)
      DIMENSION YSTP(NMAX)
      DIMENSION INDEX(40),FS(40)
      COMMON/SHOCK/ INSCT
c
c     This subroutine marches the Y direction and
c     check if the areas are balanced and then place the shock
c
      KOUNT = 0
      YEND = Y(NP)
  100 CONTINUE
      DO IND=1,40
        INDEX(IND)=0.
        INSCT = 0
      ENDDO
      CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,0)
c
c     For tail shock, no need to check the possible position of shock
      IF(YS.GT.0.7*(YSTP(NMAX)-YSTP(1)))  GOTO 400
c
c     Only one possible location of shock
      IF(INSCT.EQ.3) GOTO 400
c
c     More than one possible locations of shock
      IF(INSCT.GE.5) THEN
        YSHK = YS
        CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,3)
        IF(YSHK.LT.YS) THEN
c         The wing shock overcomes
          CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,0)
          GOTO 400
        ELSE
c         There are two separated shocks
c         for the shock is actually locate on the turning edge of F-function
c         we need to relocate it
c
c         Fix the Y1 and Y2 of this small region
          IF(YS .GT. Y(INDEX(INSCT)+1)) THEN
            Y2 = YS
            BIG = 0.
            DO ITEST=INDEX(INSCT)+1,NP
              IF(YS.GT.Y(ITEST) .AND. ABS(YS-Y(ITEST)).GT.BIG) THEN
                Y1 = Y(ITEST)
                BIG = ABS(YS-Y(ITEST))
              ENDIF
              IF(Y(ITEST).GE.YS) GOTO 300
            ENDDO
  300       CONTINUE
c
c           Find YS by bisecting Y1 and Y2
            NC = 500
            DO 320 IC=1,NC
              YS = 0.5*(Y2+Y1)
              CALL POINT(NP,Y,F,INDEX,FS,YS,INSCT)
              DO II=INSCT,1,-1
                IF(INDEX(II).LE.ITEST) THEN
                  INSCT = II
                  GOTO 310
                ENDIF
              ENDDO
  310         CONTINUE
              IS=INDEX(1)
              IE=INDEX(INSCT)
              CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLATO)
              IF(IFLATO.EQ.0) GOTO 400
              IF(IFLATO .GT. 0) THEN
                Y2 = YS
              ELSE
                Y1 = YS
              ENDIF
  320       CONTINUE
            WRITE(*,*) 'After ',NC,' steps of bisection'
          ENDIF
          GOTO 400
        ENDIF
      ENDIF
c
c
  400 CONTINUE
c
c     Form the shock
c
      IF(INSCT.LE.1 .AND. YS.GE.YEND) RETURN
      FDIS = FS(INSCT)-FS(1)
      if(FLOAT(INDEX(INSCT)-INDEX(1)).eq.0.0) then
        WRITE(15,*) 'SDT:  ZERO DIVISION ABOUT TO HAPPEN in MARCH'
        df = 1.e32
      else
        DF = FDIS/FLOAT(INDEX(INSCT)-INDEX(1))
      endif
      F(INDEX(1)) = FS(1)
      Y(INDEX(1)) = YS
      IS = INDEX(1) + 1
      DO 450 I = IS,INDEX(INSCT)
          Y(I) = YS
          F(I) = F(I-1) + DF
  450 CONTINUE
      IF(KOUNT.EQ.20) THEN
        WRITE(*,*)'KOUNT=20!!'
        RETURN
      ELSE
        KOUNT = KOUNT+1
        GOTO 100
      ENDIF
      END
```

LINE # | SOURCE TEXT

```
      SUBROUTINE SIMPSON(X,F,M,X0,X1,SUM)
      DIMENSION X(M),F(M)
C
C     M is odd
      N = M/2
      SUM = 0.
      DO 10 I=1,N
        ODD =  ODD + 2.*F(2*I+1)
        EVEN= EVEN + 4.*F(2*I)
 10   CONTINUE
C
      SUN = F(1) + ODD + EVEN + F(M)
      SUM = SUM*(X1-X0)/(6.*FLOAT(N))

      RETURN
      END
```

# Appendix B

**SAMGRID (Fortran Listing)**

LINE # SOURCE TEXT

```
      PROGRAM SAMGRID
      include "sgrid.com"
c
c     Dr. Samson Cheung
c     Date: Dec., 1993
c     This subroutine reads a surface grid in airfoils
c     sections and reformats it to produce
c     a surface grid of axisymmetric cross-sections.
c
c     Date: Dec., 1993  Version 3.0
c
c     read input geometry
c
c--------------------------------------------------------------------
      OPEN(UNIT=10,FILE='WBGRID.IN',STATUS='OLD',FORM='FORMATTED')
c      OPEN(UNIT=30,FILE='NACGRID.IN',STATUS='OLD',FORM='FORMATTED')
      OPEN(UNIT=40,FILE='XZSIZE.IN',STATUS='OLD',FORM='FORMATTED')
c--------------------------------------------------------------------
c
c
c     NSEC = # of sections (streamwise stations) of the new grid
c     NPTS = # of pts in the circumferencial direction (MUST be odd)
c     NPX=max streamwise stations
c     FAC = the first grid spacing in DISTRI
c     XLE = X leading edge
c     ARRWING > 0,  arrow wing
c     KTIP = number of points in the wrap-around direction on one surface
c     NC=number of cut in the spanwise direction
c     NU=number of points in the upper part of the wing
c     NL=number of points in the lower part of the wing
c
c
c     Read surfgrid dimensions (nsec x npts x 1)
      NAMELIST /WING/ NSEC,NPTS,FAC
      READ(40,WING)
      WRITE(*,WING)
c
c     Read the input grid
c----------------------------------------------------------------
      CALL WINGIN
c----------------------------------------------------------------
c
c
c     Setup distribution of cross-sections to be obtained (xdist)
      XRLE=XLE(1)
      XRTE=XLE(1)+CHORD(1)
      WRTE=XLE(NC)+CHORD(NC)
      IF(XRTE.GE.WRTE) THEN
        XRT = XRTE
        ARRWING = -1.
      ELSE
        XRT = WRTE
        ARRWING =  1.
      ENDIF
      WAKE = AMIN1(XRTE,WRTE)
      DO 100 J=1,NSEC
       XDIST(J)=XRLE+(XRT-XRLE)*(FLOAT(J-1)/FLOAT(NSEC-1))
100   CONTINUE
c
      KTIP=(NPTS+1)/2
c
c     The nose of the wing
      DO 187 K=1,NPTS
        XOUT(1,K)=XDIST(1)
        YOUT(1,K)=YBASE(1,1,1)
        ZOUT(1,K)=ZBASE(1,1,1)
187   CONTINUE
c
c     Begin main loop for each x-station
c
      DO 1000 L=2,NSEC
c
        XLOCAL=XDIST(L)
c
c       Redistribute the points from spanwise cut to streamwise cut.
c       The output ZDIST and YNEW are from the root to the tip, therefore
c       when doing the lower surface, need to rearrange the argument.
c       The output (ZDIST,YNEW) in both surfaces have KTIP # of pts in
c       the circumferencial direction, their last point have the same physical
c       value for both surfaces.
c
c
c       Do the lower surface
        CALL REDIST(XLOCAL,KTIP,FAC,1)
        DO 300 K=1,KTIP
          XOUT(L,K) = XLOCAL
          YOUT(L,K) = YNEW(K)
          ZOUT(L,K) = ZDIST(K)
 300    CONTINUE
c
c       Do the upper surface
        CALL REDIST(XLOCAL,KTIP,FAC,2)
        DO 400 K=KTIP+1,NPTS
          XOUT(L,K) = XLOCAL
          YOUT(L,K) = YNEW(NPTS-K+1)
          ZOUT(L,K) = ZDIST(NPTS-K+1)
 400    CONTINUE
c
c       For the computational grid of UPS3D code
c       the wake has to have two different pts in same
c       physical location, such that (Y1,Z1)=(Y2,Z2).  Here
c       the calculation divided into upper and lower parts,
c       for safty sake, set Z1=Z2.
        IF(XLOCAL.LT.WAKE) GOTO 900
        DO 500 K=1,KTIP-1
          K1=KTIP+K
          K2=KTIP-K
*Hagland model         IF(ABS(YOUT(L,K1)-YOUT(L,K2)) .LE. 3.0E-4) THEN
          IF(ABS(YOUT(L,K1)-YOUT(L,K2)) .LE. 1.0E-2) THEN
*Ref-H           IF(ABS(YOUT(L,K1)-YOUT(L,K2)) .LE. 1.0E-5) THEN
            ZOUT(L,K1)=ZOUT(L,K2)
            YOUT(L,K1)=YOUT(L,K2)
          ENDIF
500     CONTINUE
900     CONTINUE
c
c       Proceed to next x coordinate
c
1000  CONTINUE
c
c     Write out new surfgrid in plot3d format
```

LINE # | SOURCE TEXT

```
C
      KW=1
      WRITE(50)NPTS,KW,NSEC
      DO 1234 L=1,NSEC
        WRITE(50)(XOUT(L,K),K=1,NPTS),
     .           (YOUT(L,K),K=1,NPTS),
     .           (ZOUT(L,K),K=1,NPTS)
 1234 CONTINUE
C
C     Write out original database in plot3d format
C
      N1=NU+NL
      WRITE(11)N1,NC,KW
      WRITE(11)  ((XBASE(I,1,M),I=1,NU),(XBASE(I,2,M),I=NL,1,-1),
     .           M=1,NC),
     .           ((YBASE(I,1,M),I=1,NU),(YBASE(I,2,M),I=NL,1,-1),
     .           M=1,NC),
     .           ((ZBASE(I,1,M),I=1,NU),(ZBASE(I,2,M),I=NL,1,-1),
     .           M=1,NC)
C
C     Read the fuselage grid and combine the fusleage with the
C     wing grid to form a whole configuration.
      CALL WBGRID
C
C     Read the nacelles grid and combine the nacelles with the
C     wing-body grid.
C      CALL NACGRID
C
      CLOSE(10)
C      CLOSE(30)
      CLOSE(40)
      STOP
      END
```

```
*************************************************************************
      SUBROUTINE ADDGRID(NPL1,NPL2,X,Y,Z,NPI,NSEC,KDIM)
C
C     This subroutine allows us to add a grid line between streamwise section
C     NPL1 and NPL2, and the new dimension is NSEC again
C
      PARAMETER (MAX=400)
      DIMENSION YTEMP(MAX),ZTEMP(MAX)
      DIMENSION X(NPI),Y(NPI,NPI),Z(NPI,NPI)
C
      IF(MAX.LE.NPI) THEN
        WRITE(*,*)'SUB ADDGRID : MAX is less than NPI'
        STOP
      ENDIF
      IF(NPL1.GE.NPL2) THEN
        WRITE(*,*)'No plane is added in the streamwise direction'
        STOP
      ENDIF
C
C     Interpolating the new grid, and put it in a temporary array
      X1 = X(NPL1)
      X2 = X(NPL2)
      XX = 0.5*(X(NPL1)+X(NPL2))
      DO 10 K=1,KDIM
        Y1 = Y(NPL1,K)
        Y2 = Y(NPL2,K)
        Z1 = Z(NPL1,K)
        Z2 = Z(NPL2,K)
        CALL LININT(X1,X2,Y1,Y2,XX,YY)
        CALL LININT(Y1,Y2,Z1,Z2,YY,ZZ)
        YTEMP(K) = YY
        ZTEMP(K) = ZZ
 10   CONTINUE

C     Renumber the late stations
      NSEC = NSEC+1
      DO 30 L= NSEC,NPL2+1,-1
        X(L) = X(L-1)
        DO 20 K=1,KDIM
          Y(L,K) = Y(L-1,K)
          Z(L,K) = Z(L-1,K)
 20     CONTINUE
 30   CONTINUE

C     Put the temporary array in the grid
      DO 50 K=1,KDIM
          X(NPL2) = XX
          Y(NPL2,K) = YTEMP(K)
          Z(NPL2,K) = ZTEMP(K)
 50   CONTINUE

      RETURN
      END
```

SOURCE TEXT

LINE #

```
*************************************************************************
      SUBROUTINE CIRCLE(KS,KE,KMAX,Y,Z,RFIL,ARCORR)
      DIMENSION Z(KMAX),Y(KMAX)
C     Given a set of pts (Y(i),Z(i)) i=1,..,KMAX, and radius of filet RFIL,
C     this subroutine replaces the points (Y(j),Z(j)) j=KS,..,KE by the filet
C     points on filet circle.

C     Look for the center of the filet circle (YC,ZC)
      YA=Y(KS)
      ZA=Z(KS)
      YB=Y(KE)
      ZB=Z(KE)
C
      SY=YA-YB
      SZ=ZA-ZB
      BB=(YA*YA-YB*YB)+(ZA*ZA-ZB*ZB)
      R =SY/SZ
C
      A = 1.+R**2
      B = 2.*ZB*R - 2.*YB - BB*R/SZ
      C = ZB*ZB+YB*YB + (BB/(2.*SZ))**2 - ZB*BB/SZ - RFIL**2

      DET=B*B-4.*A*C
      IF(DET.LE.0.) THEN
        WRITE(15,*)'Determinant is less than 0, ',DET
        GOTO 200
      ENDIF
      YC1 = ( -B+SQRT(DET) ) / (2.*A)
      ZC1 = (BB - 2.*YC1*SY)/(2.*SZ)
      YC2 = ( -B-SQRT(DET) ) / (2.*A)
      ZC2 = (BB - 2.*YC2*SY)/(2.*SZ)
      IF(ZC1.GE.ZC2) THEN
        ZC=ZC1
        YC=YC1
      ELSE
        ZC=ZC2
        YC=YC2
      ENDIF

C
C     Find the total arc length given
      TOTARC=0.
      DO 50 K=KS,KE-1
        TOTARC=TOTARC +
     &  SQRT( (Y(K+1)-Y(K))**2 + (Z(K+1)-Z(K))**2 )
 50   CONTINUE
C
C     Find the arc length ./. two points.
      ARC = TOTARC/FLOAT(KE-KS)
      ARC = ARC*ARCORR
C
C
C     Find the coordinates for each point
      DO 100 K=KS,KE-1
        YA=Y(K)
        ZA=Z(K)
        YB=YC
        ZB=ZC
C
        SY=YA-YB
        SZ=ZA-ZB
        SR=RFIL**2-ARC**2
        BB=(YA*YA-YB*YB)+(ZA*ZA-ZB*ZB)
        R =SY/SZ
C
        A = 1.+R**2
        B = 2.*ZB*R - 2.*YB - BB*R/SZ
     &    - R*SR/SZ
        C = ZB*ZB+YB*YB + (BB/(2.*SZ))**2 - ZB*BB/SZ - RFIL**2
     &    + (0.5*SR/SZ)**2 + (BB*SR)/(2.*SZ*SZ) - SR*ZB/SZ
C
        DET=B*B-4.*A*C
        IF(DET.LE.0.) THEN
          WRITE(15,*)'Determinant is less than 0, ',DET
          GOTO 200
        ENDIF
        YC1 = ( -B+SQRT(DET) ) / (2.*A)
        ZC1 = (BB - 2.*YC1*SY + (RFIL**2-ARC**2))/(2.*SZ)
        YC2 = ( -B-SQRT(DET) )/ (2.*A)
        ZC2 = (BB - 2.*YC2*SY + (RFIL**2-ARC**2))/(2.*SZ)
        IF(YC1.GE.Y(K)) THEN
          Z(K+1)=ZC1
          Y(K+1)=YC1
        ELSE
          Z(K+1)=ZC2
          Y(K+1)=YC2
        ENDIF
 100  CONTINUE
 200  RETURN
      END
```

LINE # SOURCE TEXT

```
C*********************************************************************
      SUBROUTINE CSPLINE(X,Y,N,XNEW,YNEW,NNEW)
C
      PARAMETER (NMAX=500)
C
      REAL X(N), Y(N), XNEW(NNEW), YNEW(NNEW)
C
C ********************************************************************
C
C         THIS SUBROUTINE PRODUCES A MONOTONE CUBIC SPLINE INTERPOLANT
C     TO THE DATA (X(I),Y(I)) I=1,....N  AND COMPUTES VALUES AT
C     THE NEW POINTS XNEW(I), I=1,....,NNEW. THESE ARE RETURNED IN
C     ARRAY YNEW(I). THE ALGORITHM USED IS THAT OUTLINED BY FRITSCH AND
C     BUTLAND IN SIAM J. SCI. STAT. COMPUT., VOL. 5, NO. 2, JUNE, 1984.
C
C.....WRITTEN BY JEFF CORDOVA 10/26/86
C
C ********************************************************************
C
      REAL D(NMAX), DEL(NMAX), H(NMAX)
C
C ********************************************************************
C                    SPLINE COEFFICIENT CALCULATIONS
C ********************************************************************
C
C.....MESH SPACING AND FIRST DIVIDED DIFFERENCE
C
      DO 100 I=1,N-1
        H(I) = X(I+1) - X(I)
  100 CONTINUE
C
      DO 200 I=1,N-1
        DEL(I) = (Y(I+1) - Y(I)) / H(I)
  200 CONTINUE
C
C.....SPLINE COEFFICIENTS
C
C         *** LINEAR INTERPOLATION FOR N=2 CASE ***
C
      IF (N .EQ. 2)  THEN
        D(1) = DEL(1)
        D(N) = DEL(1)
        GO TO 399
      ENDIF
C
C         *** MONOTONE SPLINE COEFFICIENTS FOR N >= 3 CASE ***
C
C.....FIRST BOUNDARY POINT (USE THREE POINT FORMULA ALTERED TO BE
C     SHAPE PRESERVING)
C
      HSUM = H(1) + H(2)
      W1   = (H(1) + HSUM) / HSUM
      W2   = -H(1) / HSUM
      D(1) = W1*DEL(1) + W2*DEL(2)
      IF (PCHST(D(1),DEL(1)) .LE. 0.)  THEN
        D(1) = 0.
      ELSEIF (PCHST(DEL(1),DEL(2)) .LT. 0.)  THEN
        DMAX = 3.*DEL(1)
        IF (ABS(D(1)) .GT. ABS(DMAX))  D(1) = DMAX
      ENDIF
C
C.....INTERIOR POINTS (BRODLIE MODIFICATION OF BUTLAND FORMULA)
C
      CONST = 1. / 3.
      DO 300  I=2,N-1
        TOP   = DEL(I-1) * DEL(I)
        TOP   = TOP * .5 * (1. + SIGN(1.,TOP))
        ALPHA = CONST * (H(I-1) + 2.*H(I)) / (H(I-1) + H(I))
        BOT   = ALPHA * DEL(I) + (1.-ALPHA) * DEL(I-1) + 1.E-20
        D(I)  = TOP / BOT
  300 CONTINUE
C
C.....LAST BOUNDARY POINT (USE THREE POINT FORMULA ADJUSTED TO BE
C     SHAPE PRESERVING)
C
      HSUM = H(N-2) + H(N-1)
      W1   = -H(N-1) / HSUM
      W2   = (H(N-1) + HSUM) / HSUM
      D(N) = W1*DEL(N-2) + W2*DEL(N-1)
      IF (PCHST(D(N),DEL(N-1)) .LE. 0.)  THEN
        D(N) = 0.
      ELSEIF ( PCHST(DEL(N-2),DEL(N-1)) .LT. 0.)  THEN
        DMAX = 3.*DEL(N-1)
        IF (ABS(D(N)) .GT. ABS(DMAX))  D(N) = DMAX
      ENDIF
C
  399 CONTINUE
C
C ********************************************************************
C                         SPLINE EVALUATION
C ********************************************************************
C
C.....XNEW(I) .LE. X(N)
C
      IEND = 1
      DO 400 J=1,N-1
        CTHREE = (D(J) + D(J+1) - 2.*DEL(J)) / (H(J)*H(J))
        CTWO   = (3.*DEL(J) - 2.*D(J) - D(J+1)) / H(J)
        IBEG = IEND
CRAY      IEND = ISRCHFGE(NNEW,XNEW,1,X(J+1))    !On CRAY
        IEND = ISRCHGE(NNEW,XNEW,1,X(J+1))    !On WK
        DO 500 I=IBEG,IEND-1
          T =  XNEW(I) - X(J)
          YNEW(I) = Y(J) + T*(D(J) + T*(CTWO + T*CTHREE))
  500   CONTINUE
  400 CONTINUE
C
C.....XNEW(I) .GT. X(N)
C
      DO 600 I=IEND,NNEW
        T =  XNEW(I) - X(N-1)
        YNEW(I) = Y(N-1) + T*(D(N-1) + T*(CTWO + T*CTHREE))
  600 CONTINUE
C
      RETURN
      END
```

SOURCE TEXT

```
      FUNCTION  PCHST(ARG1,ARG2)
C
      PCHST = SIGN(1.,ARG1) * SIGN(1.,ARG2)
      IF ((ARG1.EQ.0.) .OR. (ARG2.EQ.0.)) PCHST = 0.
C
      RETURN
      END
```

LINE # SOURCE TEXT

```
      FUNCTION ISRCHGE(N,X,INCX,FTARGET)
      DIMENSION X(*)
      IF(N.LE.0) THEN
        ISRCHGE = 0
        RETURN
      ELSE
        IT = 1 + (N-1) * INCX
        ISRCHGE = 1
        DO 10 I=1,IT,INCX
          IF(X(I).GE.FTARGET) GOTO 11
          ISRCHGE = ISRCHGE + 1
10      CONTINUE
11      CONTINUE
      ENDIF
      RETURN
      END
```

LINE #

```
*****************************************************************************
      SUBROUTINE CUSTER
      include "sgrid.com"
      DIMENSION YWK(NPI),ZWK(NPI)
      L2 = L-1
      NFUS = KDIM - NPTS
      NBOT = NFUS/2 + 1
      NTOP = NFUS/2 + 1
C
      DO 900 LL=M1,L2
C       Note: I am leaving the nose and the wake alone
        IF(ABS(Y(LL,NBOT)-Y(LL,KDIM-NTOP+1)).LE.1.E-7) THEN
          RETURN
        ENDIF
C
C       Do the bottom first:
C
        DO 10 K=1,NBOT
          YINT(K) = Z(LL,K)
          ZINT(K) = Y(LL,K)
 10     CONTINUE
        FGSP = SQRT((Z(LL,NBOT)-Z(LL,NBOT+1))**2 +
     &              (Y(LL,NBOT)-Y(LL,NBOT+1))**2 )
        CALL DISTARC(YINT,ZINT,NBOT,YWK,ZWK,NBOT,FGSP,1)
        DO 80 K=1,NBOT
          Y(LL,K) = ZWK(K)
          Z(LL,K) = YWK(K)
 80     CONTINUE
C
C       Do the top now:
C
        DO 100 K=1,NTOP
          YINT(K) = Y(LL,K+(KDIM-NTOP))
          ZINT(K) = Z(LL,K+(KDIM-NTOP))
 100    CONTINUE
        N1 = (KDIM-NTOP)
        N2 = (KDIM-NTOP)-1
        FGSP = SQRT((Z(LL,N1)-Z(LL,N2))**2 +
     &              (Y(LL,N1)-Y(LL,N2))**2 )
        CALL DISTARC(YINT,ZINT,NBOT,YWK,ZWK,NTOP,FGSP,0)
        DO 180 K=1,NTOP
          Y(LL,K+(KDIM-NTOP)) = YWK(K)
          Z(LL,K+(KDIM-NTOP)) = ZWK(K)
 180    CONTINUE
C
 900  CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
************************************************************************
      SUBROUTINE DISTARC(X,Y,N,XNEW,YNEW,NNEW,FGS,IFLAT)
C
      DIMENSION X(N),Y(N),XNEW(NNEW),YNEW(NNEW)
C
C     This program redistribute the points (X,Y) by subroutine DISTRI
C     based on the arc length.  FGS is the first grid spacing. Note that
C     the end points of the two sets are the same.
C     IFLAT=0, grid points will custer near the first point, =1 near the end.
C     Input  array is (X(i),Y(i)), i=1,...,N,
C     Output array is (XNEW(i),YNEW(i)), i=1,...,NNEW
C
C
      PARAMETER (MAX=400)
      DIMENSION S(MAX),TOTARC(MAX),XN(MAX),YN(MAX)
C
C     Maximum number of points allowed is MAX
      IF(MAX.LE.N .OR. MAX.LE.NNEW) THEN
        WRITE(*,*) 'SUB DISTARC : MAX is less than N or NNEW'
        STOP
      ENDIF
C
C     Look for total arc length
      TOTARC(1) = 0.
      DO 10 K=2,N
        ARC = SQRT( (X(K)-X(K-1))**2 + (Y(K)-Y(K-1))**2 )
        TOTARC(K) = TOTARC(K-1) + ARC
 10   CONTINUE
C
C     Apply subroutine DISTRI to obtain the stratching function S
C     For FGS<0, equal spacing is used
      IF(FGS.GT.0.) THEN
        DELT=FGS/TOTARC(N)
        CALL DISTRI(DELT,NNEW,S,IFLAT)
      ELSE
        S(1) = 0.
        DO 25 K=2,NNEW
          S(K) = S(K-1) + 1./FLOAT(NNEW-1)
 25     CONTINUE
      ENDIF
C
C     Redistribution, put new array in a temporary arrays XN and YN
      XN(1)=X(1)
      YN(1)=Y(1)
      XN(NNEW)=X(N)
      YN(NNEW)=Y(N)
C
      DO 60 J = 2,NNEW
        ARCNEW =  S(J)*TOTARC(N)
        DO 55 K = 2,N
          IF(ABS(TOTARC(K)-ARCNEW).LE.1.E-7) THEN
            XN(J) = X(K)
            YN(J) = Y(K)
            GOTO 60
          ENDIF
          IF(TOTARC(K).GT.ARCNEW) THEN
            X1 = X(K-1)
            X2 = X(K)
            Y1 = Y(K-1)
            Y2 = Y(K)
            XX = X1 + (X(K)-X(K-1))*
     &           (ARCNEW-TOTARC(K-1))/(TOTARC(K)-TOTARC(K-1))
            CALL LININT(X1,X2,Y1,Y2,XX,YY)
            XN(J) = XX
            IF(ABS(X1-X2).LE.1.E-7) THEN
              YN(J) = Y1 + (ARCNEW-TOTARC(K-1))
            ELSE
              YN(J) = YY
            ENDIF
            GOTO 60
          ENDIF
 55     CONTINUE
 60   CONTINUE

C
C     Write the temporary arrays into the output XNEW, YNEW
      DO 70 J=1,NNEW
        XNEW(J) = XN(J)
        YNEW(J) = YN(J)
 70   CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
************************************************************************
      SUBROUTINE DISTRI(FANG,KFCS,S,IFINE)
      PARAMETER (MAX=400)
      DIMENSION S(MAX),DUM(MAX)
C
C.....Calculating the stretching function S when given
C     the first spacing, FANG, and the number of points KFCS
C.....If IFINE=1, distribution is cusltering at outer grid
C
      IF(MAX.LE.KFCS) THEN
        WRITE(*,*)'SUB-DISTRI : MAX is less than KFCS'
        STOP
      ENDIF

      IF(KFCS.EQ.1) THEN
        S(1) = 0.
        GOTO 40
      ENDIF
C
      DZ1 = FANG
      KFM = KFCS-1
      DZETA = 1./FLOAT(KFM)
      RDBETA = 1.5
      CALL GRBET(DZ1,KFM,0.0001,100,RDBETA)
      CALL FZ1(KFCS,RDBETA,DZETA,S)
C
      IF (IFINE.EQ.1) THEN
          DO 37 K=1,KFCS
            DUM(KFCS-K+1) = S(K)
 37       CONTINUE
          DO 38 K=1,KFCS
            S(K) = 1.-DUM(K)
 38       CONTINUE
      ENDIF
 40   CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
************************************************************************
      SUBROUTINE EDGE(NC,NU,NL,XL,XBK,XBASE,YBASE,ZBASE,NPK,LS)
      DIMENSION ZBASE(NPK,2,LS),XBASE(NPK,2,LS),YBASE(NPK,2,LS)

      ZLE = ZBASE(1,1,1)
      DO 200 K =1,NC
        IF(XBASE(1,1,K).GT.XBK) THEN
          X1 = XBASE(1,1,K-1)
          X2 = XBASE(1,1,K  )
          Z1 = ZBASE(1,1,K-1)
          Z2 = ZBASE(1,1,K  )
          CALL LININT(X1,X2,Z1,Z2,XBK,ZBK)
          GOTO 210
        ENDIF
  200 CONTINUE
  210 CONTINUE

C
      DO 500 K=1,NC
        IF(ZBASE(1,1,K).GT.ZBK) GOTO 700
        CALL LININT(ZLE,ZBK,XL,XBK,ZBASE(1,1,K),XLE)
        XLEOLD = XBASE(1,1,K)
        XTL    = XBASE(NL,1,K)
        DO 280 I=1,NL
          F = XBASE(I,1,K)-XLEOLD
          E = XTL-XBASE(I,1,K)
          XBASE(I,1,K) = (F*XTL + E*XLE)/(F+E)
  280   CONTINUE
        XTL    = XBASE(NU,1,K)
        DO 300 I=1,NU
          F = XBASE(I,2,K)-XLEOLD
          E = XTL-XBASE(I,2,K)
          XBASE(I,2,K) = (F*XTL + E*XLE)/(F+E)
  300   CONTINUE
  500 CONTINUE
  700 CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
***********************************************************************
      SUBROUTINE EQSPACE
      include "sgrid.com"
      L2 = L-1
      DO 130 LL=1,L2
        XIN(LL) = X(LL)
        DO 120 K=1,KDIM
          ZIN(LL,K) = Z(LL,K)
          YIN(LL,K) = Y(LL,K)
 120    CONTINUE
 130  CONTINUE

      XTOT = X(L2)-X(1)
      DX   = XTOT/FLOAT(L2-1)
      DO 160 JL=2,L2
        X(JL) = X(JL-1)+DX
        DO 150 KL=1,L2
          IF(ABS(XIN(KL)-X(JL)) .LE. 1.E-7) THEN
            DO 140 K=1,KDIM
              Z(KL,K) = ZIN(JL,K)
              Y(KL,K) = YIN(JL,K)
 140        CONTINUE
            GOTO 160
          ENDIF
          IF(XIN(KL).GT.X(JL)) THEN
            DO 145 K=1,KDIM
              X1 = XIN(KL-1)
              X2 = XIN(KL)
              Y1 = YIN(KL-1,K)
              Y2 = YIN(KL,K)
              Z1 = ZIN(KL-1,K)
              Z2 = ZIN(KL,K)
              XX = X(JL)
              CALL LININT(X1,X2,Y1,Y2,XX,YY)
              CALL LININT(X1,X2,Z1,Z2,XX,ZZ)
              Y(JL,K) = YY
              Z(JL,K) = ZZ
 145        CONTINUE
            GOTO 160
          ENDIF
 150    CONTINUE
 160  CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
******************************************************************************
      SUBROUTINE FILET(Y,Z,KDIM,MB1,MB2,MT1,MT2,RFIL)
      PARAMETER (MAX=400)
      DIMENSION Z(KDIM),Y(KDIM)
      DIMENSION D1(MAX),D2(MAX)
      COMMON /REF/ ZROOT,KTIP,ARCORR
c
c     This subroutine takes a wing-fuselage station, (Y(k),Z(k)) k=1,KDIM,
c     and find the two (top and bottom) intersections of the wing and the
c     fuselage.
c     And then, for example, at the bottom intersection (Y(K),Z(K)), it
c     extends to a segment of points, (Y(k),Z(K)) k=KF1 to KF2, where
c     KF1=K-MB1, KF2=K+MB2.
c     Similar procedure for the top part.
c     And then, call subroutine CIRCLE to replace the segment by a segment
c     of a circle with radius RFIL.
c

      IF(RFIL.EQ.0.) GOTO 735

      IF(MAX.LE.KDIM) THEN
        WRITE(*,*)'SUB FILET : MAX is less than KDIM'
        STOP
      ENDIF

c     Bottom part of the aircraft
      IF(MB1.EQ.0 .AND. MB2.EQ.0) GOTO 135
      DO 130 K=1,KDIM
        IF (K.LE.KTIP .AND. Z(K).GT.ZROOT) THEN
          KF1=K-MB1
          KF2=K+MB2
          CALL CIRCLE(KF1,KF2,KDIM,Y,Z,RFIL,ARCORR)
c
c         We have N=KF2-KF1+1 pts in filet area, employ two more points
c         from the original grid and redistribute them, the grid spacing looks
c         smoother.
          KF1=KF1+1
          KF2=KF2+1
          DO 220 KD=KF1,KF2
            D1(KD-KF1+1) = Y(KD)
            D2(KD-KF1+1) = Z(KD)
220       CONTINUE
          N=KF2-KF1+1
          CALL DISTARC(D2,D1,N,D2,D1,N,-10.,0)
          DO 330 KD=KF1,KF2
            Y(KD)=D1(KD-KF1+1)
            Z(KD)=D2(KD-KF1+1)
330       CONTINUE
          GOTO 135
        ENDIF
130   CONTINUE
135   CONTINUE

c     Top part of the aircraft
      IF(MT1.EQ.0 .AND. MT2.EQ.0) GOTO 735
      DO 700 K=KTIP,KDIM
        IF(K.GT.KTIP .AND. Z(K).LE.ZROOT) THEN
          KF1=K-MT2
          KF2=K+MT1
          CALL CIRCLE(KF1,KF2,KDIM,Y,Z,RFIL,ARCORR)
c
c         We have N=KF2-KF1+1 pts in filet area, employ two more points
c         from the original grid and redistribute them, the grid spacing looks
c         smoother.
          KF1=KF1+1
          KF2=KF2+1
          DO 420 KD=KF1,KF2
            D1(KD-KF1+1) = Y(KD)
            D2(KD-KF1+1) = Z(KD)
420       CONTINUE
          N=KF2-KF1+1
          CALL DISTARC(D1,D2,N,D1,D2,N,-10.,0)
          DO 530 KD=KF1,KF2
            Y(KD)=D1(KD-KF1+1)
            Z(KD)=D2(KD-KF1+1)
530       CONTINUE
          GOTO 735
        ENDIF
700   CONTINUE
735   CONTINUE
c
      RETURN
      END
```

LINE # SOURCE TEXT

```
************************************************************************
      SUBROUTINE FZ1(L1,TBETA,DET,Z)
C
C     COMPUTES NORMALIZED NORMAL DISTANCE, Z(L)
C
      PARAMETER (MAX=400)
      DIMENSION Z(MAX)

      IF(MAX.LE.L1) THEN
        WRITE(*,*)'SUB FZ1 : MAX is less than L1'
        STOP
      ENDIF

      IF(TBETA.EQ.1.) THEN
      DO 10 L=1,L1
      Z(L)=0.
   10 CONTINUE
      ELSE
      DO 20 L=1,L1
      ETA=(L-1)*DET
      RR=(TBETA+1.)/(TBETA-1.)
      EEE=1.-ETA
      RBETA=RR**EEE
      Z(L)=(TBETA-1.)*(RR-RBETA)/(RBETA+1.)
   20 CONTINUE
      END IF
      RETURN
      END
```

LINE # SOURCE TEXT

```
***********************************************************************
      SUBROUTINE GRBET(DFM,NPT,FPCC,ICC,BETA)
C
C     BISECTION METHOD USED TO DETERMINE STRETCHING PARAMETER, BETA,
C     WHICH GIVES DESIRED dY AT THE WALL
C
      PARAMETER (MAX=400)
      DIMENSION Z(MAX)
      IF(MAX.LE.NPT) THEN
        WRITE(*,*)'SUB GRBET : MAX is less than NPT'
        STOP
      ENDIF
C
      ICCL=ICC
      FPCCL=FPCC*DFM
      BETA1=BETA
      Z1=DFM
      DET=1./NPT
      BR=1.
      FR=-Z1
      IICC=ICC/10
      DO 10 I=1,IICC
      BF=BETA1
      BETA=0.5*(BETA1+1.)
      CALL FZ1(2,BF,DET,Z)
      FF=Z(2)-Z1
      IF(FF.GT.0.) GO TO 15
      BETA1=2.*BETA1-1.
   10 CONTINUE
   15 CONTINUE
      DO 5 NIT=1,ICCL
      CALL FZ1(2,BETA,DET,Z)
      F=Z(2)-Z1
      IF(F.GT.0.) THEN
      FF=F
      BF=BETA
      ELSE
      FR=F
      BR=BETA
      END IF
      BETA=0.5*(BF+BR)
      IF(ABS(F).LT.FPCCL) GO TO 4
    5 CONTINUE
C     WRITE(6,100) BETA,F
  100 FORMAT(1H0,36H EXCEEDED MAX. NO. OF ITS....BETA,F ,3G13.6)
    4 CONTINUE
C     CALL FZ1(2,BETA,DET,Z)
C     F=Z(2)-Z1
      BM1=BETA-1.
      BFM1=BF-1.
      BRM1=BR-1.
      RETURN
      END
```

LINE # | SOURCE TEXT

```
************************************************************************
      SUBROUTINE LININT(X1,X2,Y1,Y2,XLOCAL,YLOCAL)
C     This subroutine linearly interpolate YLOCAL when given (X1,Y1) & (X2,Y2)
      IF(ABS(X1-X2).LE.1.E-7) THEN
        YLOCAL=(Y2+Y1)/2.
        GOTO 100
      ENDIF
      SLOPE = (Y2-Y1)/(X2-X1)
      YLOCAL = SLOPE*(XLOCAL-X2) + Y2
100   CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```fortran
***********************************************************************
      SUBROUTINE MOUNT(JN,LNUM,IPRNT)
      include "sgrid.com"
      DIMENSION NPAIR(2,NPI),LNUM(4)
      DIMENSION YWK(NPI),ZWK(NPI),YNAC(NPI),ZNAC(NPI)
      DIMENSION YWNG(2*NPI),ZWNG(2*NPI)
      COMMON /ENG/  XENG(2,NPI),YENG(2,NPI,NPI),ZENG(2,NPI,NPI)
     &              ,MNAC(2),MNACP(2)
c
c     (XENG,YENG,ZENG)      Coordinate of engine
c     MNAC(*)               Number of stations in nacelle
c     MNACP(*)              Number of points in each station
c     (1,*,*)  inner nacelle, (2,*,*) outer nacelle
c
c     MC       =  # of pts added in the grid ( =# pts at nacelle)
c     IPRNT    =  0 no writing out
c
      MC = 40
      NPWN = NPTS+MC
      IF(LNUM(1).EQ.LNUM(3)  .AND. LNUM(2).EQ.LNUM(4)) THEN
        NPTS = NPTS + MC
        NPWN = NPTS + MC
      ENDIF
      NPH = (NPTS+1)/2
      NPWNH = (NPWN+1)/2
      NNAC = MNAC(JN)
      NNACP = MNACP(JN)
c
      IF(IPRNT.NE.0) WRITE(IPRNT)NPWN,1,LNUM(2)-LNUM(1)+1
c
c     Now the big job!
c
      DO 800 L=LNUM(1),LNUM(2)
c
c       Interpolate the points of the nacelle at x=XOUT
        DO 100 LN=1,NNAC
          IF(ABS(XENG(JN,LN)-XOUT(L,1)).LE.1.E-7) THEN
            DO K=1,NNACP
              YNAC(K) = YENG(JN,LN,K)
              ZNAC(K) = ZENG(JN,LN,K)
            ENDDO
            GOTO 105
          ELSEIF(XENG(JN,LN).GT.XOUT(L,1)) THEN
            DO K=1,NNACP
              X1 = XENG(JN,LN-1)
              Y1 = YENG(JN,LN-1,K)
              Z1 = ZENG(JN,LN-1,K)
              X2 = XENG(JN,LN  )
              Y2 = YENG(JN,LN  ,K)
              Z2 = ZENG(JN,LN  ,K)
              XX = XOUT(L,1)
              CALL LININT(X1,X2,Y1,Y2,XX,YY)
              CALL LININT(X1,X2,Z1,Z2,XX,ZZ)
              YNAC(K) = YY
              ZNAC(K) = ZZ
            ENDDO
            GOTO 105
          ENDIF
 100    CONTINUE
 105    CONTINUE
        RADNAC = SQRT( (ZNAC(1)-ZNAC(NNAC/2))**2 +
     &                 (YNAC(1)-YNAC(NNAC/2))**2 )

c       Count the number of points in the wake if we are in the wake
        KOUNT = 0
        DO 120 K=1,NPH-1
          K1=NPH-K
          K2=NPH+K
          IF(ABS(YOUT(L,K1)-YOUT(L,K2)).LE.1.E-7 .AND.
     &    ABS(ZOUT(L,K1)-ZOUT(L,K2)).LE.1.E-7 ) THEN
            KOUNT = KOUNT + 1
            NPAIR(1,KOUNT) = K1
            NPAIR(2,KOUNT) = K2
          ENDIF
 120    CONTINUE
c
c       Nacelle totally under the wing, INTRAIL = 0
c       Part under the wing, part in the wake, INTRAIL = 1
c       Nacelle totally in the wake, INTRAIL = 2
        INTRAIL = 0
        IF(KOUNT.EQ.0) GOTO 415
        IF(ZOUT(L,NPAIR(1,1)).GT.ZNAC(1    )) INTRAIL=1
        IF(ZOUT(L,NPAIR(1,1)).GT.ZNAC(NNACP)) INTRAIL=2
        write(*,*)'INTRAIL = ' ,INTRAIL
        IF(INTRAIL.NE.2) THEN
c         We are not in the wake region or the trailing edge
          GOTO 415
        ELSE
c         We are in the wake region or the trailing edge
          WRITE(*,*)'We are in the wake region'
          NTEMP = 1
 130      CONTINUE
c         Obtain all points under the wake line
          DO 180 K=1,KOUNT
            IF(ZOUT(L,NPAIR(1,K)).LT.ZNAC(NTEMP)) THEN
              KS = K
              r1 = abs(zout(L,NPAIR(1,K))-zout(L,NPAIR(1,K+1)))
              r2 = abs(zout(L,NPAIR(1,K))-ZNAC(NTEMP))
              if(r2.le.r1/8.) KS=K+1
              IF(YOUT(L,NPAIR(1,KS)).LT.YNAC(NTEMP)) THEN
                NTEMP = NTEMP + 1
                GOTO 130
              ENDIF
              GOTO 185
            ENDIF
 180      CONTINUE
 185      CONTINUE
          NS = NTEMP
          IF(NS.NE.1) THEN
            CALL LININT(YNAC(NS-1),YNAC(NS),ZNAC(NS-1),ZNAC(NS),
     &                  YOUT(L,NPAIR(1,KS)),ZZS)
            YYS = YOUT(L,NPAIR(1,KS))
          ELSE
            CALL LININT(ZOUT(L,NPAIR(1,KS)),ZOUT(L,NPAIR(1,KS)-1),
     &                  YOUT(L,NPAIR(1,KS)),YOUT(L,NPAIR(1,KS)-1),
     &                  ZNAC(NS),YYS)
            ZZS = ZNAC(NS)
          ENDIF

          NTEMP = NNACP
```

LINE # SOURCE TEXT

```
190       CONTINUE
          IF(INTRAIL.EQ.2) THEN
            DO 200 K=1,KOUNT
              IF(ZOUT(L,NPAIR(1,K)).LT.ZNAC(NTEMP)) THEN
                KE = K-1
                IF(YOUT(L,NPAIR(1,KE)).LT.YNAC(NTEMP)) THEN
                  NTEMP = NTEMP - 1
                  GOTO 190
                ENDIF
                GOTO 215
              ENDIF
200         CONTINUE
215         CONTINUE
            NE = NTEMP
            IF(NTEMP.NE.NNACP) THEN
              CALL LININT(YNAC(NE-1),YNAC(NE),ZNAC(NE-1),ZNAC(NE),
     &                    YOUT(L,NPAIR(1,KE)),ZZE)
              YYE = YOUT(L,NPAIR(1,KE))
            ELSE
              CALL LININT(ZOUT(L,NPAIR(1,KE)),ZOUT(L,NPAIR(1,KE)+1),
     &                    YOUT(L,NPAIR(1,KE)),YOUT(L,NPAIR(1,KE)+1),
     &                    ZNAC(NE),YYE)
              ZZE = ZNAC(NE)
            ENDIF
          ELSE
            DO 230 K=1,NPH
              IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
                KE = K
                IF(YOUT(L,KE).LT.YNAC(NTEMP)) THEN
                  NTEMP = NTEMP - 1
                  GOTO 190
                ENDIF
                GOTO 235
              ENDIF
230         CONTINUE
235         CONTINUE
            NE = NTEMP
            CALL LININT(ZOUT(L,KE),ZOUT(L,KE-1),YOUT(L,KE),YOUT(L,KE-1),
     &                  ZNAC(NE),YYE)
            ZZE = ZNAC(NE)
          ENDIF
C
C         Form the lower surface
C
C         Store wingroot grid
          DO K=1,NPAIR(1,KS)
            YWNG(K) = YOUT(L,K)
            ZWNG(K) = ZOUT(L,K)
          ENDDO
C         Store wingtip grid to a temp array
          IF(INTRAIL.EQ.2) THEN
            DO K=NPAIR(1,KE),NPH
              YINT(K-NPAIR(1,KE)+1) = YOUT(L,K)
              ZINT(K-NPAIR(1,KE)+1) = ZOUT(L,K)
            ENDDO
          ELSE
            DO K=KE,NPH
              YINT(K-KE+1) = YOUT(L,K)
              ZINT(K-KE+1) = ZOUT(L,K)
            ENDDO
          ENDIF
C         Get the nacelle grid under the wake line ready
          NADD = NE-NS+1 +2
          YWK(1) = YYS
          ZWK(1) = ZZS
          YWK(NADD) = YYE
          ZWK(NADD) = ZZE
          DO K=NS,NE
            YWK(K-NS+2) = YNAC(K)
            ZWK(K-NS+2) = ZNAC(K)
          ENDDO
          IF(INTRAIL.EQ.2) THEN
            NNACC = NPWNH-NPAIR(1,KS)-(NPH-NPAIR(1,KE)+1)
          ELSE
            NNACC = NPWNH-NPAIR(1,KS)-(NPH-KE+1)
          ENDIF
          CALL DISTARC(YWK,ZWK,NADD,YWK,ZWK,NNACC,-10,0)
C         Stick the nacelle grid under the wing
          DO K=1,NNACC
            ZWNG(NPAIR(1,KS)+K) = ZWK(K)
            YWNG(NPAIR(1,KS)+K) = YWK(K)
          ENDDO
C         Put back the wingtip grid into the wing
          IF(INTRAIL.EQ.2) THEN
            KREP = NPH-NPAIR(1,KE)+1
            DO K=1,KREP
              ZWNG(NPAIR(1,KS)+NNACC+K) = ZINT(K)
              YWNG(NPAIR(1,KS)+NNACC+K) = YINT(K)
            ENDDO
          ELSE
            KREP = NPH-KE+1
            DO K=1,KREP
              ZWNG(NPAIR(1,KS)+NNACC+K) = ZINT(K)
              YWNG(NPAIR(1,KS)+NNACC+K) = YINT(K)
            ENDDO
          ENDIF
          NWNACH = NPWNH
C
C         Form the upper surface
C
C         Store the wingtip grid
          IF(INTRAIL.EQ.2) THEN
            DO K=NPH,NPAIR(2,KE)
              YWNG(NWNACH+K-NPH) = YOUT(L,K)
              ZWNG(NWNACH+K-NPH) = ZOUT(L,K)
            ENDDO
            N1 = NWNACH+NPAIR(2,KE)-NPH
          ELSE
            DO K=NPH,NPTS
              IF(ZOUT(L,K).GT.ZZE) THEN
                KN1 = K
                GOTO 260
              ENDIF
            ENDDO
260         CONTINUE
            DO K=NPH,KN1
              YWNG(NWNACH+K-NPH) = YOUT(L,K)
              ZWNG(NWNACH+K-NPH) = ZOUT(L,K)
            ENDDO
            N1 = NWNACH+KN1-NPH
```

LINE # SOURCE TEXT

```
            ENDIF
C           Store the wingroot grid to a temp array
            DO K=NPAIR(2,KS),NPTS
              YINT(K-NPAIR(2,KS)+1) = YOUT(L,K)
              ZINT(K-NPAIR(2,KS)+1) = ZOUT(L,K)
            ENDDO
C           Get the nacelle grid above the wake line ready
            NADD = NNACP-NE + NS-1 +2
            YWK(1) = YYE
            ZWK(1) = ZZE
            DO K=NE+1,NNACP
              YWK(K-NE+1) = YNAC(K)
              ZWK(K-NE+1) = ZNAC(K)
            ENDDO
            DO K=1,NS-1
              YWK(NNACP-NE+1+K) = YNAC(K)
              ZWK(NNACP-NE+1+K) = ZNAC(K)
            ENDDO
            YWK(NADD) = YYS
            ZWK(NADD) = ZZS
            CALL DISTARC(YWK,ZWK,NADD,YWK,ZWK,NNACC,-10,0)
C           Stick the nacelle grid above the wing
            DO K=1,NNACC
              ZWNG(N1+K) = ZWK(K)
              YWNG(N1+K) = YWK(K)
            ENDDO
C           Put back the wingroot grid into the wing
            KREP = NPTS-NPAIR(2,KS)+1
            DO K=1,KREP
              ZWNG(N1+NNACC+K) = ZINT(K)
              YWNG(N1+NNACC+K) = YINT(K)
            ENDDO
            GOTO  700
          ENDIF

C         We are before trailing edge, normal redistribution
 415      CONTINUE
C         Bottom part
C         Find out the points that being replaced by the nacelle
          NTEMP = 1
 430      CONTINUE
          DO 440 K=1,NPH
            IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
              KS = K-1
                r1 = abs(zout(L,KS)-zout(L,KS-1))
                r2 = abs(zout(L,KS)-ZNAC(NTEMP))
                if(r2.le.r1/8.) then
                  KS=KS-1
                endif
              IF(ABS(YOUT(L,KS)-YNAC(NTEMP)).LE.RADNAC/800.) THEN
                NTEMP = NTEMP + 1
                GOTO 430
              ENDIF
              CALL LININT(ZOUT(L,KS),ZOUT(L,KS+1),YOUT(L,KS),
     &                    YOUT(L,KS+1),ZNAC(NTEMP),YYS)
              ZZS = ZNAC(NTEMP)
              GOTO 445
            ENDIF
 440      CONTINUE
 445      CONTINUE
          NFRIS = NTEMP
C
          NTEMP = NNACP
 450      CONTINUE
          DO 460 K=KS+1,NPH
            IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
              KE = K
              IF(ABS(YOUT(L,KE)-YNAC(NTEMP)).LE.RADNAC/800.) THEN
                NTEMP = NTEMP - 1
                GOTO 450
              ENDIF
              CALL LININT(ZOUT(L,KE-1),ZOUT(L,KE),YOUT(L,KE-1),
     &                    YOUT(L,KE),ZNAC(NTEMP),YYE)
              ZZE = ZNAC(NTEMP)
              GOTO 465
            ENDIF
 460      CONTINUE
 465      CONTINUE
          NFRIE = NTEMP
C
C         Store wingroot grid
          DO 500 K=1,KS
            YWNG(K) = YOUT(L,K)
            ZWNG(K) = ZOUT(L,K)
 500      CONTINUE
C         Get the nacelle grid under the wake line ready
          N2 = NFRIE-NFRIS+1  +2
          YINT(1 ) = YYS
          ZINT(1 ) = ZZS
          YINT(N2) = YYE
          ZINT(N2) = ZZE
          DO 520 K=NFRIS,NFRIE
            YINT(K-NFRIS+2) = YNAC(K)
            ZINT(K-NFRIS+2) = ZNAC(K)
 520      CONTINUE
          NNACC = NPWNH-KS-(NPH-KE+1)
          CALL DISTARC(YINT,ZINT,N2,YWK,ZWK,NNACC,-10,0)
C         Stick the nacelle grid under the wing
          DO K=1,NNACC
            ZWNG(KS+K) = ZWK(K)
            YWNG(KS+K) = YWK(K)
          ENDDO
C         Put the wingtip grid
          DO 540 K=KE,NPH
            YWNG(KS+NNACC+K-KE+1) = YOUT(L,K)
            ZWNG(KS+NNACC+K-KE+1) = ZOUT(L,K)
 540      CONTINUE
C
C
C         Upper part
          DO 590 K=NPH,NPTS
            IF(ZOUT(L,K).LE.ZNAC(NFRIE)) THEN
              NN1 = K-1
              CALL LININT(ZOUT(L,NN1+1),ZOUT(L,NN1),YOUT(L,NN1+1),
     &                    YOUT(L,NN1),ZNAC(NFRIE),YY1)
              ZZ1 = ZNAC(NFRIE)
              GOTO 595
            ENDIF
 590      CONTINUE
 595      CONTINUE
```

LINE # SOURCE TEXT

```
        DO 600 K=NPH,NPTS
          IF(ZOUT(L,K).LE.ZNAC(NFRIS)) THEN
            NN2 = K
            CALL LININT(ZOUT(L,NN2),ZOUT(L,NN2-1),YOUT(L,NN2),
     &                  YOUT(L,NN2-1),ZNAC(NFRIS),YY2)
            ZZ2 = ZNAC(NFRIS)
            GOTO 605
          ENDIF
 600    CONTINUE
 605    CONTINUE
C       Store the wingtip grid
        DO 620 K=NPH,NN1
          YWNG(NPWNH+K-NPH) = YOUT(L,K)
          ZWNG(NPWNH+K-NPH) = ZOUT(L,K)
 620    CONTINUE
C       Redistibute the points above the nacelle
        N2 = NN2-NN1+1
        YWK(1 ) = YY1
        ZWK(1 ) = ZZ1
        YWK(N2) = YY2
        ZWK(N2) = ZZ2
        DO K=NN1+1,NN2-1
          YWK(K-NN1+1) = YOUT(L,K)
          ZWK(K-NN1+1) = ZOUT(L,K)
        ENDDO
        NNA = NPWNH-(NN1-NPH+1)-(NPTS-NN2+1)
        CALL DISTARC(YWK,ZWK,N2,YWK,ZWK,NNA,-10,0)
        NNT = NPWNH+(NN1-NPH)
        DO K=1,NNA
          YWNG(NNT+K) = YWK(K)
          ZWNG(NNT+K) = ZWK(K)
        ENDDO
C       Store the wingroot grid
        DO K=NN2,NPTS
           YWNG(NNT+NNA+K-NN2+1) = YOUT(L,K)
           ZWNG(NNT+NNA+K-NN2+1) = ZOUT(L,K)
        ENDDO
 700    CONTINUE
        IF(IPRNT.NE.0)THEN
           WRITE(IPRNT)(XOUT(L,1),K=1,NPWN),
     &                 (YWNG(K),K=1,NPWN),
     &                 (ZWNG(K),K=1,NPWN)
           CALL FLUSH (IPRNT)
        ENDIF

        DO 750 K=1,NPWN
          XOUT(L,K) = XOUT(L,1)
          YOUT(L,K) = YWNG(K)
          ZOUT(L,K) = ZWNG(K)
 750    CONTINUE
C
 800  CONTINUE
      IF(LNUM(1).EQ.LNUM(3) .AND. LNUM(2).EQ.LNUM(4)) THEN
        NPTS = NPTS - MC
      ENDIF
      RETURN
      END
```

LINE # SOURCE TEXT

```
*********************************************************************
      SUBROUTINE NACGRID
      include "sgrid.com"
c     This subroutine read the nacelle grid and combine it with the
c     wing grid.
c
      COMMON /ENG/  XENG(2,NPI),YENG(2,NPI,NPI),ZENG(2,NPI,NPI)
     &              ,MNAC(2),MNACP(2)
      COMMON /NACG/ XNAC(NPI),YNAC(NPI,NPI),ZNAC(NPI,NPI)
      COMMON /NDIM/ NNAC,NNACP,LNUM(4)

c     (XENG,YENG,ZENG)      Coordinate of engine
c     MNAC(*)               Number of stations in nacelle
c     MNACP(*)              Number of points in each station
c     (1,*,*)  inner nacelle, (2,*,*) outer nacelle
c
c-------------------------------------------------------------------
c     Read the nacelles geometry:
c
c     Inner nacelle geometry
      CALL NACIN
      XIN1 = XNAC(1)
      XIN2 = XNAC(NNAC)
      MNAC(1) = NNAC
      MNACP(1) = NNACP
      DO 100 L=1,NNAC
        XENG(1,L) = XNAC(L)
        DO 80  K=1,NNACP
          YENG(1,L,K) = YNAC(L,K)
          ZENG(1,L,K) = ZNAC(L,K)
 80     CONTINUE
100   CONTINUE
c
c     Outer nacelle geometry
      CALL NACIN
      XOUT1 = XNAC(1)
      XOUT2 = XNAC(NNAC)
      MNAC(2) = NNAC
      MNACP(2) = NNACP
      DO 200 L=1,NNAC
        XENG(2,L) = XNAC(L)
        DO 180  K=1,NNACP
          YENG(2,L,K) = YNAC(L,K)
          ZENG(2,L,K) = ZNAC(L,K)
180     CONTINUE
200   CONTINUE
c-------------------------------------------------------------------
c
c     In a case of 2 nacelles, three zone will be made.
c     NEWING :      Station(s) will be added to the wing at inlet and outlet
c                   of the nacelle
c
c     MOUNT :       Mount the nacelle under the wing and/or wake line
c
c     JN = 1        Inner nacelle
c          2        Outer nacelle
c
c
c     The first zone, only one nacelle appears
      WRITE(*,*)'zone 1'
      JN = 1
      CALL NEWING(LNUM,XIN1,XOUT1)
      CALL MOUNT(JN,LNUM,21)
      LNUM(3) = LNUM(1)
      LNUM(4) = LNUM(2)

c     The second zone consists two necelles appear
      WRITE(*,*)'zone 2'
      JN = 1
      CALL NEWING(LNUM,XOUT1,XIN2)
      LNUM(1) = LNUM(1) + 1
      CALL MOUNT(JN,LNUM,0)
      LNUM(3) = LNUM(1)
      LNUM(4) = LNUM(2)
c
      WRITE(*,*)'zone 2'
      CALL NEWING(LNUM,XOUT1,XIN2)
      LNUM(1) = LNUM(1) + 1
      JN = 2
      CALL MOUNT(JN,LNUM,22)
      LNUM(3) = LNUM(1)
      LNUM(4) = LNUM(2)

c     The third zone, only one nacelle appears
      WRITE(*,*)'zone 3'
      JN = 2
      CALL NEWING(LNUM,XIN2,XOUT2)
      LNUM(1) = LNUM(1) + 1
      CALL MOUNT(JN,LNUM,23)
c
      RETURN
      END
```

LINE # SOURCE TEXT

```
C***********************************************************************
      SUBROUTINE NEWING(LNUM,XX1,XX2)
      include "sgrid.com"
      DIMENSION XWNG(NPI,NPI),YWNG(NPI,NPI),ZWNG(NPI,NPI)
      DIMENSION LNUM(4)


C     Rewrite the coordinate of the wing
      DO 10 L=1,NSEC
      DO 10 K=1,NPTS
        XWNG(L,K) = XOUT(L,K)
        YWNG(L,K) = YOUT(L,K)
        ZWNG(L,K) = ZOUT(L,K)
 10   CONTINUE

C     Find out where the x-location of start and end of nacelle
      XNSTRT = XX1
      XNEND  = XX2

C     Add two stations in the wing, these two stations lie exactly on
C     XNSTRT and XNEND.
      DO 55 NN=1,2
        IF(NN.EQ.1) XX=XNSTRT
        IF(NN.EQ.2) XX=XNEND
        DO 50 LW=1,NSEC
C         The wing section is very close to nacelle's station (XNSTRT or XNEND)
          IF(ABS(XWNG(LW,1)-XX).LE.1.E-7) THEN
            DO K=1,NPTS
              XWNG(LN,K) = XX
            ENDDO
            LNUM(NN) = LW
            GOTO 55
          ENDIF
C         Create an extra station in the wing
          IF(XWNG(LW,1).GT.XX) THEN
            X(LW)    = XX
            LNUM(NN) = LW
            DO K=1,NPTS
              X1 = XWNG(LW-1,K)
              Y1 = YWNG(LW-1,K)
              Z1 = ZWNG(LW-1,K)
              X2 = XWNG(LW  ,K)
              Y2 = YWNG(LW  ,K)
              Z2 = ZWNG(LW  ,K)
              CALL LININT(X1,X2,Y1,Y2,XX,YY)
              CALL LININT(X1,X2,Z1,Z2,XX,ZZ)
              Y(LW,K) = YY
              Z(LW,K) = ZZ
            ENDDO
C           If we are in wake, make sure top pts intersect the bottom pts
            DO 24 K1=2,(NPTS+1)/2
            DO 20 K2=(NPTS+1)/2+1,NPTS
              IF(ABS(Y(LW,K1)-Y(LW,K2)).LT.1.E-6 .AND.
     &           ABS(Z(LW,K1)-Z(LW,K2)).LT.1.E-6 )  THEN
                 Y(LW,K1) = Y(LW,K2)
                 Z(LW,K1) = Z(LW,K2)
                 GOTO 24
              ENDIF
 20         CONTINUE
 24         CONTINUE
C           Put the rest of the station into (X,Y,Z)
            DO 35 L=LW,NSEC
              DO 30 K=1,NPTS
                X(L+1) = XWNG(L,K)
                Y(L+1,K) = YWNG(L,K)
                Z(L+1,K) = ZWNG(L,K)
 30           CONTINUE
 35         CONTINUE
            NSEC = NSEC + 1
            DO 45 L=LW,NSEC
              DO 40 K=1,NPTS
                XWNG(L,K) = X(L)
                YWNG(L,K) = Y(L,K)
                ZWNG(L,K) = Z(L,K)
 40           CONTINUE
 45         CONTINUE
            GOTO 55
          ENDIF
 50     CONTINUE
 55   CONTINUE

      DO 910 L=1,NSEC
      DO 910 K=1,NPTS
        XOUT(L,K) = XWNG(L,K)
        YOUT(L,K) = YWNG(L,K)
        ZOUT(L,K) = ZWNG(L,K)
910   CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
 ***********************************************************************
       SUBROUTINE NOSE(FNBOT,FNTOP)
       include "sgrid.com"
       DIMENSION D1(NPI),D2(NPI),S(NPI)

       KTIP = (KDIM+1)/2

C      From the nose to the leading edge
C      ie., from station 1 to M1
C
C      First point of the nose
       DO 12 K=1,KDIM
         X(1) = XIN(1)
         Y(1,K) = YIN(1,1)
         Z(1,K) = ZIN(1,1)
  12   CONTINUE
C
C
C      Loop for all stations, from station 2 to station M1
       DO 500 M=2,M1
         L = M
         XLOCAL = XIN(M)
C        Store the input to dummy arry
         DO 33 K=1,NFP
           YINT(K) = YIN(M,K)
           ZINT(K) = ZIN(M,K)
  33     CONTINUE
C
C        YREF is value of the first point of the wing.
C        KREF is the corresponding index of each station.
         YREF=YOUT(1,NPTS)
C-----------------------------------------------------------------------
C OPTIONS :-
C
C        Boeing Baseline Configuration
         DO 43 K=1,NFP
           IF(YINT(K).GE.YREF) THEN
             KREF=K
             GOTO 44
           ENDIF
  43     CONTINUE
  44     CONTINUE
         KREF=NFP/2 +KREFADD
C-----------------------------------------------------------------------
C        Langley's Low-Boom Configuration
c          KREF = 31
C-----------------------------------------------------------------------
C
C        Lower part of the nose (-Y to Y=YREF)
         KS=1
         KE=KREF
         KN=KE-KS+1
         DO 74 K= KS,KE
           KK = K-KS+1
           D1(KK) = YINT(K)
           D2(KK) = ZINT(K)
  74     CONTINUE
         CALL DISTARC(D1,D2,KN,YNEW,ZDIST,KTIP,FNBOT,1)
         DO 85 K=1,KTIP
           Y(L,K) = YNEW(K)
           Z(L,K) = ZDIST(K)
           X(L) = XLOCAL
  85     CONTINUE
C
C        From Y=YREF to pos Y
         KS=KREF
         KE=NFP
         KN=KE-KS+1
         DO 100 K= KS,KE
           KK = K-KS+1
           D1(KK) = YINT(K)
           D2(KK) = ZINT(K)
 100     CONTINUE
         CALL DISTARC(D1,D2,KN,YNEW,ZDIST,KTIP,FNTOP,0)
         DO 400 K=KTIP,KDIM
           Y(L,K) = YNEW(K-KTIP+1)
           Z(L,K)= ZDIST(K-KTIP+1)
           X(L)  = XLOCAL
 400     CONTINUE
 500   CONTINUE
       RETURN
       END
```

SOURCE TEXT

LINE #

```
****************************************************************************
      SUBROUTINE REDIST(XLOCAL,KTIP,FAC,IFLAT)
      include "sgrid.com"
c
c     This is the main subroutine to redistributethe points from
c     spanwise cut to streamwise cut.
c     When there is "cheung-SIMP", it means the output is for SIMP code
c
      PARAMETER(INT=400)
      DIMENSION S(INT)
c
c     NU = number of point in the upper surface
c     NL = number of point in the lower surface
c     NC = number of spanwise sections
c     XLE = X leading edge
c     (ZINT,YINT) = point of streamwise cut for X=XLOCAL at each surface
c     KT = # of pts in circum. direction extracted from the old grid.
c     KTIP = # of pts in the circum. direction in one surface
c
c     IFLAT = 2 : upper surface
c     IFLAT = 1 : lower surface
c
      IF(INT.LE.KTIP) THEN
        WRITE(*,*)'SUB REDIST : INT is less than KTIP'
        STOP
      ENDIF

      IF(IFLAT.EQ.2) THEN
        NUL = NU
      ELSE
        NUL = NL
      ENDIF
c
c
c     The streamwise distance passes the leading edge
      IF(XLOCAL.GT.XLE(NC))THEN
c
c       X-station is at the wing tip
        IF(XLOCAL.LE.(XLE(NC)+CHORD(NC)) )THEN
c
c         Should call WINGNWAKE if wake is contained in this station
          IF(XLOCAL.GT.XLE(1)+CHORD(1)) THEN
            CALL WINGNWAKE(XLOCAL,KT,NUL,IFLAT)
            GOTO 200
          ENDIF
c
c         Streamwise distance is between the leading edge and the trailing edge
          DO 60 M=1,NC
          DO 50 I=2,NUL
            IF(XBASE(I,IFLAT,NC-M+1).GE.XLOCAL .OR.
     &         ABS(XBASE(I,IFLAT,NC-M+1)-XLOCAL).LE.1.E-7) THEN
              X1 = XBASE(I-1,IFLAT,NC-M+1)
              X2 = XBASE(I  ,IFLAT,NC-M+1)
              Y1 = YBASE(I-1,IFLAT,NC-M+1)
              Y2 = YBASE(I  ,IFLAT,NC-M+1)
              Z1 = ZBASE(I-1,IFLAT,NC-M+1)
              Z2 = ZBASE(I  ,IFLAT,NC-M+1)
              CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
              CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
              YINT(M) = YY
              ZINT(M) = ZZ
              GOTO 60
            ENDIF
 50       CONTINUE
 60       CONTINUE
          KT=NC
          GOTO200
c
        ELSE
c
c         The X-station passes the wing tip, should have wake
          CALL WINGNWAKE(XLOCAL,KT,NUL,IFLAT)
          GOTO 200
        ENDIF
      ENDIF
c
c     Streamwise distance is in the leading edge
      IF(XLOCAL.LE.XLE(NC)) THEN
c
c       Should call WINGNWAKE if wake is contained in this station
        IF(XLOCAL.GT.XLE(1)+CHORD(1)) THEN
          CALL WINGNWAKE(XLOCAL,KT,NUL,IFLAT)
          GOTO 200
        ENDIF
c
        KT = 0
        DO 160 I = 1,NUL
         KT = KT + 1
c        Create a point at the root
         IF(XBASE(I,IFLAT,1).GT.XLOCAL) THEN
           X1 = XBASE(I-1,IFLAT,1)
           X2 = XBASE(I  ,IFLAT,1)
           Y1 = YBASE(I-1,IFLAT,1)
           Y2 = YBASE(I,IFLAT,1)
           Z1 = ZBASE(I-1,IFLAT,1)
           Z2 = ZBASE(I  ,IFLAT,1)
           CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
           CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
           YINT(KT) = YY
           ZINT(KT) = ZZ
           GOTO 200
         ENDIF
         DO 150 M = 2,NC
          XL = XLOCAL-XBASE(I,IFLAT,M-1)
          XR = XLOCAL-XBASE(I,IFLAT,M)
          IF(XL*XR.LT.0.) THEN
            X1 = XBASE(I,IFLAT,M-1)
            X2 = XBASE(I,IFLAT,M)
            Z1 = ZBASE(I,IFLAT,M-1)
            Z2 = ZBASE(I,IFLAT,M)
            Y1 = YBASE(I,IFLAT,M-1)
            Y2 = YBASE(I,IFLAT,M)
            CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
            CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
            YINT(KT) = YY
            ZINT(KT) = ZZ
            GOTO 160
          ELSEIF (ABS(XL*XR) .LE. 1.E-7) THEN
            IF(ABS(XL).LE.1.E-7) THEN
```

LINE # SOURCE TEXT

```
              YINT(KT) = YBASE(I,IFLAT,M-1)
              ZINT(KT) = ZBASE(I,IFLAT,M-1)
            ELSE
              YINT(KT) = YBASE(I,IFLAT,M)
              ZINT(KT) = ZBASE(I,IFLAT,M)
            ENDIF
            GOTO 160
          ENDIF
 150     CONTINUE
 160    CONTINUE
       ENDIF
C
 200   CONTINUE
C
C      Right now YINT and ZINT is from wing tip to the root, in order to
C      the cubic spline program, need from root to tip.
C
       DO 220 KK=1,KT
         S(KK) = YINT(KK)
  220  CONTINUE
       DO 230 KK=1,KT
         YINT(KK) = S(KT-KK+1)
  230  CONTINUE
       DO 240 KK=1,KT
         S(KK) = ZINT(KK)
  240  CONTINUE
       DO 250 KK=1,KT
         ZINT(KK) = S(KT-KK+1)
  250  CONTINUE
C
C OPTIONS :-
C      Distribute z coordinates (raps around direction) from root to
C      leading edge and back (zdist).
C      IT=1, grid points will custer near the wing tip, =0 near the root.
C      For WESS, IT=0  ;  for Boeing, IT=1
       IT=1
       CALL DISTARC(ZINT,YINT,KT,ZDIST,YNEW,KTIP,FAC,IT)
       IF(L.NE.NSEC)CALL CSPLINE(ZINT,YINT,KT,ZDIST,YNEW,KTIP)
C
       RETURN
       END
```

LINE # SOURCE TEXT

```
************************************************************************
      SUBROUTINE SUBTRACGRID(NPL1,X,Y,Z,NPI,NSEC,KDIM)
c
c     This subroutine allows us to substract a grid line NPL1 in
c     streamwise section, and the new dimension is NSEC again.
c
      DIMENSION X(NPI),Y(NPI,NPI),Z(NPI,NPI)
c
c     Renumber the late stations
      NSEC = NSEC-1
      DO 30 L= NPL1,NSEC
        X(L) = X(L+1)
        DO 20 K=1,KDIM
          Y(L,K) = Y(L+1,K)
          Z(L,K) = Z(L+1,K)
 20     CONTINUE
 30   CONTINUE

      RETURN
      END
```

LINE # SOURCE TEXT

```
*************************************************************************
      SUBROUTINE TAIL(FAC1,FAC2)
      include "sgrid.com"
      DIMENSION S(NPI)
      COMMON /REF/ ZROOT,KTIP,ARCORR

      KTIP = (KDIM+1)/2
      LE = L-1
c     The tail of the configuration
c     WAKEPT is the Y value that the wake is (ie Y value of ZROOT)
c
      PT1= (YOUT(NSEC,1)+YOUT(NSEC,NPTS))/2.
      PT2= (YIN(NF,NFP)+YIN(NF,1))/2.
      X1 = X(LE)
      X2 = XIN(NF)

      L1 = LE+1
      L2 = L1+(NF-M2)-1
      DO 300 L=L1,L2
        M=M2+L-L1+1
        X(L) = XIN(M)
        XLOCAL = XIN(M)
C OPTIONS :-
C-----------------------------------------------------------------
c        WAKEPT = (YIN(NF,NFP)+YIN(NF,1))/2.
C-----------------------------------------------------------------
        CALL LININT(X1,X2,PT1,PT2,XLOCAL,WAKEPT)
C-----------------------------------------------------------------
C
c       Design the 1st point (KF) will be the off fuselage side
c       It is also the number of pts in the upper or lower fuselage, therefore
c       it depends on the previous station.
c       (no. of pt in this station)/(no. of pts in previous) =
c       (radius of this station)  /(radius of previous)
        RATIO=ABS(YIN(M,1)-YIN(M,NFP))/ABS(YIN(M-1,1)-YIN(M-1,NFP))
        KF=(KDIM-NPTS)/2+1
        KF=(KDIM-NPTS)/2
*        IF(RATIO.LE.0.9 .OR. RATIO.GE.1.1) THEN
*        KF=IFIX(RATIO*FLOAT(KF))
*        ENDIF
c
c
c       Find KT1 the point at old grid where YIN(M,KT1)=WAKEPT
c       and calculate the points from neg Y to Y at KT1
        DO 222 K=1,NFP
          IF(ABS(YIN(M,K)-WAKEPT).LE.1.E-7) THEN
            KT1=K
            GOTO 223
          ENDIF
          IF(YIN(M,K).GT.WAKEPT) THEN
            KT1=K
            Y1=YIN(M,K-1)
            Y2=YIN(M,K)
            Z1=ZIN(M,K-1)
            Z2=ZIN(M,K)
            YY=WAKEPT
            CALL LININT(Y1,Y2,Z1,Z1,YY,ZZ)
            YIN(M,KT1)=YY
            ZIN(M,KT1)=ZZ
            GOTO 223
          ENDIF
 222    CONTINUE
 223    CONTINUE

        DO 230 K=1,KT1
          YINT(K)=YIN(M,K)
          ZINT(K)=ZIN(M,K)
 230    CONTINUE
        CALL DISTARC(YINT,ZINT,KT1,YNEW,ZDIST,KF,FAC1,1)
        DO 240 K=1,KF
          Y(L,K)=YNEW(K)
          Z(L,K)=ZDIST(K)
 240    CONTINUE
c
c
c       Find KT2 the point at old grid where YIN(M,KT2)=WAKEPT
c       note : KT2=NFP-KT1+1
c       and calculate the points from neg Y to Y at KT1
        KT2=NFP-KT1+1
        DO 250 K=1,KT2
          YINT(K)=YIN(M,KT1+K-1)
          ZINT(K)=ZIN(M,KT1+K-1)
 250    CONTINUE
        CALL DISTARC(YINT,ZINT,KT2,YNEW,ZDIST,KF,FAC1,0)
        DO 260 K=1,KF
          Y(L,KDIM-KF+K)=YNEW(K)
          Z(L,KDIM-KF+K)=ZDIST(K)
 260    CONTINUE
c
c
c       These are points in off fuselage side
c       KN=No. of pts in the off-fuselage side
        KN=KDIM-2*KF
        KNH=(KN+1)/2
        RSPAN= ABS( Z(LE,KTIP)-Z(L,KF) )
        IF(FAC2.LT.0.)FAC2=1.E+15
        DELT=FAC2*(RSPAN)/FLOAT(KNH)
        CALL DISTRI(DELT,KNH+1,S,0)
c
        CALL LININT(X1,X2,YOUT(NSEC,NPTS/2),PT2,XLOCAL,YOB)
c
        DO 270 K=1,KNH
          Z(L,KF+K)=Z(L,KF)+S(K+1)*RSPAN
          CALL LININT(ZROOT,RSPAN,WAKEPT,YOB,Z(L,KF+K),YY)
          Y(L,KF+K)=YY
 270    CONTINUE
        DO 280 K=1,KNH
          Z(L,KDIM-KF-K+1)=Z(L,KF+K)
          Y(L,KDIM-KF-K+1)=Y(L,KF+K)
 280    CONTINUE

c       Smooth the fuselage-wake part
        DO 285 KK=1,KDIM
          YINT(KK)=Y(L,KK)
          ZINT(KK)=Z(L,KK)
 285    CONTINUE
        ZROOT = ZIN(M,KT2)
        RFILT = RFIL
        IF(X(L).GE.XSTART .AND. X(L).LE.XOFF)
     &    CALL FILET(YINT,ZINT,KDIM,MB1,MB2,MT1,MT2,RFILT)
```

LINE # SOURCE TEXT

```
          DO 290 K=1,KDIM
            Y(L,K)=YINT(K)
            Z(L,K)=ZINT(K)
290       CONTINUE
cheung-SIMP
c          NPH=(NFP+1)/2
c          DO 293 K=1,NPH
c            YINT(K)=YIN(M,K)
c            ZINT(K)=ZIN(M,K)
c293       CONTINUE
c          CALL DISTARC(YINT,ZINT,NPH,YNEW,ZDIST,KTIP,0.05,1)
c          DO 295 K=1,KTIP
c            Z(L,K)=ZDIST(K)
c            Y(L,K)=YNEW(K)
c295       CONTINUE
c          DO 296 K=1,NPH
c            YINT(K)=YIN(M,NPH+K-1)
c            ZINT(K)=ZIN(M,NPH+K-1)
c296       CONTINUE
c         CALL DISTARC(YINT,ZINT,NPH,YNEW,ZDIST,KTIP,0.05,0)
c          DO 297 K=1,KTIP
c            Z(L,KTIP+K-1)=ZDIST(K)
c            Y(L,KTIP+K-1)=YNEW(K)
c297       CONTINUE
cheung-SIMP
300     CONTINUE
c       This allows the tail has equal thickness
c        DO 313 L=L1+1,L2
c          DO 312 K=1,KDIM
c            Z(L,K) = Z(L-1,K)
c            Y(L,K) = Y(L-1,K)
c 312      CONTINUE
c 313    CONTINUE
C
        RETURN
        END
```

LINE # SOURCE TEXT

```
***********************************************************************
      SUBROUTINE WBGRID
      include "sgrid.com"
c     This subroutine read the fuelage grid and combine it with the
c     wing grid.
c
      DIMENSION D1(NPI),D2(NPI),S(NPI)
      COMMON /REF/ ZROOT,KTIP,ARCORR
c
c     Read surfgrid dimensions
c
c     XIN,YIN,ZIN = read in grid
c     X,Y,Z = output grid
c     YINT,ZINT,D1,D2 = dummy vectors
c     KDIM = # of points of the whole body in the warp-around direction
c     FNBOT= FGS in the nose bottom part (circumferencial direction)
c     FNTOP= FGS in the nose top    part (circumferencial direction)
c     FAC2 = FGS in the tail part (circumferencial direction)
c     RFIL = Filet radius
c     ARCORR = This make filet looks smooth
c     BLWAKE = Distance of the trailing wake in body length
c     BL     = Body length
c     NWAKE = Number points along the trailing wake
c     SCALE = Scale factor for the wing-body
      NAMELIST /BODY/ KDIM,FNBOT,FNTOP,FAC2,
     &                BLWAKE,NWAKE,CUSTERON,SCALE
     &         /FIL/ RFIL,ARCORR,MB1,MB2,MT1,MT2,XSTART,XOFF
     &         /OPTN/ KREFADD,K1ADD
      READ(40,BODY)
      READ(40,FIL)
      READ(40,OPTN)
      WRITE(*,BODY)
      WRITE(*,FIL)
      WRITE(*,OPTN)

c
c     Read the fuselage geometry
c-----------------------------------------------------------------------
      CALL FUSEIN
c-----------------------------------------------------------------------
c
c     Take this initial grid
c       1) convert it into area distribution
c       2) add shape function
c       3) convert it back to grid
c
c      CALL FSHAPE
c
      BL   = XIN(NF)
      XCUS = XCUS*BL
      KTIP = (KDIM+1)/2
c
c     Find number of streamwise station.  We keep the stations
c     of the fuselage.  As to the common area of the wing and fuselage
c     we use the stations of the wing.
c     M1 is the station the nose ends
c     M2 is the station the wing ends
      M1=0
      M2=0
      DO 10 M = 1,NF
        IF(ABS(XIN(M)-XOUT(1,1)).LE.1.E-7) M1 = M
        IF(ABS(XIN(M)-XOUT(NSEC,1)).LE.1.E-7) M2 = M+1
 10   CONTINUE
      IF(M1.EQ.0) THEN
        DO 20 M=2,NF
          IF(XIN(M-1).LE.XOUT(1,1) .AND. XIN(M).GT.XOUT(1,1))
     &      M1=M
 20     CONTINUE
      ENDIF
      IF(M2.EQ.0) THEN
        DO 30 M=1,NF
          IF(XIN(M).LE.XOUT(NSEC,1) .AND. XIN(M+1).GT.XOUT(NSEC,1))
     &      M2=M+1
 30     CONTINUE
      ENDIF
c
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     We divid the configuration into four parts :
c     the nose, the wing-body, the tail, and the extended wake.  And
c     for each part we distribute points for the lower and upper part
c     separately.
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c****************************************
c
c     From the nose to the leading edge
      CALL NOSE(FNBOT,FNTOP)
c
c********************************************************
c
c     From the leading edge to the trailing edge (the whole wing)
c      CALL WFMATCH
      CALL WING_BODY

c
c********************************************************
c
c     From the trailing edge to the end of tail
      CALL TAIL(-1.,FAC2)

c
c********************************************************
c
c     Redistribute each streamwise station, so that the grid
c     is custer near the wing tip.
      IF(CUSTERON.GT.0.)  CALL CUSTER
c
c********************************************************
c
c     Redistribute the streamwise stations, so that
c     it is equally spaced.
       IF(CUSTERON.GT.0.) CALL EQSPACE
c
c********************************************************
c     Add the extended wake
c
      L2 = L-1
      XLAST = X(L2) + BL*BLWAKE
      TOTLEN = XLAST - X(L2)
```

LINE # | SOURCE TEXT

```
      IF(TOTLEN.EQ.0.) GOTO 510
      DELT=(X(L2)-X(L2-1))/TOTLEN
      CALL DISTRI(DELT,NWAKE+1,S,0)
      DO 500 LD=1,NWAKE
       L = L2+LD
        X(L) = X(L2)+S(LD+1)*TOTLEN
        DO 480 K=1,KDIM
          Z(L,K) = Z(L-1,K)
          Y(L,K) = Y(L-1,K)
480     CONTINUE
500   CONTINUE
510   CONTINUE
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
c     We might like to scale the whole thing by a reference length
c
      KW=1
      NSECT=L2+NWAKE
      DO 550 L=1,NSECT
        X(L)=X(L)/SCALE
        DO 550 K=1,KDIM
          Y(L,K)=Y(L,K)/SCALE
          Z(L,K)=Z(L,K)/SCALE
550   CONTINUE
c
c------------------------------------------------------------------------
c OPTIONS :-
c
c     Boeing Baseline Configuration
c     Before write out the grid, this boeing wing needed to be fixed
c     for some stupid reasons.
c      DO 570 K=1,NSECT
c        Y(12,K)=Y(11,K)
c        Z(12,K)=Z(11,K)
c570    CONTINUE
c      CALL SUBTRACGRID(31,X,Y,Z,NPI,NSECT,KDIM)
c      CALL ADDGRID(19,20,X,Y,Z,NPI,NSECT,KDIM)
c------------------------------------------------------------------------
c     Langley's Low-Boom Configuration
c      CALL ADDGRID(20,21,X,Y,Z,NPI,NSECT,KDIM)
c------------------------------------------------------------------------
c
c
c     Write out new surfgrid in plot3d format
c
      WRITE(60)KDIM,KW,NSECT
      DO 600 L=1,NSECT
        WRITE(60)(X(L),K=1,KDIM),
     .            (Y(L,K),K=1,KDIM),
     .            (Z(L,K),K=1,KDIM)
600   CONTINUE
      CALL FLUSH (60)
c
c     Write the wing-body grid (X,Y,Z) to (XOUT,YOUT,ZOUT)
c
      NPTS = KDIM
      NSEC = NSECT
      DO 610 L=1,NSECT
      DO 610 K=1,KDIM
        XOUT(L,K) = X(L)
        YOUT(L,K) = Y(L,K)
        ZOUT(L,K) = Z(L,K)
610   CONTINUE

      RETURN
      END
```

SOURCE TEXT

```
***********************************************************************
      SUBROUTINE WFMATCH
      include "sgrid.com"
      DIMENSION D1(NPI),D2(NPI)
      COMMON /REF/ ZROOT,KTIP,ARCORR
c
c     This subroutine moves the whole inward or outward such that the
c     fuselage and the wing-root match.
c
c
c     The wing-body section, itself has NSEC sections
c     LW is the section index for YIN,ZIN
      DZMAX = 0.
      DZMIN = 1.E+20
      LB = M1+1
      LE = M1+(NSEC-1)
      DO 200 L=LB,LE
        LW=L-M1+1
        XLOCAL = XOUT(LW,1)
c
c       Calculate the points (YINT,ZINT) on the fuselage at XLOCAL
c       Note: assumed that in this section, each station has same number
c             of points in rape-around direction.
        DO 10 M=M1,NF
          IF(XIN(M).GE.XLOCAL .OR.
     &       ABS(XIN(M)-XLOCAL).LE.1.E-7) THEN
            MW = M
            GOTO 15
          ENDIF
 10     CONTINUE
 15     CONTINUE
        DO 30 K=1,NFP
          X1=XIN(MW-1)
          X2=XIN(MW)
          Y1=YIN(MW-1,K)
          Y2=YIN(MW,K)
          Z1=ZIN(MW-1,K)
          Z2=ZIN(MW,K)
          CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
          CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
          ZINT(K) = ZZ
          YINT(K) = YY
 30     CONTINUE
 35     CONTINUE

c       Move the wing in the z-direction (spanwise) to make sure
c       the wing is not inside or outside the fuselage
        ZROOT = ZINT((NFP+1)/2)
        DO KZ1 = 1,NFP
          IF(ZINT(KZ1).GT.ZROOT) ZROOT=ZINT(KZ1)
        ENDDO
        DZ = ZOUT(LW,1)-ZROOT
        IF(ABS(DZ).GT.ABS(DZMAX)) DZMAX=DZ
        IF(ABS(DZ).LT.ABS(DZMIN)) DZMIN=DZ
 200  CONTINUE
      IF(DZMAX.GT.0.) DZ=DZMIN
      IF(DZMAX.LT.0.) DZ=DZMAX
      DO 400 K=1,NPTS
      DO 400 L=1,NSEC
       ZOUT(L,K) = ZOUT(L,K) - DZ
 400  CONTINUE
          write(*,*)'DZ =', DZ
      RETURN
      END
```

LINE # SOURCE TEXT

```
***********************************************************************
      SUBROUTINE WING_BODY
      include "sgrid.com"
      DIMENSION D1(NPI),D2(NPI)
      COMMON /REF/ ZROOT,KTIP,ARCORR

c
c     The wing-body section, itself has NSEC sections
c     LW is the section index for YIN,ZIN
c
      LB = M1+1
      LE = M1 + NSEC -1
      DO 200 L=LB,LE
        LW=L-M1+1     !start with second wing station
        XLOCAL = XOUT(LW,1)
c
c       Calculate the points (YINT,ZINT) on the fuselage at XLOCAL
c       Note: assumed that in this section, each station has same number
c             of points in rape-around direction.
        DO 10 M=M1,NF
          IF(XIN(M).GE.XLOCAL .OR.
     &       ABS(XIN(M)-XLOCAL).LE.1.E-7) THEN
            MW = M
            GOTO 15
          ENDIF
 10     CONTINUE
 15     CONTINUE
        DO 30 K=1,NFP
          X1=XIN(MW-1)
          X2=XIN(MW)
          Y1=YIN(MW-1,K)
          Y2=YIN(MW,K)
          Z1=ZIN(MW-1,K)
          Z2=ZIN(MW,K)
          CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
          CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
          ZINT(K) = ZZ
          YINT(K) = YY
 30     CONTINUE
 35     CONTINUE
c
c
c       There are NPTS points in the wing area, need to check out
c       how many point is in the fuselage section.
c       K1 is the inters  pt ./. the fuselage & wing at bottom in old grid
c       K2 is the inters  pt ./. the fuselage & wing at top in old grid
c       MID is the inter  point ./. the fuselage & wing at bottom in new grid
c       There are KT points from 1 to K1
        KT=(NFP+1)/2
        MID=(KDIM-NPTS+2)/2
        MID=(KDIM-NPTS)/2
c
c       Find K1
        DO 60 K=KT,1,-1
          IF(YINT(K).LT.YOUT(LW,1) .AND. YINT(K).NE.YINT(K-1)) THEN
            K1 = K
            GOTO 62
          ENDIF
 60     CONTINUE
 62     CONTINUE
c
c       Find K2
        DO 70 K=1,NFP
          IF(YINT(K).GT.YOUT(LW,NPTS).AND.YINT(K).NE.YINT(K+1)) THEN
            K2 = K
            GOTO 72
          ENDIF
 70     CONTINUE
 72     CONTINUE
c       For arrow-wing type, K2 needed to be relocated
        IF(ABS(YOUT(LW,1)-YOUT(LW,NPTS)).LE.1.E-7) THEN
             K1 = K1 + K1ADD
             K2 = K1
             YINT(K2)=YINT(K1)
             ZINT(K2)=ZINT(K1)
        ENDIF
c
c       Calculate the points at the bottom (from neg. Y to Y at MID)
        KS=1
        KE= K1
        KN=KE-KS+1
        DO 82 K= KS,KE
          KK = K-KS+1
          D1(KK) = YINT(K)
          D2(KK) = ZINT(K)
 82     CONTINUE
        CALL DISTARC(D1,D2,KN,YNEW,ZDIST,MID,-10.,1)
        DO 100 K=1,MID
          Y(L,K) = YNEW(K)
          Z(L,K) = ZDIST(K)
          X(L) = XLOCAL
 100    CONTINUE
c
c       The points of the wing section
        DO 110 K=1,NPTS
          Y(L,MID+K) = YOUT(LW,K)
          Z(L,MID+K) = ZOUT(LW,K)
          X(L) = XOUT(LW,K)
 110    CONTINUE
c
c       Calculate the points at the top
        KS=K2
        KE= NFP
        KN=KE-KS+1
        DO 115 K= KS,KE
          KK = K-KS+1
          D1(KK) = YINT(K)
          D2(KK) = ZINT(K)
 115     CONTINUE
        CALL DISTARC(D1,D2,KN,YNEW,ZDIST,MID,-10.,0)
        DO 120 K=1,MID
          Y(L,K+MID+NPTS) = YNEW(K)
          Z(L,K+MID+NPTS) = ZDIST(K)
          X(L) = XLOCAL
 120    CONTINUE
c
c       For arrow-wing type, make sure wake points ok
        IF(ABS(YOUT(LW,1)-YOUT(LW,NPTS)).LE.1.E-7) THEN
          Y(L,MID+NPTS+1) = YOUT(LW,1)
          Y(L,MID) = Y(L,MID+NPTS+1)
```

LINE # SOURCE TEXT

```
            Z(L,MID) = Z(L,MID+NPTS+1)
          ENDIF
C
C
C     Fill the unsmooth part by FILET
C     First of all, find the set of points needed to be rearrange
        DO 125 KK=1,KDIM
          YINT(KK)=Y(L,KK)
          ZINT(KK)=Z(L,KK)
125     CONTINUE
        RFILT = RFIL
        IF(X(L).GE.XSTART .AND. X(L).LE.XOFF)
     &    CALL FILET(YINT,ZINT,KDIM,MB1,MB2,MT1,MT2,RFILT)
        DO 140 K=1,KDIM
          Y(L,K)=YINT(K)
          Z(L,K)=ZINT(K)
140     CONTINUE
200   CONTINUE
      RETURN
      END
```

LINE # | SOURCE TEXT

```
*********************************************************************
      SUBROUTINE WINGNWAKE(XLOCAL,KT,NUL,IFLAT)
      include "sgrid.com"
c     This subroutine creates the data when the station is in the place
c     where some points are on the wing, some are the wake.
c     At that x-station, we put 10 points in the wake part.
c
      KT = 0.
      IF(ARRWING.GT.0.) THEN
c         The wing is sweeped backwards
          DO 75 J=2,NC
            J1=J-1
            J2=J
            IF(XLOCAL.GT.XLE(J)+CHORD(J)) GOTO 75
            XI1=XLE(J1)+CHORD(J1)
            Z1=ZBASE(NUL,IFLAT,J1)
            XI2=XLE(J2)+CHORD(J2)
            Z2=ZBASE(NUL,IFLAT,J2)
            ZTIP=Z1+((XLOCAL-XI1)/(XI2-XI1))*(Z2-Z1)
            ZFLAT=ZTIP
c
            NLEDG = NC
            DO MRUN=2,NC
              IF(XLOCAL.LT.XBASE(1,IFLAT,MRUN)) THEN
c               If XLOCAL < leading edge, we should do the following
                NLEDG = MRUN-1
                X1 = XBASE(1,IFLAT,MRUN-1)
                X2 = XBASE(1,IFLAT,MRUN  )
                Y1 = YBASE(1,IFLAT,MRUN-1)
                Y2 = YBASE(1,IFLAT,MRUN  )
                Z1 = ZBASE(1,IFLAT,MRUN-1)
                Z2 = ZBASE(1,IFLAT,MRUN  )
                CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
                CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
                YINT(1) = YY
                ZINT(1) = ZZ
                GOTO 40
              ENDIF
            ENDDO
 40         CONTINUE

            DO 65 M=NLEDG,J2,-1
            DO 64 I=2,NUL
              IF(XLOCAL.LE.XBASE(I,IFLAT,M)) THEN
                X1 = XBASE(I-1,IFLAT,M)
                X2 = XBASE(I  ,IFLAT,M)
                Y1 = YBASE(I-1,IFLAT,M)
                Y2 = YBASE(I  ,IFLAT,M)
                Z1 = ZBASE(I-1,IFLAT,M)
                Z2 = ZBASE(I  ,IFLAT,M)
                CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
                CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
                IF(NLEDG.EQ.NC) THEN
                  MM=NLEDG-M+1
                ELSE
                  MM=NLEDG-M+2
                ENDIF
                YINT(MM) = YY
                ZINT(MM) = ZZ
                GOTO 65
              ENDIF
 64         CONTINUE
 65         CONTINUE
c
c           Add 10 points for the wake.
            ZROOT = ZBASE(NUL,IFLAT,1)
            Y1=YBASE(NUL,IFLAT,J1)
            Y2=YBASE(NUL,IFLAT,J2)
            Z1=ZBASE(NUL,IFLAT,J1)
            Z2=ZBASE(NUL,IFLAT,J2)
            CALL LININT(XI1,XI2,Y1,Y2,XLOCAL,YY)
            CALL LININT(XI1,XI2,Z1,Z2,XLOCAL,ZZ)
            DZ = (ZFLAT-ZROOT)/9.
            DO 70 M=1,10
              IF(NLEDG.EQ.NC) THEN
                MM = NLEDG-J2+1+M
              ELSE
                MM = NLEDG-J2+1+M    +1
              ENDIF
              ZINT(MM) = ZROOT + FLOAT(10-M)*DZ
              CALL LININT(ZROOT,ZZ,YOUT(L-1,1),YY,ZINT(MM),YYY)
              YINT(MM) = YYY
 70         CONTINUE
            IF(NLEDG.EQ.NC) THEN
              KT = (NLEDG-J2+1)+10
            ELSE
              KT = (NLEDG-J2+1    +1)+10
            ENDIF
            GOTO 200
 75       CONTINUE

c
      ELSE

c         The wing is sweeped forwards
          DO 105 J=2,NC
            J1=J-1
            J2=J
            IF(XLOCAL.LT.XLE(J)+CHORD(J)) GOTO 105
            XI1=XLE(J1)+CHORD(J1)
            Z1=ZBASE(NUL,IFLAT,J1)
            XI2=XLE(J2)+CHORD(J2)
            Z2=ZBASE(NUL,IFLAT,J2)
            ZTIP=Z1+((XLOCAL-XI1)/(XI2-XI1))*(Z2-Z1)
            ZFLAT=ZTIP
cheung-SIMP c       ZTIP=ZBASE(NUL,IFLAT,NC)
            ZTIP=ZBASE(NUL,IFLAT,NC)
cheung
c
c           Add 10 points for the wake.
            Y1=YBASE(NUL,IFLAT,J1)
            Y2=YBASE(NUL,IFLAT,J2)
            Z1=ZBASE(NUL,IFLAT,J1)
            Z2=ZBASE(NUL,IFLAT,J2)
            CALL LININT(XI1,XI2,Y1,Y2,XLOCAL,YY)
            CALL LININT(XI1,XI2,Z1,Z2,XLOCAL,ZZ)
            DZ = (ZTIP-ZFLAT)/9.
            DO 80 M=1,10
              ZINT(M) = ZTIP - FLOAT(M-1)*DZ
              CALL LININT(ZTIP,ZZ,YOUT(L-1,NPTS/2+1),YY,ZINT(M),YYY)
```

LINE # SOURCE TEXT

```
            YINT(M) = YYY
 80       CONTINUE
          DO 100 M=J1,1,-1
            DO 90 I=2,NUL
              IF(XLOCAL.LE.XBASE(I,IFLAT,M)) THEN
                X1 = XBASE(I-1,IFLAT,M)
                X2 = XBASE(I  ,IFLAT,M)
                Y1 = YBASE(I-1,IFLAT,M)
                Y2 = YBASE(I  ,IFLAT,M)
                Z1 = ZBASE(I-1,IFLAT,M)
                Z2 = ZBASE(I  ,IFLAT,M)
                CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
                CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
                YINT(J1-M+1 +10) = YY
                ZINT(J1-M+1 +10) = ZZ
                GOTO 100
              ENDIF
 90         CONTINUE
100       CONTINUE
          KT = 10 + J1
          GOTO200
105     CONTINUE
      ENDIF
C
200   CONTINUE
      RETURN
      END
```

LINE # SOURCE TEXT

```
C***********************************************************************
      SUBROUTINE WINGIN
      include "sgrid.com"
C
C     NC  = # of sections in the spanwise direction
C     ZBASE    = Z value of Kth section
C     XLE(K)   = leading edge X value of Kth section
C     YLE(K)   = leading edge Y value of Kth section
C     CHORD(K) = Chord length of the Kth section
C     NU       = # of points in the upper & lower (Kth) section
C     Note : the end points of upper and lower sections are same physical pts
C

C     MACH2 configuration
      READ(10,900)
      READ(10,910)
      READ(10,920)NC,NU
      NL = NU

      K=1
111   CONTINUE
      READ(10,930)
      READ(10,950)
      DO 12 I=1,NU
12    READ(10,*)XBASE(I,2,K),YBASE(I,2,K),ZBASE(I,2,K)
      READ(10,940)
      READ(10,950)
      DO 15 I=1,NU
15    READ(10,*)XBASE(I,1,K),YBASE(I,1,K),ZBASE(I,1,K)
      XLE(K) = XBASE(1,1,K)
      YLE(K) = YBASE(1,1,K)
      CHORD(K) = ABS(XBASE(1,1,K)-XBASE(NU,1,K))
      K=K+1
      IF(K.LE.NC)GOTO111
C
900    FORMAT(1X)
910    FORMAT(1X)
920    FORMAT(25X,I5,9X,I5/)
930    FORMAT(1X)
940    FORMAT(1X)
950    FORMAT(1X)
960    FORMAT(3F16.7)
       RETURN
       END
```

LINE # SOURCE TEXT

```
      SUBROUTINE FUSEIN
      include "sgrid.com"
      DIMENSION YT(NPI),ZT(NPI)
c
c
c     NF      = # of section in the fuselage
c     NFP     = # of points in mth section
c
c
c     Read the fuselage geometry

c     MACH2 configuration
      READ(10,810)
      READ(10,820)
      READ(10,830)NF,NFP
c
      M=0
 40   CONTINUE
c
      READ(10,840)
      M = M + 1
c
      DO 100 K=1,NFP
        READ(10,*) XIN(M),YIN(M,K),ZIN(M,K)
        YT(K)=YIN(M,K)
        ZT(K)=ZIN(M,K)
 100  CONTINUE
c      CALL DISTARC(YT,ZT,NFP,YT,ZT,NFP,-10.,0)
      DO 110 K=1,NFP
       YIN(M,K)=YT(K)
       ZIN(M,K)=ZT(K)
 110  CONTINUE
      IF(M.LT.NF) GOTO 40
c
c     Write the fuselage geometry into PLOT3D Planar format
      KW=1
      KK=NFP
      WRITE(51)KK,KW,M
      DO 800 L=1,M
        WRITE(51)(XIN(L),K=1,KK),
     .               (YIN(L,K),K=1,KK),
     .               (ZIN(L,K),K=1,KK)
 800  CONTINUE
c
 810  FORMAT(1X)
 820  FORMAT(1X)
 830  FORMAT(31X,I5,8X,I5/)
 840  FORMAT(1X)
 850  FORMAT(2X,3F16.7)
      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE NACIN
      include "sgrid.com"
      DIMENSION YT(NPI),ZT(NPI)
      COMMON /NACG/ XNAC(NPI),YNAC(NPI,NPI),ZNAC(NPI,NPI)
      COMMON /NDIM/ NNAC,NNACP,LNUM(4)
c
c
c     NNAC     = # of section in the nacelle
c     NNACP    = # of points in mth section
c
c
c     Read the nacelle geometry
      READ(30,810)
      READ(30,820)
      READ(30,830)NNAC,NNACP
c
      M=0
 40   CONTINUE
c
      READ(30,840)
      M = M + 1
c
      DO 100 K=1,NNACP
        READ(30,850) XNAC(M),YNAC(M,K),ZNAC(M,K)
        YT(K)=YNAC(M,K)
        ZT(K)=ZNAC(M,K)
100   CONTINUE
c      CALL DISTARC(YT,ZT,NNACP,YT,ZT,NNACP,-10.,0)
c      DO 110 K=1,NNACP
c        YIN(M,K)=YT(K)
c        ZIN(M,K)=ZT(K)
c110   CONTINUE
      IF(M.LT.NNAC) GOTO 40
c
c     Write the nacelle geometry into PLOT3D Planar format
c      KW=1
c      KK=NNACP
c      WRITE(52)KK,KW,M
c      DO 800 L=1,M
c        WRITE(52)(XNAC(L),K=1,KK),
c     .            (YNAC(L,K),K=1,KK),
c     .            (ZNAC(L,K),K=1,KK)
c800   CONTINUE
c       call flush (52)

810   FORMAT(1X)
820   FORMAT(1X)
830   FORMAT(31X,I5,8X,I5/)
840   FORMAT(1X)
850   FORMAT(2X,3F16.7)

      RETURN
      END
```

LINE # SOURCE TEXT

```
      SUBROUTINE WINGMAKER
c
      include "sgrid.com"
c
c     This program generates a 'clipped' delta wing with no twist
c     based on airfoil coordintes read in from fort.90. To use as
c     part of samgrid, WINGIN is not necessary (nor is VARISWEEP).
c
c             Written by: Donovan L. Mathias
c                       : July 1992
c
c     Whenever possiblem the same variables are used as in samgrid.f
c        fort.90  airfoil coordinatres
c        fort.77  desorition of the wing
c
c-----------------------------------------------------------------
c     Declarations
c
      REAL XTE(LS),SLOPE1,SLOPE2,SCALE,XAF(150)
      REAL YU(150),YL(150)
      real SPAN
      INTEGER L,I,J,K
c
c     Initialization
c
      read(77,*)NC
      read(77,*)XLE(1),XLE(NC)
      read(77,*)XTE(1),XTE(NC)
      read(77,*)ZBASE(1,1,1),ZBASE(1,1,NC)
c
c     Read in aifoil coordinates (L is # of X coords.)
c
      READ(90,*)
      READ(90,*)
      READ(90,*)NU
      READ(90,*)
      NL=NU
      DO I=1,NU
         READ(90,19)XAF(I),YU(I),YL(I)
      ENDDO
19    format(3x,f9.7,3x,f9.7,3x,f9.7)
c
c     Establish Z distance (Spanwise)
c
      SPAN = ZBASE(1,1,NC)- ZBASE(1,1,1)
c
      DO K=0,NC-1
         DO I=1,NU
            ZBASE(I,1,K+1) = (K*(SPAN/(NC-1)))
            ZBASE(I,2,K+1) = (K*(SPAN/(NC-1)))
         ENDDO
      ENDDO
c
c     Establish Sweep  (1 FOR LE, 2 FOR TE)
c
      SLOPE1 = (XLE(NC)-XLE(1))/(ZBASE(1,1,NC)-ZBASE(1,1,1))
      SLOPE2 = (XTE(NC)-XTE(1))/(ZBASE(1,1,NC)-ZBASE(1,1,1))
c
c     Generate leading and trailing edges
c
      DO K=1,NC
         XLE(K) = XLE(1) + SLOPE1*ZBASE(1,1,K)
         XTE(K) = XTE(1)+SLOPE2*(ZBASE(1,1,K)-ZBASE(1,1,1))
      ENDDO
c
c     Distribute grid points
c
      DO K=1,NC
         SCALE = XTE(K)-XLE(K)
         DO I=1,NU
            XBASE(I,1,K) = XLE(K) + SCALE*XAF(I)
            YBASE(I,1,K) = YU(I)*SCALE
            XBASE(I,2,K) = XLE(K) + SCALE*XAF(I)
            YBASE(I,2,K) = YL(I)*SCALE
         ENDDO
      ENDDO
c
c     Return values to original names
c
      DO K=1,NC
         CHORD(K) = ABS(XLE(K)-XTE(K))
         YLE(K) = YBASE(1,1,K)
      ENDDO
      RETURN
      END
```

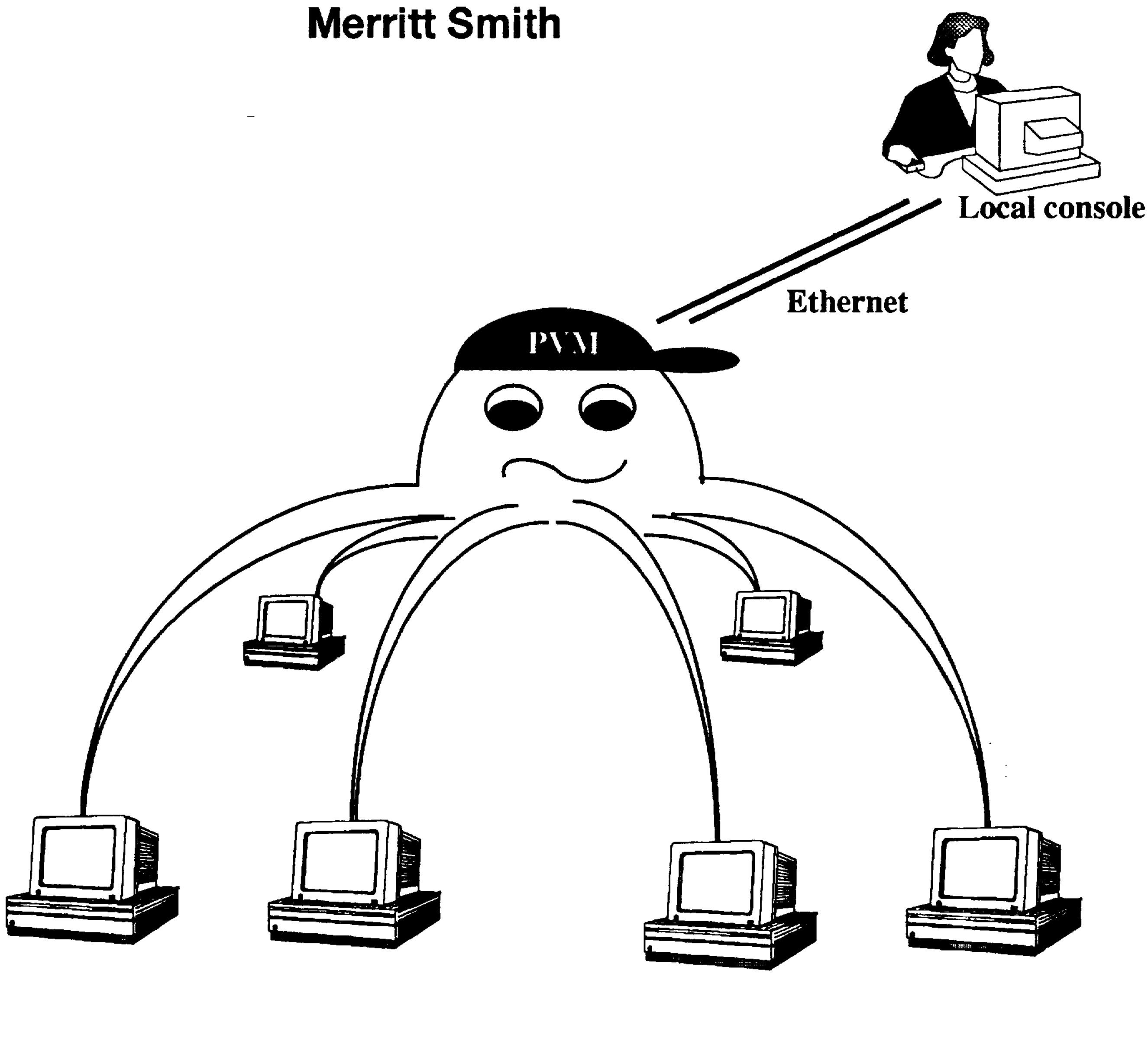
# Appendix C

## PVM Manual

# *Manual of PVM*

**Samson Cheung**

**Merritt Smith**



# Table of Contents

# *Preface*

This manual serves as a supplementary document for the official reference manual of a relatively new research software, PVM, which has been developed at Oak Ridge National Laboratory. A beginner, who has no previous experience with PVM, would find this manual useful.

We would like to thank you in advance that if you find any problems in PVM or this manual, please contact one of us.

Mr. Merritt Smith
NASA Ames Research Center, MS 258-1
Moffett Field, CA 94035
e-mail: mhsmith@nas.nasa.gov
phone: (415) 604-4462

Dr. Samson Cheung
MCAT Institute
NASA Ames Research Center, MS 258-1
Moffett Field, CA 94035
e-mail: cheung@nas.nasa.gov
phone: (415) 604-4462

# 1 *INTRODUCTION*

This manual provides you with an introduction to PVM and provides the fundamentals necessary to write FORTRAN programs in the PVM environment through a tutorial sample. This manual is designed for those who have no previous experience with PVM. However, you should know basic FORTRAN programming and UNIX. If you are ready for an advanced PVM application, please consult the official PVM Reference Manual.

## Software Package

PVM stands for Parallel Virtual Machine. It is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. PVM allows you to link up all or some of the computational systems on which you have accounts, to work as a single distributed-memory (parallel) machine. We call this a Virtual Machine.


PVM is useful for the following reasons. Unlike large mainframe computers or vector supercomputers, workstations spend most of the time idle. The idle time on a workstation represents a significant computational resource. PVM links these workstations up to become a powerful multi-processor computational machine. With PVM, the lack of supercomputer resources should not be an obstacle to number crunching computational programs. Furthermore, the annual maintenance costs of a vector supercomputer is often sufficient to purchase the equivalent computing resource in the form of workstation CPU's.

## Definitions

Here are some terms we use throughout this document:

| | |
|---|---|
| *Virtual Machine* | PVM links different user-defined computers together to perform as one large distributed-memory computer. We call this computer the Virtual Machine. |
| *Host* | Individual computer (member) in the virtual machine. |
| *Process* | Individual program operating on different computers or hosts. |
| *Processor* | The processing unit in computers. A virtual machine can be viewed as a multi-processor computer. |

| | |
|---|---|
| *Task* | The unit of computation handled by the virtual machine. You may want to think of one processor handling one task. |
| *Tid* | Task identification number which is a unique number used by the daemon and other tasks. |
| *Console* | A program from which you can directly interact with the virtual machine. (Add hosts, kill processes,...) |

## Structure of PVM

The PVM software is composed of two parts. The first part is a daemon. We call it *pvmd*. This is the control center of the virtual machine. It is responsible for starting processes, establishing links between processes, passing messages, and many other activities in PVM. Since the daemon runs in the background, you have to use PVM console to directly interact with the virtual machine.

The second part of the system is a library of PVM interface routines located in `libpvm3.a`. This library contains user callable routines for message passing, spawning processes, coordinating tasks, and modifying your machine. In writing your application, you will need to call the routines in this library.

## Directory Setup

This setup is for NAS system. Before you use PVM, you need to set up the following directories on *all* the machines that you want PVM to link:

- Make a directory `$HOME/pvm3/bin/ARCH` in all the hosts of the virtual machine.

  Note `ARCH` is used throughout this manual to represent the architecture name that PVM uses for a given computer. The table in the Appendix lists all the ARCH names that PVM supports. For example, for Silicon Graphic IRIS workstations, you should make a directory `$HOME/pvm3/bin/SGI`.

- Make a directory `$HOME/pvm3/include`, and copy the file `fpvm3.h` from `/usr/nas/pkg/pvm3.2/include`. (If you are on different system from NAS, please consult your system consultant.)
- Make a directory `$HOME/pvm3/codes`, and write your application programs in this directory. You can actually put your programs anywhere you like as long as the correct "include" files are includes. The current setup is for clarity.

# 2 Programming Concept

Unlike graphical software or a word-processor, you cannot *see* PVM working by clicking your mouse buttons. In fact, a virtual machine is quite an abstract concept because you don't physically have a multi-processor machine! In this chapter, you will learn a simple concept, which will help you to visualize how PVM works.

## Master and Slaves

A common way to work with PVM is a *Master/Slave* relationship. A Master process starts Slave routines and distributes work. However, a Master does not actively participate in the computation. A Master process most often resides on the originating host (user's computer), while the Slave programs are distributed to the hosts of the virtual machine.

You need to distribute executables of Slave programs to the directory `$HOME/pvm3/bin/ARCH` on every host. You can locate this Master program anywhere you like.

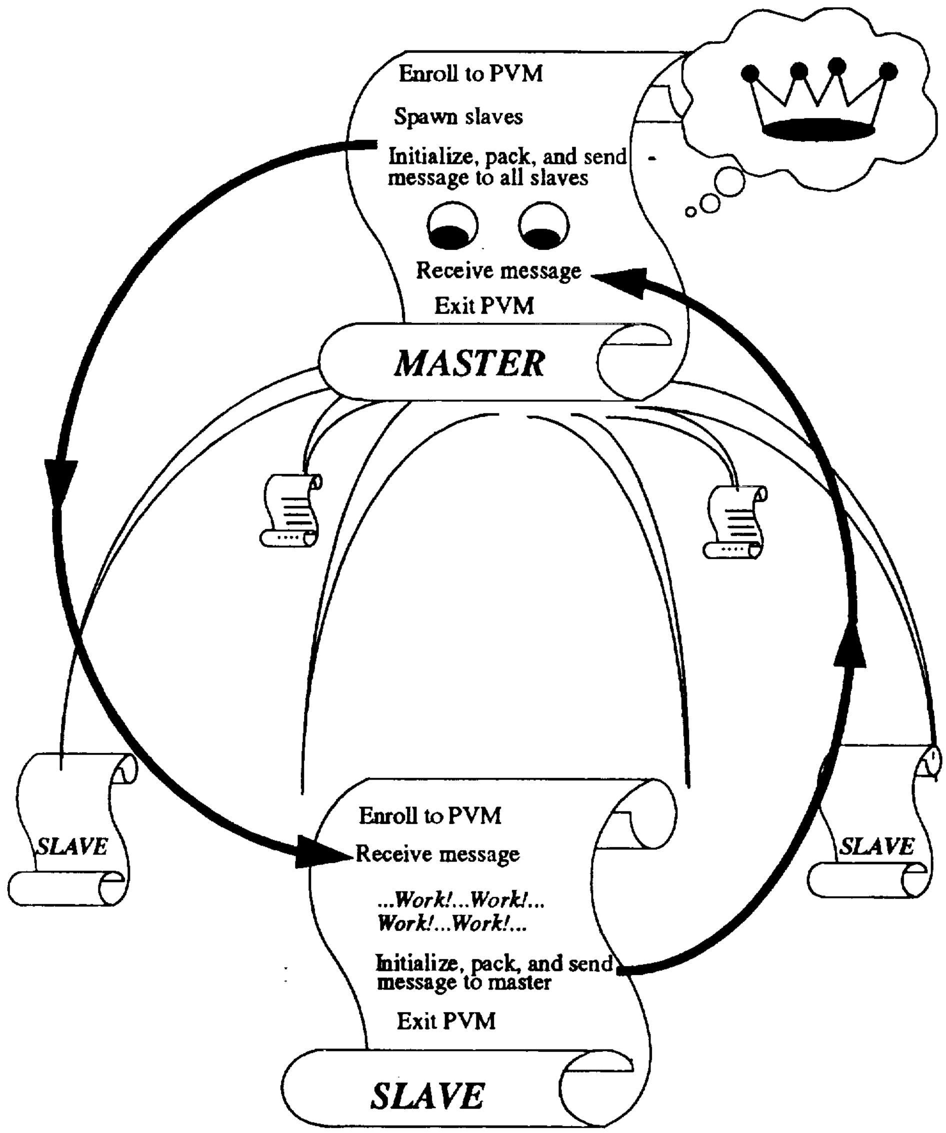
Since the Master program spawns Slave programs on each of the hosts to do jobs, it is important to understand the communication (message passing) amoung the hosts in PVM.

Typically, a Master and a Slave have the following logic:

| | Master | | Slave |
|---|---|---|---|
| 1 | Enroll itself to PVM | 1 | Enroll itself to PVM |
| 2 | Spawn slave processes | 2 | Receive message from master |
| 3 | Initialize buffer, pack, and send message to all slaves. | 3 | *...do something useful...* |
| 4 | *...wait for slaves to finish...* | 4 | Initialize buffer, pack, and send message to master |
| 5 | Receive message from slave(s) | 5 | Exit PVM |
| 6 | Exit PVM | | |

The figure on the opposite page graphically describes a Master/Slave relationship and shows the exchange of information.

**FIGURE 1. Communication in *Master/Slave* programs.**



## SPMD

Another common way to work with PVM is the *SPMD*, Single Program Multiple Data model. There is only a single program, and there is no Master program directing the computation. The user starts the first copy of the program and using the routine `pvmfparent()`, this copy can determine that it was not spawned by PVM, and thus must be the first copy (parent). It then spawns multiple copies (children) of itself and passes them the array of *tids*. At this point each copy is equal and can work on its partition of the data in collaboration with the other processes.

Typically, a SPMD program has the following logic:

1. **Enroll in pvm**
2. **If I am the first copy (parent)**
   a) Spawn child processes
   b) Initialize buffer, pack, and send message out
3. **If I am a secondary copy (child)**
   Receive messages
4. **Work!...Work!...Work!**
5. **Exit PVM**

The program on the opposite page describes a SPMD logic and shows the exchange of information. Please spend some time to study the program.

In the next chapter we will introduce the PVM daemon and the fundamentals of message passing.

## SPMD Program

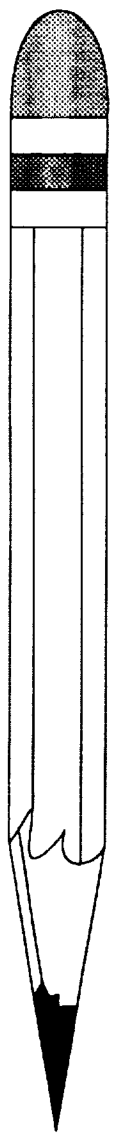
```
    c-------------------------------------
    c  SPMD Fortran example using PVM 3.0
    c-------------------------------------
          program spmd
          include '../include/fpvm3.h'
          PARAMETER( NPROC=4 )
          integer mytid, me, i
          integer tids(0:NPROC)

    c     Enroll in pvm
1         call pvmfmytid( mytid )

    c     ---------------------------------------------------
    c     Find out if I am parent or child
    c     ---------------------------------------------------
          call pvmfparent(tids(0))
          if( tids(0) .lt. 0 )  then
            tids(0) = mytid
            me = 0
    c       -------------------------
    c       start up copies of myself
    c       -------------------------
            call pvmfspawn('spmd',PVMDEFAULT,'*',
2        * NPROC-1,tids(1), info)
    c       ---------------------------
    c       multicast tids array to children
    c       ---------------------------
            call pvmfinitsend( PVMDEFAULT, info )
            call pvmfpack( INTEGER4, tids, NPROC, 1, info )
            call pvmfmcast( NPROC-1, tids(1), 0, info )
          else
    c       ---------------------------------
    c       receive the tids array and set me
    c       ---------------------------------
3           call pvmfrecv( tids(0), 0, info )
            call pvmfunpack( INTEGER4, tids, NPROC,1,info)
            do 30 i=1, NPROC-1
              if( mytid .eq. tids(i) ) me = i
       30   continue
          endif
    c-------------------------------------------------------
    c  all NPROC tasks are equal now
    c  and can address each other by tids(0) thru tids(NPROC-1)
    c  for each process me => process number [0-(NPROC-1)]
    c-------------------------------------------------------
          print*,'me =',me, '  mytid =',mytid
4         call dowork( me, tids, NPROC )

    c     -------------------------
    c     program finished exit pvm
    c     -------------------------
5         call pvmfexit(info)
          stop
          end
```

## *Notes*



```
      subroutine dowork( me, tids, nproc )
      include '../include/fpvm3.h'
c-------------------------------------------------
c Simple subroutine to pass a token around a ring
c-------------------------------------------------
      integer me, nproc
      integer tids( 0:nproc)

      integer token, dest, count, stride, msgtag

      count  = 1
      stride = 1
      msgtag = 4

      if( me .eq. 0 ) then
        token = tids(0)
        call pvmfinitsend( PVMDEFAULT, info )
        call pvmfpack( INTEGER4,token,count,stride,info)
        call pvmfsend( tids(me+1), msgtag, info )
        call pvmfrecv( tids(nproc-1), msgtag, info )
        print*, 'token ring done'
      else
        call pvmfrecv( tids(me-1), msgtag, info )
        call pvmfunpack( INTEGER4,token,count,stride,info)
        call pvmfinitsend( PVMDEFAULT, info )
        call pvmfpack( INTEGER4,token,count,stride,info)
        dest = tids(me+1)
        if( me .eq. nproc-1 ) dest = tids(0)
        call pvmfsend( dest, msgtag, info )
      endif

      return
      end
```

# 3 *PVM Daemon*

The PVM daemon is the control center of the virtual machine. You can activate the PVM daemon by starting the PVM console or by invoking the daemon directly with a list of hosts. The latter will be discussed in chapter 6. To start the console, enter `pvm` at UNIX prompt on your local machine. The PVM console prints the prompt

```
pvm>
```

and accepts commands from standard input. The console allows interactive adding and deleting of hosts to the virtual machine as well as interactive starting and killing of PVM processes. Even if the daemon is started directly, the console can be used to modify the virtual machine.

## Console Commands

Here are the commands available in the PVM console:

| | |
|---|---|
| ADD | add other computers (hosts) to PVM |
| ALIAS | define and list command aliases you set |
| CONF | show members in virtual machine |
| DELETE | remove hosts from pvm |
| ECHO | echo arguments |
| HALT | stop all pvm processes and exit deamon |
| HELP | print this information |
| ID | print console task identity |
| JOBS | display list of running jobs |
| KILL | terminate tasks |
| MSTAT | show status of hosts |
| PS | list tasks |
| PSTAT | show status of tasks |
| QUIT | exit PVM console, but PVM daemon is still activated |
| RESET | kill all tasks |
| SETENV | display or set UNIX environment variables |
| SIG | send signal to task |
| SPAWN | spawn task |
| UNALIAS | remove alias commands you previous set |
| VERSION | show PVM version |



C-2

## Console Usage

Suppose the console is running on workstation *win210*. This computer will automatically be a host in your virtual machine. Here are some examples of using PVM console:

1. **Activate PVM console**

```
win210> pvm
```

2. **Add amelia and fred to your virtual machine**

```
pvm> add amelia
1 successful
    HOST        DTID
    amelia      c0000
pvm> add fred
1 successful
    HOST        DTID
    fred        100000
```

3. **Check the configuration of your virtual machine**

```
pvm> conf
3 host, 1 data format
    HOST        DTID        ARCH        SPEED
    win210      40000       SGI         1000
    amelia      c0000       SGI         1000
    fred        100000      SGI         1000
```

4. **Delete amelia**

```
pvm> delete amelia
1 successful
    HOST        STATUS
    amelia      deleted
```

5. **Exit PVM console, but PVM daemon is still running**

```
pvm> quit
pvmd still running
win210>
```

# 4 PVM Library

This chapter introduces the PVM library. In writing your application programs, you need to call the subroutines in the library to instruct PVM to control processes, send information, pack/unpack data, and send/receive messages. Many subroutines have pre-defined option values for some arguments. These are defined in the include file `fpvm3.h` and are listed in the Appendix.

## Process Control

**call pvmfmytid( tid )**

This routine enrolls this process with the PVM daemon on its first call, and generates a unique **tid**. You call this routine at the beginning of your program.

**call pvmfexit( info )**

This routine tells the local PVM daemon that this process is leaving PVM. You call this routine at the end of your program. Values of **info** less than zero indicate an error.

**call pvmfkill( tid, info)**

This routine kills a PVM task identified by **tid**. Values of **info** less than zero indicate an error.

**call pvmfspawn( pname,flag,where,ntask,tids,numt )**

This routine starts up **ntask** instances of a single process named **pname** on the virtual machine. Here are the definition of the other arguments:

**flag**

| Option Value | Meaning |
|---|---|
| PVMDEFAULT (0) | PVM can choose any machine to start task |
| PVMHOST (1) | where specifies a particular host |
| PVMARCH (2) | where specifies a type of architecture |
| PVMDEBUG (4) | start up processes under debugger |
| PVMTRACE (8) | processes will generate PVM trace data |

**where** is where you want to start the PVM process. If **flag** is 0, **where** is ignored.

**tids** contains identification numbers of PVM processes started by this routine.

**numt** indicates how many processors started; negative values indicate an error.

Note You should always check **tids** and **numt** to make sure all processes started correctly.

**call pvmfparent ( tid )**

This routine returns the tid of the process that spawned this task. If the calling process was not created with pvmfspawn, then tid=PvmNoParent.

## Dynamic Configuration

**call pvmfaddhost( host, info )**

**call pvmfdelhost( host, info )**

These routines add and delete hosts to the virtual machine respectively. Values of info less than zero indicate an error.

Note Both routines are expensive operations that require the synchronization of the virtual machine.

## Message Buffers

**call pvmfinitsend( encoding, bufid )**

This routine clears the send buffer, and creates a new one for packing a new message.

encoding

| Encoding Value | Meaning |
|---|---|
| PVMDEFAULT (0) | XDR encoding if virtual machine configuration is heterogeneous |
| PVMRAW (1) | no encoding is done. Messages are sent in their original format. |
| PVMINPLACE (2) | data left in place. Buffer only contains sizes and pointers to the sent items. |

This is not implemented in PVM v3.2.

bufid contains the message buffer identifier. Values less than zero indicate an error.

**call pvmffreebuf( bufid, info)**

This routine disposes the buffer with identifier bufid. You use it after a message has been sent, and is no longer needed. Values of info less than zero indicate an error.

## Packing and Unpacking

**call pvmfpack( what, xp, nitem, stride, info )**

**call pvmfunpack( what, xp, nitem, stride, info )**

These routines pack/unpack your message xp, which can be a number or a string. You can call these routines multiple times to pack/unpack a single message. Thus a message can contain several arrays, each with a different data type.

Note There is no limit to the complexity of the packed messages, but you must unpack them exactly as they were packed.

**what** indicates what type of data **xp** is

| | |
|---|---|
| STRING (0) | REAL (4) |
| BYTE1 (1) | COMPLEX8 (5) |
| INTEGER2 (2) | REAL8 (6) |
| INTEGER4 (3) | COMPLEX16 (7) |

**nitem** is number of items in the pack/unpack. If **xp** is a vector of 5, **nitem** is 5.

**stride** is the stride to use when packing.

**info** is status code returned by this routine. Values less than zero indicate an error.

## Sending and Receiving

**call pvmfsend( tid, msgtag, info )**

This routine labels the message with an integer identifier **msgtag**, and sends it immediately to the process **tid**. Values of **info** less than zero indicate an error.

**call pvmfmcast( ntask, tids, msgtag, info )**

This routine labels the message with an integer identifier **msgtag**, and broadcasts the message to all **ntask** number of tasks specified in the integer array **tids**. Values of **info** less than zero indicate an error.

**call pvmfrecv( tid, msgtag, bufid )**

This routine blocks the flow of your program until a message with label **msgtag** has arrived from **tid**. A value of -1 in **msgtag** or **tid** matches anything (wildcard). This routine creates a new active receive buffer, and puts the message in it. Values of **bufid** identify the newly created buffer; values less than zero indicate an error.

**call pvmfnrecv( tid, msgtag, bufid )**

This routine performs in the same way as **pvmfrecv**, except that it does not block the flow of your program. If the requested message has not arrived, this routine returns **bufid**=0. This routine can be called multiple times for the same message to check if it has arrived, while performing useful work between calls. When no more useful work can be performed, the blocking receive **pvmfrecv** can be used for the same message.

**call pvmfprobe( tid, msgtag, bufid )**

This routine checks if a message has arrived; however, it does not receive the message. If the requested message has not arrived, this routine returns **bufid**=0. This routine can be called multiple times for the same message to check if it has arrived, while performing useful work between calls.

**call pvmfbufinfo (bufid, bytes, msgtag, tid, info)**

This routine returns information about the message in the buffer identified by **bufid**. The information returned is the actual **msgtag**, source **tid**, and message length in **bytes**. Values of **info** less than zero indicate an error.

# 5 Tutorial

This chapter shows you how PVM may be applied to your application programs through a simple example. The example chosen is the Golden Section rule for finding the maximum of a function. You may remember it from Math class in high school. Let us review the method and the algorithm.

## Golden Section

Suppose we want to find the maximum of a curve y=f(x); where x is between the interval $a_1$ and $a_2$. The points $a_3$ and $a_4$ are symmetrically placed in this interval, so that

$$a_3 = (1-\alpha)\, a_1 + \alpha\, a_2 \qquad \text{(EQ 1)}$$

$$a_4 = \alpha\, a_1 + (1-\alpha)\, a_2 \qquad \text{(EQ 2)}$$

See Figure 1 at left. Golden Section rule requires $\alpha$ to be 0.382.

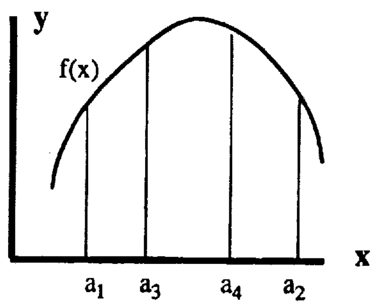



Figure 1. Interval division for Golden Section

The algorithm of finding the maximum is as follow:

| If $f(a_4) < f(a_3)$ | | If $f(a_4) > f(a_3)$ | |
|---|---|---|---|
| 1 | Consider new interval $(a_1,a_4)$ | 1 | Consider new interval $(a_3,a_2)$ |
| 2 | Apply EQ.(1) and (2) again | 2 | Apply EQ. (1) and (2) again |
| 3 | Until maximum is reached | 3 | Until maximum is reached |

If $f(a_3)=f(a_4)$, the maximum is found

The FORTRAN program (Serial Program) on the opposite page is the Golden Section rule that a programmer would write on a normal serial computer. Please spend a few minutes to study the flow of the program. This simple program consists of two parts, the main (calling) program and the function subroutine. The latter has only four lines.

Note Notice that for each interval $(a_1,a_2)$, we need to call the function evaluation four times to find $f(a_1)$, $f(a_2)$, $f(a_3)$, and $f(a_4)$.

## Serial Program

Golden Section rule → ALPHA = 0.382

Equations (1) and (2) → A(3), A(4)

Four function evaluations: FN(1)–FN(4)

If $f(a_4) > f(a_3)$

If $f(a_4) < f(a_3)$

Function evaluation: FUNCTION F(X)

```
C     Linear optimization:
C
C     Search for maximum of a x-y curve.
C
      DIMENSION A(4),FN(4)
C
C     Initial interval
      L = 0
      TOL = 1.E-3
      A(1) = 0.4
      A(2) = 1.6
      ALPHA = 0.382
C
C
 10   CONTINUE
C
C     Loop begins:
      L = L + 1

      A(3) = (1.-ALPHA)*A(1) + ALPHA*A(2)
      A(4) = ALPHA*A(1) + (1.-ALPHA)*A(2)
      FN(1) = F(A(1))
      FN(2) = F(A(2))
      FN(3) = F(A(3))
      FN(4) = F(A(4))
      WRITE(10,*) 'A  ',A(1),A(2),A(3),A(4)
      WRITE(10,*) 'F  ',FN(1),FN(2),FN(3),FN(4)
      WRITE(10,*) ' '
      ERR = ABS(FN(2)-FN(3))
      IF(ERR.LE.TOL) GOTO 999
C
C
      IF(FN(4) .GT. FN(3)) THEN
        B1 = A(3)
        B2 = A(2)
        A(1) = B1
        A(2) = B2
        GOTO 10
      ELSEIF(FN(4) .LT. FN(3)) THEN
        B1 = A(1)
        B2 = A(4)
         A(1) = B1
        A(2) = B2
        GOTO 10
      ENDIF
 999  CONTINUE
       STOP
      END

      FUNCTION F(X)
      F = TANH(X)/(1.+X*X)
      RETURN
      END
```

## PVM Master Guideline

Recall that in the procedure of finding a new interval, the program calls the function evaluation four times *serially* to get $f(a_1)$, $f(a_2)$, $f(a_3)$, and $f(a_4)$. We would like to assign four processors to perform the four function evaluations *simultaneously* on the virtual machine. Therefore, we modify the Serial Program by writing the main (calling) program as a Master program, and the function subroutine as a Slave program.

The following steps are general guidelines to writing a Master program. Please study the steps, and compare them with the program on the opposite page. Also compare it with the Serial Program.

1. **Include fpvm3.h**

   Include this file in your program, you are able to use the PVM preset variables; such as `PVMDEFAULT`, `REAL4`, and more, mentioned in Chapter 4 and the Appendix.

2. **Enroll Master to PVM**

   Use `pvmfmytid(mytid)` to enroll.

3. **Assign virtual processors**

   Use the following call to spawn `nproc` function processes.
   `pvmfspawn(pname,PVMDFAULT,where,nproc,tids,numt)`
   Also tell PVM the name of the Slave program (`pname`). PVM returns `tids`, the identifier of the `nproc` processors.

4. **Initialize buffer and pack data**

   Use `pvmfinitsend` to clear buffer.
   Use the following routine to pack a real array `A` of dimension `m`.
   `pvmfpack(REAL4,A,m,1,info)`

5. **Send message**

   Use the following call to send the packed message to the Slave process identified by `tids`.
   `pvmfmcast(nproc,tids,msgtag,info)`

# Master Program

1

2

3

Assign four processors

Slave program's name

Equations (1) and (2)

4

Pack nproc, tids, A, and ERR

5

msgtype value matches the one received in Slave program

```
c     Linear optimization:
c     Search for maximum of a x-y curve.
       PROGRAM MASTER
c
c
      include '../include/fpvm3.h'
      DIMENSION A(4),FN(4)
      integer tids(0:32),who
      character*8 where
      character*12 pname

c     Enroll this program in PVM
      call pvmfmytid(mytid)
c     Start up the four processors
      nproc = 4
      where = '*'
      pname = 'function'
      call pvmfspawn(pname,PVMDFAULT,where,nproc,tids,numt)
      do 20 i=0,nproc-1
        write(*,*) 'tid', i, tids(i)
 20   continue
c
c     Initial interval
      L = 0
      A(1) = 0.4
      A(2) = 1.6
      ALPHA = 0.382
      TOL = 1.E-3
      ERR = 1.
c
 10   CONTINUE
c
c
c     Loop begins:
      L = L + 1

      A(3) = (1.-ALPHA)*A(1) + ALPHA*A(2)
      A(4) = ALPHA*A(1) + (1.-ALPHA)*A(2)

c
c     Broadcast data to all node programs
c     first pack them, then send them
      call pvmfinitsend(PVMDEFAULT,info)
      call pvmfpack(INTEGER4,nproc,1,1,info)
      call pvmfpack(INTEGER4,tids,nproc,1,info)
      call pvmfpack(REAL4,A,4,1,info)
      call pvmfpack(REAL4,ERR,1,1,info)
c
c
      msgtype = 1
      call pvmfmcast(nproc,tids,msgtype,info)
c
```

6. **Wait until messages come from Slaves**

   Use `pvmfrecv()` to block until Slaves return function values. Make sure value of `msgtype` matches values coming from Slaves.

7. **Receive and Unpack data**

   The sequence of unpacking is the same as the packing in the Slave.

8. **Exit PVM**

   Use `pvmfexit(info)` to exit PVM.

6

msgtype value matches the one sent from Slave program

7

Receive /unpack FN and 'who' from the 4 processors one by one

8

```
c     Wait for results from processors

      msgtype = 2
      do 100 i=1,nproc
        call pvmfrecv(-1,msgtype,info)
        call pvmfunpack(INTEGER4,who,1,1,info)
        call pvmfunpack(REAL4,FN(who),1,1,info)
100   continue

      WRITE(10,*) 'A  ',A(1),A(2),A(3),A(4)
      WRITE(10,*) 'F  ',FN(1),FN(2),FN(3),FN(4)
      WRITE(10,*) ' '
      ERR = ABS(FN(2)-FN(3))
      IF(ERR.LE.TOL) GOTO 999
C
C
      IF(FN(4) .GT. FN(3)) THEN
        B1 = A(3)
        B2 = A(2)
        A(1) = B1
        A(2) = B2
        GOTO 10
      ELSEIF(FN(4) .LT. FN(3)) THEN
        B1 = A(1)
        B2 = A(4)
        A(1) = B1
        A(2) = B2
        GOTO 10
      ENDIF
c
c     Program finished leave PVM before exiting
999   continue
      call pvmfexit(info)
      STOP
      END
```

## PVM Slave Guideline

The Slave program is basically the function evaluation program. In order to do the function evaluation, it needs information from Master. For example, it needs the identity numbers (`tids(1),...,tids(4)`) that PVM assigns, and the values of $a_1,...,a_4$.

The following steps are general guidelines to writing a Slave program. Please study the steps, and compare them with the program on the opposite page. Also try to find the connection with the Master Program. You may find Figure 1 helpful.

1. **Include fpvm3.h**

   Include this file in your program, you are able to use the PVM preset variable names; such as `PVMDEFAULT`, `REAL4`, and more, mentioned in all tables in Chapter 4 and the Appendix.

2. **Enroll Slave with PVM**

   Use `pvmfmytid(mytid)` to enroll.

3. **Identify the parent of this process**

   Use the following call to obtain the task identifyer (`mtid`) of parent process. This is useful for returning solutions to the Master.
   `pvmfparent(mtid)`

4. **Receive and Unpack data**

   Make sure the value of `msgtype` matches the one from Master. The sequence of unpacking is the same as the packing in Master.

5. **Perform function evaluation**

6. **Initialize buffer and pack data**

   Use `pvmfinitsend` to clear buffer.

   Use the following call to pack a real array `F` of dimension `n`.
   `pvmfpack(REAL4,F,n,1,info)`

7. **Send data**

   Use the following call to send the packed message to Master:
   `pvmfsend(mtid,msgtag,info)`

8. **Exit PVM**

   Use `pvmfexit(info)` to exit PVM.

# Slave Program

```
c
      program function
c
c
      include '../include/fpvm3.h'
```

1

```
      integer tids(0:32),who
      real a(32)
      tor = 1.e-3
c
c     Enroll this program in PVM
```

2

```
      call pvmfmytid(mytid)
c     Get the parent's task id
```

3

```
      call pvmfparent(mtid)
c
 3    continue
c
c     Receive data from host
```

4

Unpack the same way as Master sends

```
      msgtype = 1
      call pvmfrecv(mtid,msgtype,info)
      call pvmfunpack(INTEGER4,nproc,1,1,info)
      call pvmfunpack(INTEGER4,tids,nproc,1,info)
      call pvmfunpack(REAL4,A,4,1,info)
      call pvmfunpack(REAL4,ERR,1,1,info)
c
c
      if(err.le.tor) go to 99
c
c     Determine which processor I am
      do 5 i=0,nproc-1
        if(tids(i).eq.mytid) me = i
 5    continue
      who = me + 1
```

Function evaluation

5

```
c     Calculate the function
      X = A(who)
      f = TANH(X)/(1.+X*X)
c
c
```

Pack f and processor 'who'

6

```
c     Send the result to Master
      call pvmfinitsend(PVMDEFAULT,info)
      call pvmfpack(INTEGER4,who,1,1,info)
      call pvmfpack(REAL4,f,1,1,info)
      msgtype = 2
```

7

```
      call pvmfsend(mtid,msgtype,info)
      go to 3
```

Go to 3 and wait for another call from master

```
c
c     Program finished. Leave PVM before exiting
 99   continue
```

8

```
      call pvmfexit(info)
      stop
      end
```

## Compilation and Running

After you finish your program, it is time to compile and run. Follow the steps below to compile your programs.

1. **Make sure you have the correct directory setup**

   Follow the advice from *Directory Setup* in Chapter 1.

2. **Compile the program**

   Use the sample `Makefile` on the opposite page to compile your programs.

Note The `Makefile` links the PVM library, `libfpvm3.a`.

3. **Copy executables to all the hosts**

   Follow the advice from *Directory Setup* in Chapter 1, and distribute the executables to `$HOME/pvm3/bin/ARCH`.

4. **Activate PVM**

   Activate PVM by entering `pvm` at UNIX prompt.

5. **Decide the configuration of the virtual machine**

   Add or delete hosts to the virtual machine. (Chapter 3)

6. **Quit PVM console**

   Leave PVM console (don't halt daemon) by entering `quit` at the `pvm` prompt.

## Makefile

PVM Library

Make appropriate changes for your own path

```
#
# Custom section
# Set PVM_ARCH to your architecture type (SUN4, HP9K, RS6K, # SGI,
etc.)
# if PVM_ARCH = BSD386 then set ARCHLIB = -lrpc
# if PVM_ARCH = SGI    then set ARCHLIB = -lsun
# if PVM_ARCH = I860   then set ARCHLIB = -lrpc -lsocket
# if PVM_ARCH = IPSC2  then set ARCHLIB = -lrpc -lsocket
# otherwise leave ARCHLIB blank
#
# PVM_ARCH and ARCHLIB are set for you if you use 'aimk'.
#
PVM_ARCH        =       SGI
ARCHLIB         =       -lsun
# END of custom section - leave this line here
#
PVMDIR  =       /amd/fs02/pub/iris4d_irix4/nas/pkg/pvm3.2
PVMLIB  =       $(PVMDIR)/lib/$(PVM_ARCH)/libpvm3.a
SDIR    =       .
BDIR    =       /u/wk/cheung/pvm3/bin
XDIR    =       $(BDIR)/$(PVM_ARCH)

CFLAGS  =       -g -I../include
LIBS    =       $(PVMLIB) $(ARCHLIB)

F77     =       f77
FFLAGS  =       -g
FLIBS   =       $(PVMDIR)/lib/$(PVM_ARCH)/libfpvm3.a $(LIBS)

default:        master function

$(XDIR):
        - mkdir $(BDIR)
        - mkdir $(XDIR)

clean:
        rm -f *.o bfgs quadfunct

master: $(SDIR)/master.f  $(XDIR)
        $(F77) $(FFLAGS) -o master $(SDIR)/master.f $(FLIBS)
        mv master $(XDIR)

function: $(SDIR)/function.f  $(XDIR)
        $(F77) $(FFLAGS) -o function $(SDIR)/function.f $(FLIBS)
        mv function $(XDIR)
```

# 6 *Problems and Tips*

PVM is a relatively new piece of software. It is not advanced enough to warn you ahead of time before problems come. Here are a couple of cases that you may encounter as a beginner.

## Problems

### Can't activate PVM

- If the message you get, after entering `pvm` at UNIX prompt, is
  `libpvm [pid-1]: Console: Can't start pvmd,`
  it is possible that the last time you halted PVM daemon, the daemon created a residual file `/tmp/pvmd.xxxx`; where `xxxx` is an unique number for you. Delete this file, and start PVM again.
- If the daemon is running but the PVM console will not start, it is possible that you have too many processes running. You have to kill all the processes before you re-activate PVM console.

Note Use `ps -ef | username` at UNIX prompt to locate your running processes.

### Can't add hosts

It is possible that there are no links between your local computer and the other hosts. Check the following two things:

- Make sure each of your hosts has a `.rhosts` file in the `$HOME` directory, and this file points to your local computer.
- Make sure the `.rhosts` file is "read" and "write" protected from others users.

# Host File

You can create the following file to build the virtual machine without activating the PVM console. The addresses must be recognizable by your system.

```
computer1.address
computer2.address      host file
computer3.address
computer4.address
```

> Note The first machine listed must be the initiating host.
>
> Note If tasks are to be spawned on specific systems, the system name contained in `where` ( routine **pvmfspawn**) must match the name in the host file <u>exactly</u>.
>
> Note If spawning tasks are on the initiating host, use the truncated host name. For example, if the full address is
> `win210.nas.nasa.gov`;
> use `win210` instead. This is a bug in PVM v3.2.

Having the host file ready, enter the following at UNIX prompt,

win210> `pvmd3 host`

## *Notes*

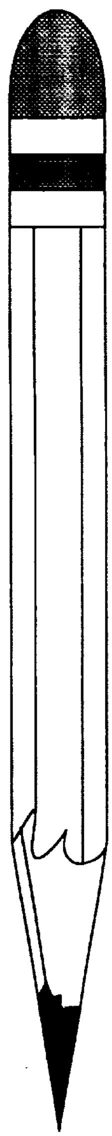
Place to jot down problems.

If encounter problems, please contact:

Merritt Smith: mhsmith@nas.nasa.gov

or

Samson Cheung: cheung@nas.nasa.gov



# Appendix

**TABLE 1. ARCH names used in PVM.**

| ARCH | Machine | Note |
|---|---|---|
| AFX8 | Alliant FX 8 | |
| ALPHA | DEC Alpha | DEC OSF-1 |
| BAL | Sequent Balance | DYNIX |
| BFLY | BBN Butterfly TC2000 | |
| BSD386 | 80386/486 Unix box | BSDI |
| CM2 | Thanking Machines CM2 | Sun front-end |
| CM5 | Thanking Machines CM5 | |
| CNVX | Convex C-series | |
| CNVXN | Convex C-series | native mode |
| CRAY | C-90, YMP, Cray-2 | UNICOS |
| CRAYSMP | Cray S-MP | |
| DGAV | Data General Ayiion | |
| HP300 | HP-9000 model 300 | HPUX |
| HPPA | HP-9000 PA-RISC | |
| I860 | Intel iPSC/860 | link-lprc |
| IPSC2 | Intel iPSC/860 host | SysV |
| KSR1 | Kendall Square KSR-1 | OSF-1 |
| NEXT | NeXT | |
| PGON | Intel Paragon | link -lprc |
| PMAX | DECstation 3100,5100 | Ultrix |
| RS6K | IBM/RS6000 | AIX |
| RT | IBM RT | |
| SGI | Silicon Graphics IRIS | link -lsun |
| SUN3 | Sun 3 | SunOS |
| SUN4 | Sun 4, SPARCstation | |
| SYMM | Sequent Symmetry | |
| TITN | Staedent Titan | |
| UVAX | DEC Micro VAX | |

**TABLE 2. Error codes returned by PVM routines**

| Error Code | Meaning |
|---|---|
| PvmOK (0) | All right |
| PvmBadParam (-2) | Bad parameter |
| PvmMismatch (-3) | Barrier count mismatch |
| PvmNoData (-5) | Read past end of buffer |
| PvmNoHost (-6) | No such host |
| PvmNoFile (-7) | No such executable |
| PvmNoMem (-10) | Can't get memory |
| PvmBadMsg (-12) | Can't decode received massage |
| PvmSysErr (-14) | Pvmd not responding |
| PvmNoBuf (-15) | No current buffer |
| PvmNoSuchBuf (-16) | Bad message identifyer |
| PvmNukkGroup (-17) | Null group name is illegal |
| PvmDupGroup (-18) | Already in group |
| PvmNoGroup (-19) | No group with that name |
| PvmNotInGroup (-20) | Not in group |
| PvmNoInst (-21) | No such instance in group |
| PvmHostFail (-22) | Host failed |
| PvmNoParent (-23) | No parent task |
| PvmNoImpl (-24) | Function not implemented |
| PvmDSysErr (-25) | Pvmd system error |
| PvmBadVersion (-26) | Pvmd-pvmd protocol mismatch |
| PvmOutOfRes (-27) | Out of resources |
| PvmDupHost (-28) | Host already configurated |
| PvmCantStart (-29) | Fail to execute new slave pvmd |
| PvmAlready (-30) | Slave pvmd already running |
| PvmNoTask (-31) | Task does not exist |
| PvmNoEntry (-32) | No such (group,instance) |
| PvmDupEntry (-33) | (Group,instance) already exists |