View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NASA Technical Reports Server

Mississippi State University

Department of Aerospace Engineering

David L. Whitfield

Drawer A

Mississippi State, MS 39762

601-325-3623

# Advanced 3-D Viscous SSME Turbine Rotor Stator CFD Algorithms

6 September 85 - 5 September 86

September 1986

Final Report

NAS8-36486

J. Mark Janus

David L. Whitfield

Prepared for

George C. Marshall Space Flight Center

Marshall Space Flight Center, Alabama 35812

(NASA-CR-178997) ADVANCED 3-D VISCOUS SSME TURBINE ROTOR STATOR CFD ALGORITHMS Final Report, 6 Sep. 1985 - 5 Sep. 1986 (Mississippi State Univ., Mississippi State.) 31 p

N87-16829

CSCL 21E G3/07

Unclas 43858

Abstract

Current SSME high pressure fuel turbopump problems have generated a desire to analyze the flowfield of rotating machinery. The status of CFD has reached the point that soon the capability to solve unsteady three-dimensional viscous flowfields will be at hand. The work presented here involves upgrading the computational efficiency of an operational three-dimensional algorithm. The modifications include algorithm development, algorithm approximation and acceleration, and special coding optimizations. The overall result of these modifications has reduced processing time by nearly 80%.

## Introduction

Presently, there exists a need for a robust and accurate computational algorithm to compute three-dimensional flow over a rotor-stator configuration. An internal flowfield analysis of rotating machinery necessitates the use of an advanced time-accurate three-dimensional algorithm with dynamic grid capability. Additionally, this algorithm must be computationally efficient, i.e., be capable of producing solutions of engineering accuracy to real world flows in a reasonable length of time. Until recently, the CFD community didn't even consider routine calculations of a typical three-dimensional unsteady viscous flow. This was primarily due to the large number of cells required to resolve the pertinent flowfield phenomena. The domain must be subdivided into several million cells, requiring approximately 30 computer words each,[1] the result of which is excessive computational processing times using present day codes. The introduction of the Cray 2 supercomputer and the availability of Cray's SSD storage have alleviated much of the concern regarding storage for those with access to this state-of-the-art equipment.

The initial step toward developing a code which can fulfill the requirements stated above begins with the choice of a baseline algorithm. The baseline algorithm chosen for this endeavor is that described in Ref. 2. This algorithm has been shown to yield satisfactorily resolved solutions of the flow past a variety of geometries in many flow regimes. The baseline computer algorithm, herein referred to as BMULE, is lacking in the computational speed required to accomplish the task previously mentioned. The storage problem is of a lesser concern

in light of the recent advances in computer hardware introduced by Cray Research. An estimate at the onset of this project placed the memory requirement at approximately 100,000,000 words and CRAY X-MP CPU time at approximately 400 hours for a typical three-dimensional rotor-stator configuration flow analysis. It is evident that improvements in the baseline algorithm are in order, before any three-dimensional viscous rotating machinery flowfield solutions will be feasible.

This first year effort lays the ground work for an algorithm, the purpose of which will be to perform the tasks outlined in the first paragraph of this section. The intention here is to improve an operational computer algorithm, keeping in mind the basic goal of approaching the computational efficiency required if one ever expects to produce routine flow analysis of rotating machinery.

The material presented in this report entails three basic approaches to enhance computational efficiency. These are: 1) algorithm improvement, 2) acceptable algorithm approximations, and 3) coding modifications. The first section of this report briefly describes the equations and the baseline algorithm. This is followed by the note-worthy accomplishments in each of the specified approaches, given in the second through the fourth sections. The results of this first year effort are then presented in the fifth section. Finally, the sixth section presents the conclusions of this report based on the year's findings. Also, presented are suggestions for future development in the field of rotating machinery flow analysis.

Equations & Algorithm

The present work is based on the equation set developed in Ref. 3 and the algorithm developed in Ref. 2. Consider the fully discretized linearized integral form of the flux split fluid dynamic equations written as

$$[I + \Delta\tau(\delta_i A^{+}_{\cdot} + \delta_i A^{-}_{\cdot} + \delta_j B^{+}_{\cdot} + \delta_j B^{-}_{\cdot} + \delta_k C^{+}_{\cdot} + \delta_k C^{-}_{\cdot})]\Delta Q^n = -\Delta\tau R^n \qquad (1)$$

where

$$\Delta Q^n = Q^{n+1} - Q^n \qquad (2)$$

and the flux jacobians

$$A^{+} = \left(\frac{\partial F^{+}}{\partial Q}\right)^n \qquad (3)$$

$$A^{-} = \left(\frac{\partial F^{-}}{\partial Q}\right)^n$$

$$B^{+} = \left(\frac{\partial G^{+}}{\partial Q}\right)^n$$

$$\vdots$$

with

$$R^n = (\delta_i F^{+} + \delta_i F^{-} + \delta_j G^{+} + \delta_j G^{-} + \delta_k H^{+} + \delta_k H^{-}) \qquad (4)$$

for inviscid flow analysis, or

$$R^n = (\delta_i F^{+} + \delta_i F^{-} + \delta_j G^{+} + \delta_j G^{-} + \delta_k H^{+} + \delta_k H^{-} + S^n) \qquad (5)$$

where

$$S^n = (\delta_i F_v + \delta_j G_v + \delta_k H_v) \qquad (6)$$

for viscous flow analysis. For a detailed review of $F^+$, $F^-$, $G^+$,...etc., the reader is referred to Refs. 2 and 3. $F_v$, $G_v$, and $H_v$ (not mentioned in Refs. 2 and 3) represent the effects of the viscous stresses acting on a cell. They are implemented in an explicit fashion and have been shown not to markedly affect the speed of the baseline algorithm.

In the interest of limited computational funds, it was deemed impractical to operate with the viscous stresses activated when running the test cases presented herein. This is solely due to an inabililty to adequately resolve a flowfield when restricted to coarse mesh funding. It should be noted that the additional computations per cell introduced by the viscous stresses are minimal.

Algorithm Improvement

The matrix structure of Eq.(1) tends to be cumbersome and difficult to invert, not to mention very costly. Presently, two well-known methods are used to approximate Eq.(1), approximate factorization (AF) and relaxation methods. As in Ref. 2, the AF approach is preferred to a relaxation method.

There exist several methods for approximately factoring Eq.(1). Some retain the full implicit nature of the system, yet they require relatively sophisticated solution algorithms. Others possess only a point implicit nature and lend themselves to much simpler solution algorithms. These methods are inevitably multi-step, upper or lower block triangular in matrix structure with the attractive feature of requiring point simultaneous solutions and backward or forward substitution.

Two of the attractive factoring schemes were scrutinized during the course of this project. The first of these is that which the baseline algorithm utilized as a solution procedure. It will be referred to here as the six factor(6F) scheme. This scheme involves the factorization of Eq.(1) based on the sign of each spatial direction in computational space yielding the following

$$(I+\Delta\tau\delta_i A^{+}_{\bullet})(I+\Delta\tau\delta_i A^{-}_{\bullet})(I+\Delta\tau\delta_j B^{+}_{\bullet})(I+\Delta\tau\delta_j B^{-}_{\bullet})(I+\Delta\tau\delta_k C^{+}_{\bullet})(I+\Delta\tau\delta_k C^{-}_{\bullet})\Delta Q^n=-\Delta\tau R^n \quad (7)$$

The solution of Eq. (7) can be carried out as follows:

$$(I + \Delta\tau\delta_i A^{+}_{\bullet})X^1 = -\Delta\tau R^n \quad (8a)$$

$$(I + \Delta\tau\delta_i A^{-}_{\bullet})X^2 = X^1 \quad (8b)$$

$$(I + \Delta\tau\delta_j B^{+}_{\bullet})X^3 = X^2 \tag{8c}$$

$$(I + \Delta\tau\delta_j B^{-}_{\bullet})X^4 = X^3 \tag{8d}$$

$$(I + \Delta\tau\delta_k C^{+}_{\bullet})X^5 = X^4 \tag{8e}$$

$$(I + \Delta\tau\delta_k C^{-}_{\bullet})X^6 = X^5 \tag{8f}$$

$$\Delta Q^n = X^6 \tag{8g}$$

An alternate factorization is based on the like sign of all three spatial directions in computational space yielding the following

$$[I + \Delta\tau(\delta_i A^{+}_{\bullet} + \delta_j B^{+}_{\bullet} + \delta_k C^{+}_{\bullet})][I + \Delta\tau(\delta_i A^{-}_{\bullet} + \delta_j B^{-}_{\bullet} + \delta_k C^{-}_{\bullet})]\Delta Q^n = -\Delta\tau R^n$$

This factoring scheme is referred to as the two factor(2F) scheme, the solution of which can be carried out as follows

$$[I + \Delta\tau(\delta_i A^{+}_{\bullet} + \delta_j B^{+}_{\bullet} + \delta_k C^{+}_{\bullet})]X^1 = -\Delta\tau R^n \tag{9a}$$

$$[I + \Delta\tau(\delta_i A^{-}_{\bullet} + \delta_j B^{-}_{\bullet} + \delta_k C^{-}_{\bullet})]X^2 = X^1 \tag{9b}$$

$$\Delta Q^n = X^2 \tag{9c}$$

The error introduced via the 6F scheme has the form

$$\{\Delta\tau^2\delta_i A^{+}_{\bullet}\delta_i A^{-}_{\bullet}(I + \alpha)(I + \beta) + [\Delta\tau(\delta_i A^{+}_{\bullet} + \delta_i A^{-}_{\bullet})][\beta + \alpha(I + \beta)] +$$

$$\Delta\tau^2\delta_j B^{+}_{\bullet}\delta_j B^{-}_{\bullet}(I + \beta) + [\Delta\tau(\delta_j B^{+}_{\bullet} + \delta_j B^{-}_{\bullet})]\beta + \Delta\tau^2\delta_k C^{+}_{\bullet}\delta_k C^{-}_{\bullet}\}\Delta Q^n \tag{10}$$

where

$$\alpha = [\Delta\tau(\delta_j B^{+}_{\bullet} + \delta_j B^{-}_{\bullet}) + \Delta\tau^2\delta_j B^{+}_{\bullet}\delta_j B^{-}_{\bullet}] \tag{11a}$$

$$\beta = [\Delta\tau(\delta_k C^{+}_{\bullet} + \delta_k C^{-}_{\bullet}) + \Delta\tau^2\delta_k C^{+}_{\bullet}\delta_k C^{-}_{\bullet}] \tag{11b}$$

An examination of the terms indicates the error to be of higher order of accuracy than that of Eq.(1), consequently the overall order of the scheme is maintained. The error introduced via the 2F scheme has the form

$$[\Delta\tau(\delta_i A^{+}_{\bullet} + \delta_j B^{+}_{\bullet} + \delta_k C^{+}_{\bullet})][\Delta\tau(\delta_i A^{-}_{\bullet} + \delta_j B^{-}_{\bullet} + \delta_k C^{-}_{\bullet})]\Delta Q^n \qquad (12)$$

Inspection of this error also indicates it to be of higher order of acccuracy than Eq.(1), again the overall order of the scheme is maintained.

Judging from Eq. (10) and Eq. (12), it seems plausible to expect the 2F scheme to be somewhat superior to the 6F scheme. Although Eq. (1) represents an unconditionally stable scheme, due to the error introduced by the factoring process a restriction may be imposed on the time-step to maintain stability. A stability analysis of both schemes was presented by Anderson, et al.,[4] the results of which are given in Fig. 1. For the Courant numbers sampled, the figure indicates the 6F scheme to be more restrictive than the 2F scheme regarding maximum allowable time-step. It is encouraging to note that the 2F scheme seems to retain the favorable quality of unconditional stability.

There are advantages and disadvantages to both factoring schemes. For example, the 2F scheme tends to have an appetite for in-core memory stemming from the simultaneous storage of at least three flux jacobians, whereas the 6F scheme sequentially overwrites the same memory cells allocated for a single flux jacobian. Another example is the number of linear systems solved per cycle; the 6F scheme requires six compared to only two for the 2F scheme. Based on all the aspects of both schemes,

the 2F scheme seems to stand out as a superior factorization method. As a consequence of this analysis, the 2F scheme was implemented in all following results unless indicated otherwise.

Although the 2F scheme is superior to the 6F scheme, there is still the ever present AF error (Eq.(12)) introduced by this scheme. Presently, there is much interest regarding the elimination of this AF error. One method to eliminate the AF error of the 2F scheme is to alter the second step as follows

$$\{I + \Delta\tau[I + \Delta\tau(\delta_i A^{+}_{\bullet} + \delta_j B^{+}_{\bullet} + \delta_k C^{+}_{\bullet})]^{-1}[\delta_i A^{-}_{\bullet} + \delta_j B^{-}_{\bullet} + \delta_k C^{-}_{\bullet}]\}X^2 = X^1 \quad (13)$$

Total elimination of the AF error is achieved by the premultiplication of the $\Delta\tau$ term in the second step by the inverse of the first step operator.

The difficulty with this theory is that it involves the inversion of a matrix of difference operators. An approximate version of this inverse method can be formulated by substituting the following matrix for the proposed inverse of the first step

$$[I + \Delta\tau(\delta_i A^{+} + \delta_j B^{+} + \delta_k C^{+})]^{-1} \quad (14)$$

Notice the operator dot ($\bullet$) has been removed leading to a method which could readily be investigated. Although Eq.(13) should completely eliminate the AF error, the approximate version (Eq.(14)) falls short of this goal. Evidently Eq.(14) retains enough AF error to adversely affect the solution process. This method, when condensed to a single line equation, closely mimics a relaxation scheme, which was avoided from the beginning.

Approximation and Acceleration

The previous section was concerned with reducing or eliminating the error introduced as a consequence of factoring Eq.(1). This section approaches the problem from what could be considered an opposite viewpoint, i.e., what can one get away with and still obtain an acceptable solution. The primary objective here is to shorten and/or simplify the path to the desired solution.

First, consider a method which substitutes the true flux jacobians (TFJ) given by Eq.(3) with their corresponding spectral radii. A matrix of the following form is used in lieu of the TFJ matrices

$$\lambda_{SR,K}\, I \tag{15}$$

where

$$\lambda_{SR,K} = \max\left(|\lambda_K^i|\right) \tag{16}$$

with

$$\lambda_K^i = \text{eigenvalues of } K, \qquad i = 1,2,3,4,5$$

and K is either $A^+$, $A^-$, $B^+$, $B^-$, $C^+$ or $C^-$. In Eq. (15), I has the usual meaning of the identity matrix.

The inclusion of the matrices given by Eq.(15) yields the following

$$[I + \Delta\tau(\delta_i \lambda_{SR,A^+} I\cdot + \delta_j \lambda_{SR,B^+} I\cdot + \delta_K \lambda_{SR,C^+} I\cdot)]\ X^1 = -\Delta\tau R^n \tag{18a}$$

$$[I + \Delta\tau(\delta_i \lambda_{SR,A^-} I\cdot + \delta_j \lambda_{SR,B^-} I\cdot + \delta_K \lambda_{SR,C^-} I\cdot)]\ X^2 = X^1 \tag{18b}$$

$$\Delta Q^n = X^2 \tag{18c}$$

Equations (18) represent an uncoupled system thus eliminating the need for the simultaneous solution at a point. The solution process is simplified to that of a backward and a forward substitution. Thus far everything sounds great, the drawback happens to be multi-faceted.

The first difficulty is the isolation of the dominant eigenvalue. This can be accomplished using an iterative scheme referred to as the power method.[5] Although it is a simple iterative routine, it tends to be costly. Secondly, the stability of the spectral radius method is much more restrictive than that of the 2F scheme, as expected. This degradation in stability imposes a stringent time-step limitation. Together, the two drawbacks cited are enough to exclude the spectral radius method as an integral part of the solution algorithm.

Another approximation technique involves the infrequent updating of the flux jacobians (flux jacobian freezing). The method is quite simple assuming a large quantity of in-core memory is available (or rapid transfer out-of-core storage devices are accessible). The basic approach is to calculate, store, and periodically update the flux jacobian matrices. The period of acceptable flux jacobian "freezing" varies from flow to flow. In many instances (steady state solutions), the flux jacobians need only be calculated once.

This approach of flux jacobian freezing cuts per cycle computation time nearly in half depending, of course, on the frequency of updating. Although run time is significantly reduced, the storage requirement is doubled. The availability of abundant Cray SSD storage encouraged the inclusion of this approach as an integral part of the solution algorithm.

The final topic in this section, the use of multiple grids, is both an approximation and an acceleration technique. The multi-grid technique used here is that of a full approximation scheme (FAS). The theory is that the low frequency (LF) errors which are not damped well by the solution algorithm on a fine grid appear as high frequency (HF) errors on coarser grids and subsequently can be damped out on these grids. This is assuming, of course, that the solution algorithm damps HF errors well. This scheme can be applied to progressively coarser grids to damp most of the components of the error. For an in-depth study of the FAS method as it is applied to a similar finite volume code, the reader is referred to the work of Anderson.[6]

The basic approach involves solving an error equation on increasingly coarse meshes. This error equation and background material are given in Ref. 6 thus they will not be repeated here. The coarsening can be accomplished in a variety of ways. The simplest procedure involves starting with a fine grid then eliminating every other grid point in each computational direction to construct a coarser grid. The use of multi-grid requires increasing the total storage by no more than 25%. Also, the computations on the coarser grids are relatively insignificant because each coarse grid contains only 1/8 as many points as it's parent grid.

Reference 6 gives a detailed outline of the steps involved in the FAS multi-grid V cycle which is used in this study. Various investigative studies were performed to access the benefits derived from the multi-grid procedure. An ONERA M6 wing (49x9x9) was used as a test case to examine the multi-grid characteristics; such as, acceleration of con-

vergence, additional memory penalties, runtime per cycle per grid point, etc. In Fig. 2, the effect of the multi-grid procedure on reducing the number of cycles to reach a given level of convergence is shown. The asymptotic spectral radius (convergence rate) for the single grid was -.95, whereas that using a 2-level multi-grid V cycle with 3 iterations on the coarse grid was - .87. From the data collected, a computational savings of 32% was realized using the multi-grid procedure.

Thus far, it is evident that multi-grid is beneficial for steady-state solutions. It also has applications to unsteady solutions. Jesperson has successfully demonstrated the use of a time-accurate multi-grid algorithm.[7] Recently, significant savings have been realized using a time-accurate multi-grid approach for a harmonically plunging airfoil.[8] It is expected that multi-grid will become a useful aspect of rotor-stator flow analysis, consequently it has been included in the solution algorithm.

Coding Developments

The following section deals with computer algorithm modifications which do not alter the solution, yet represent significant advances. Some of these were developed by colleagues and integrated into the flow code. Several less significant code modifications have also been implemented over the past year. With the exception of the freezing of the flux jacobians, code developments have resulted in the largest reduction in code length and execution time.

The first of the significant developments is referred to here as ribbon vector dynamic allocation, a concept developed by Belk.[3] The principle of ribbon vector storage is frequently used to compress storage to avoid wasting memory space (achieve a minimum core requirement). It functions well with modular codes because the bulk memory requirement can be allotted in the main calling routine, and floating dimensions can be used in the subroutines. Belk has demonstrated the use of this principle, coupled with the flexibility of the Cray to allow user defined dynamic memory management, to enable a dynamic mesh size. The plan is to use this in connection with a multi-block procedure to minimize in-core requirements. A version similar to that proposed by Belk has been integrated into this solution algorithm.

This leads to another concept exhaustively investigated by Belk. It involves the segmenting of the computational domain into relatively compact computational blocks. The impetus for such an approach emanates from the difficulties encountered when generating a single grid about a complex geometrical shape such as a wing-pylon-store or the blades in rotating machinery. An additional attractive feature of multi-block is

the ability to sequentially solve several smaller flowfields comprising one larger, nearly insurmountable flowfield. As mentioned before, this section involves coding mods which do not alter the solution. Frequently, some slight degradation of a solution is acceptable when dealing with multiple block solutions.

Belk has shown that some liberties can be taken regarding the inter-block transfer of data while still maintaining acceptable solution development. Based on these findings, a multi-block procedure is expected to function well in the analysis of a rotor-stator configuration. Related work has indicated that multi-block dynamic grid procedures are not difficult to implement. The detailed analysis of this procedure is on-going relative to its quantitative results and the influence of inter-block communication in rotating machinery analysis.

The final contribution to code development was an attempt to optimize the solution procedure of the 2F scheme. The 2F scheme involves a point simultaneous solution and backward (or forward) substitutions. The substitution portion poses quite a problem when attempting to utilize vector hardware to it's fullest. The difficulty, referred to as a vector dependency[9] (or a recursion problem), occurs when all the elements of a computational vector are not independent. For example, the equation

$$X_i = X_{i-1} \qquad i = 1,\ldots,N_i \tag{19}$$

involves a vector dependency when coded, because the $i^{th}$ element depends on the $(i - 1)^{th}$ element. A method has been developed in order to circumvent this problem in a multi-dimensional space. The concept of

this method was originally conceived by Belk[10] and has no previous publication.

To begin, consider the example of a two-dimensional space in Fig. 3 and the following equation

$$X_{i,j} = X_{i-1,j} + X_{i,j-1} \qquad i=1,\ldots,N_i,\ j=1,\ldots N_j \tag{20}$$

If one attempts to logically construct computational vectors using the elements lying on any computational line, as shown, a vector dependency will be present. Thus the equation can only be solved in a much slower scalar mode. Now consider Fig. 4 and Eq. (20). If computations are carried out along the computational vectors (diagonal lines) as shown, the equation can be solved in a vector mode. This is because all the elements in a single computational vector are independent of one another. The elements of each computational vector are given by the line

$$i + j = C \tag{21}$$

where C is the vector number, ranging from 0 to $(N_i + N_j)$. In addition, the vector length varies from 1 to $\min\{(N_i+1),(N_j+1)\}$. This is a very simple procedure in two dimensions, now for the easy extension to three-dimensions.

Consider the three-dimensional analogy to Eq. (20)

$$X_{i,j,k} = X_{i-1,j,k} + X_{i,j-1,k} + X_{i,j,k-1} \qquad \begin{matrix} i = 1,\ldots,N_i \\ j = 1,\ldots,N_j \\ k = 1,\ldots,N_k \end{matrix} \tag{22}$$

Referring to Fig. 5, a similar procedure can be followed in order to vectorize a three-dimensional problem. The points lying on a diagonal plane constitute the computational vector with independent elements. These points are given by the equation of the plane

$$i + j + k = C \tag{23}$$

where C is the vector number, ranging from 0 to $(N_i + N_j + N_k)$. In addition, the vector length varies from 1 to $\min\{(N_i+1)(N_j+1),(N_i+1)(N_k+1),(N_j+1)(N_k+1)\}$. This procedure for vectorizing a backward (or forward) substitution has become an integral part of the solution algorithm. A timing analysis is difficult to obtain due to the maximum vector length depending on the mesh structure. It is anticipated that the significance of the diagonal plane method will increase as the mesh sizes increase.

Results

The year's progress can be accessed by comparing the status of the code of September 1985 to that of September 1986. The comparison can be made using the ONERA M6 wing (49x9x9) case presented in Table 1 of Ref. 2. A comparable flow trace was generated using the modified version of BMULE, refer to Table 1b. Although a direct comparison cannot be made regarding individual subroutines, a direct comparison can be made between subroutine groups serving the same function. Consider, for example, the following list of modified BMULE subroutines and the corresponding BMULE counterparts.

| September 1986 | September 1985 | Performance Ratio $\frac{1986}{1985}$ |
|---|---|---|
| METRIC | METRIC | .86 |
| BC | BC | 2.09 |
| RESID + -(3x9x9) x FLUX | FLUX | .43 |
| TSTEP | EIGENV | 1.14 |
| 6 X CMAT | 2 x AE<br>2 x BE<br>2 x CE | 1.00 |
| 2 x DOO | DOOIF<br>DOOIB<br>DOOJF<br>DOOJB<br>DOOKF<br>DOOKB | .25 |

The performance ratios should be examined in light of the percent of total runtime spent in each subroutine, given in Tables 1. This is accounted for when one examines the overall code performance ratio, found to be .21. Thus the modified version operates nearly 5 times as efficient as the previous version. This is mainly attributed to flux jacobian freezing and the diagonal plane solution path. The code now operates at $2.1 \times 10^{-5}$ seconds per grid point per cycle.

The code was used to analyze the flowfield of a typical cascade geometry.[11] Fig. 6 gives a view of the H-mesh used in this test case. Figs. 7 and 8 are Mach and pressure contour plots, respectively, of the flowfield resolved by the flow code. The qualitative agreement represented in these figures is good. No quantitative solutions are presented here because the final solution of the modified BMULE code is the same as that of the original code, whose validity was established in Ref. 2.

## Conclusions

Several modifications were made to an operational implicit upwind finite volume computer algorithm. These include: ribbon vector storage, two-factor scheme, multigrid, diagonal plane solution path, etc. The goal of improving computational efficiency has been modestly achieved, although more improvement is necessary. The present computer code operates 5 times more efficiently than it did previously. Advanced CFD procedures have been integrated to produce a robust, accurate and efficient algorithm with the potential to provide advanced three-dimensional flow analysis of rotating machinery.

Future emphasis should continue to be focused toward improving computational efficiency. At the present rate of improvement, the analysis of rotating machinery flowfields is well within view. Once the efficiency is achieved, then and only then, should detailed flow studies be expected.

The modified version of the BMULE code presently resides on the NASA Marshall CRAY X-MP.

References


1. Chapman, D.R., "Computational Aerodynamics Development and Outlook", AIAA Journal, Vol. 17, No. 12, 1979.

2. Whitfield, D.L., "Implicit Upwind Volume Schemes for the Three-Dimensional Euler Equations," Mississippi State University Report MSSU-EIRS-ASE-85-1, September 1985.

3. Belk, D.M., "Three-Dimensional Euler Equations Solutions on Dynamic Blocked Grids," Ph.D. Dissertation, Mississippi State University, August 1986.

4. Anderson, W.K., Thomas, J.L., and Whitfield, D.L., "Multigrid Acceleration of the Flux Split Euler Equations," AIAA Paper 86-0274, January 1986.

5. Burden, Richard L. and J. Douglas Faires, Numerical Analysis, PWS Publishers, Boston, Massachusetts, 1985.

6. Anderson, W.K., "Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations," Ph.D. Dissertation, Mississippi State University, August 1986.

7. Jespersen, D.C., "A Time-Accurate Multiple-Grid Algorithm," AIAA 7th Computational Fluid Dynamics Conference Proceedings, Cincinnati, Ohio, pp. 58-66, July 1985.

8. Anderson, W.K., Private Communication.

9. CRAY X-MP and CRAY-1 Computer Systems (Fortran (CFT) Reference Manual SR-0009), Cray Research, Inc., Mendota Heights, Minnesota, 1984.

10. Belk, D.M., Private Communication.

11. Swafford, T.W., Private Communication.

Table 1a. Flow Trace September 1985[2]

| | ROUTINE | TIME | % | CALLED | AVERAGE T |
|---|---|---|---|---|---|
| 1 | BMULEWR | 0.306154 | 0.70 | 1 | 0.306154 |
| 2 | IC | 0.001208 | 0.00 | 1 | 0.001208 |
| 3 | METRIC | 0.010898 | 0.02 | 1 | 0.010898 |
| 4 | BC | 0.275058 | 0.63 | 101 | 0.002723 |
| 5 | STEP | 0.351348 | 0.80 | 100 | 0.003513 |
| 6 | FLUX | 6.024503 | 13.72 | 100 | 0.060245 |
| 7 | EIGENV | 0.337293 | 0.77 | 100 | 0.003373 |
| 8 | AE | 5.961722 | 13.58 | 200 | 0.029809 |
| 9 | DOOIF | 4.540607 | 10.34 | 100 | 0.045406 |
| 10 | DOOIB | 4.611240 | 10.50 | 100 | 0.046112 |
| 11 | BE | 6.575230 | 14.97 | 200 | 0.032876 |
| 12 | DOOJF | 2.043788 | 4.65 | 100 | 0.020438 |
| 13 | DOOJB | 2.129168 | 4.85 | 100 | 0.021292 |
| 14 | CE | 6.588037 | 15.00 | 200 | 0.032940 |
| 15 | DOOKF | 2.021897 | 4.60 | 100 | 0.020219 |
| 16 | DOOKB | 2.129334 | 4.85 | 100 | 0.021293 |
| 17 | PVAR | 0.007076 | 0.02 | 1 | 0.007076 |
| *** | TOTAL | 43.914561 | | | |
| *** | OVERHEAD | 0.051794 | | | |

Table 1b. Flow Trace September 1986

| | ROUTINE | TIME | % | CALLED | AVERAGE T |
|---|---|---|---|---|---|
| 1 | BULLYII | 0.000213 | 0.00 | 1 | 0.000213 |
| 2 | POINTER | 0.000077 | 0.00 | 1 | 0.000077 |
| 3 | SETMEM | 0.007671 | 0.08 | 1 | 0.007671 |
| 4 | INDEX | 0.000360 | 0.00 | 102 | 0.000004 |
| 5 | IC | 0.000888 | 0.01 | 1 | 0.000888 |
| 6 | GRID | 0.017251 | 0.19 | 1 | 0.017251 |
| 7 | METRIC | 0.009412 | 0.10 | 1 | 0.009412 |
| 8 | BC | 1.146532 | 12.51 | 201 | 0.005704 |
| 9 | DPMAP | 0.063612 | 0.69 | 100 | 0.000636 |
| 10 | STEP | 0.557665 | 6.09 | 100 | 0.005577 |
| 11 | RESID | 0.602808 | 6.58 | 100 | 0.006028 |
| 12 | FLUX | 1.676988 | 18.30 | 20800 | 0.000081 |
| 13 | TSTEP | 0.383998 | 4.19 | 100 | 0.003840 |
| 14 | CMAT | 0.191312 | 2.09 | 6 | 0.031885 |
| 15 | DOO | 4.435661 | 48.40 | 200 | 0.022178 |
| 16 | UPDATE | 0.064985 | 0.71 | 100 | 0.000650 |
| 17 | PVAR | 0.005143 | 0.06 | 1 | 0.005143 |
| ***TOTAL | | 9.164578 | | | |
| ***OVERHEAD | | 0.593479 | | | |

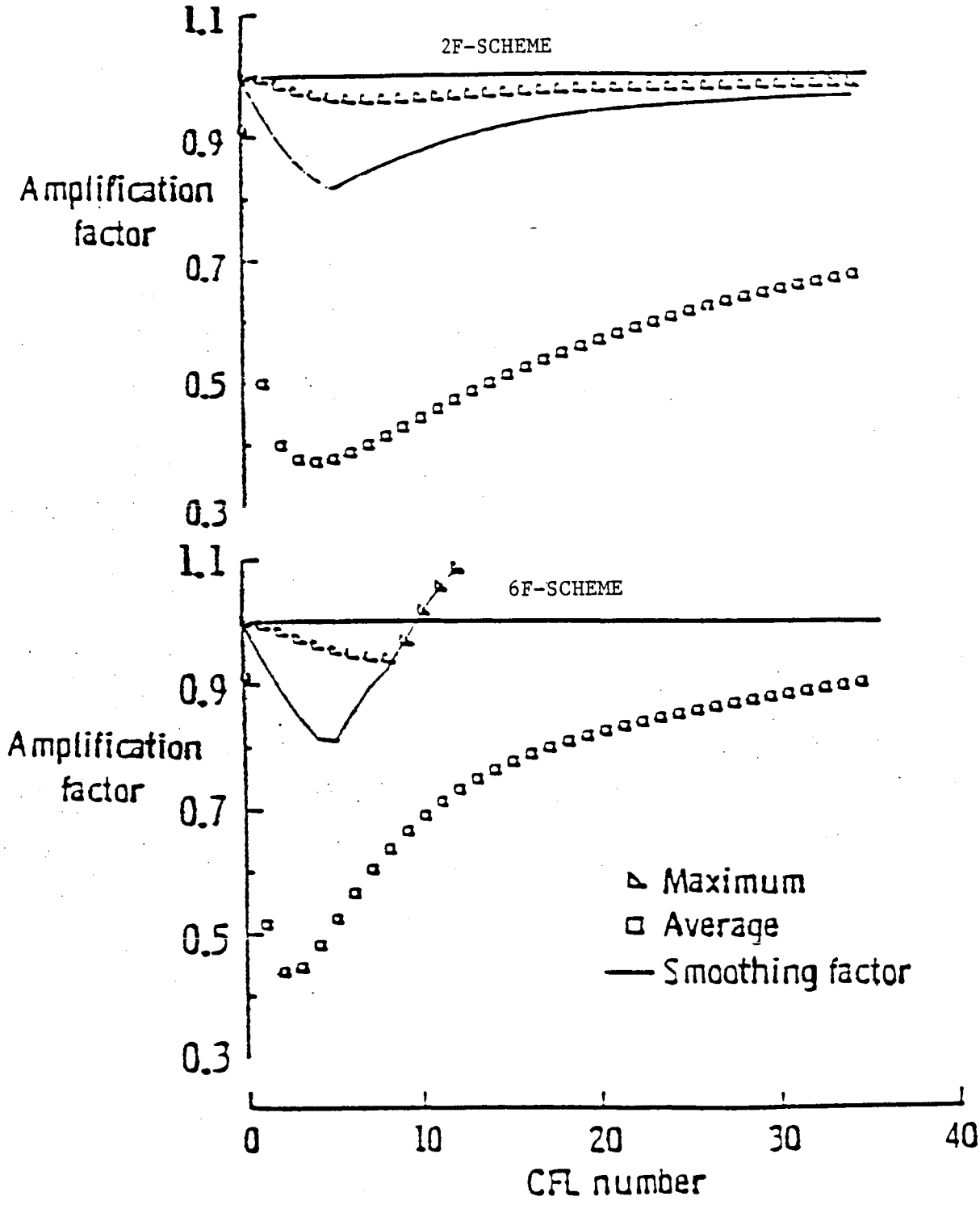


Fig. 1 Stability Analysis[4]

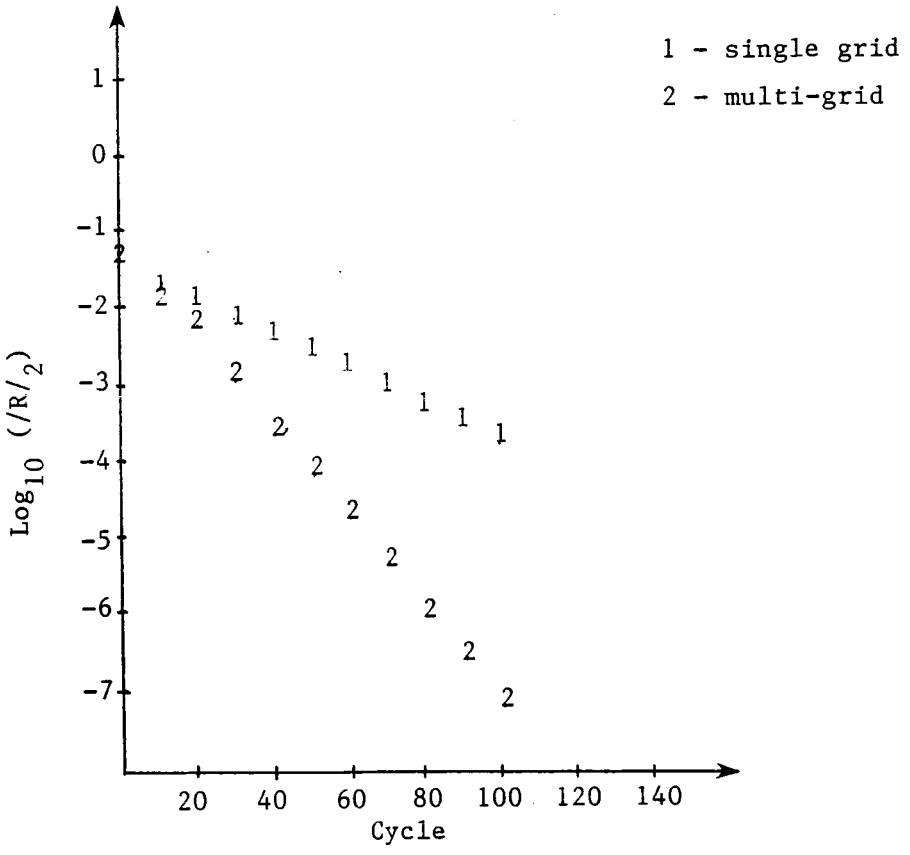


Fig. 2 Residual Plot (49x9x9)

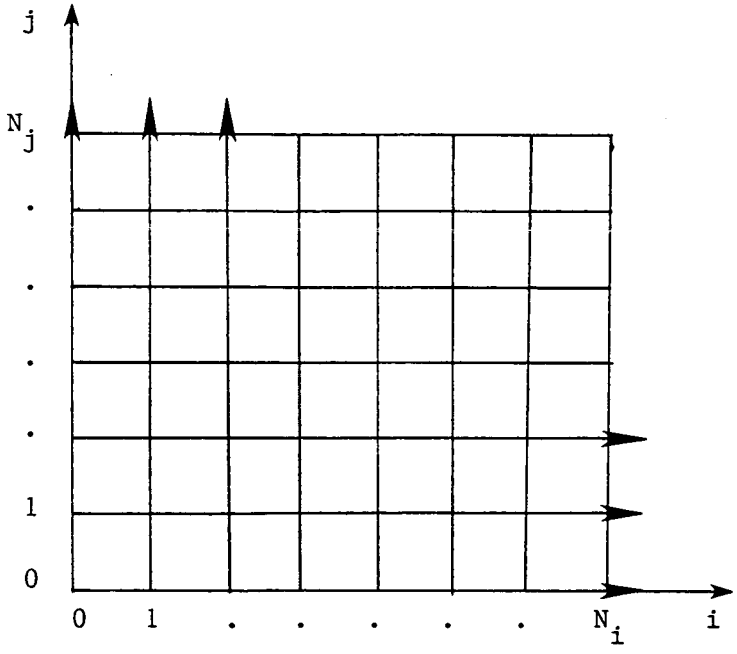


Fig. 3 Two-Dimensional Dependent Computational Vectors

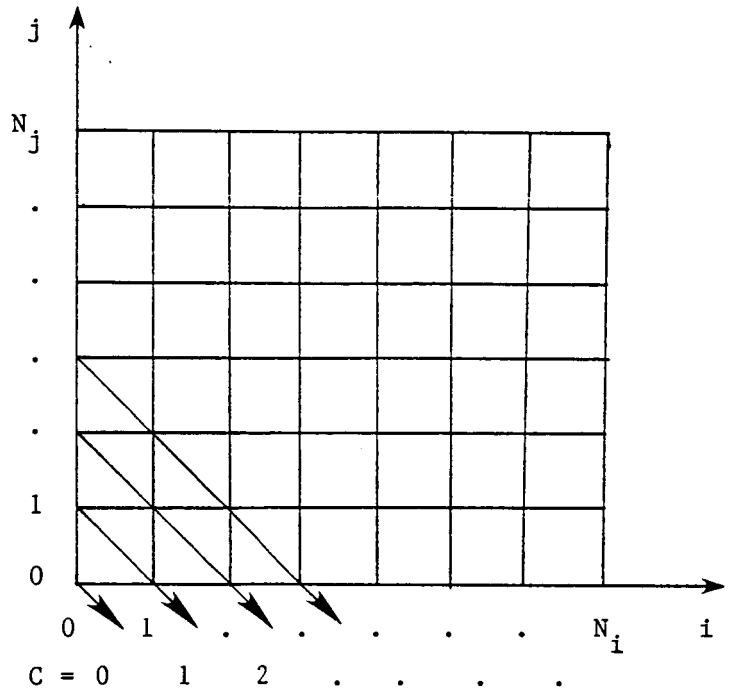


Fig. 4 Two-Dimensional Independent Computational Vectors

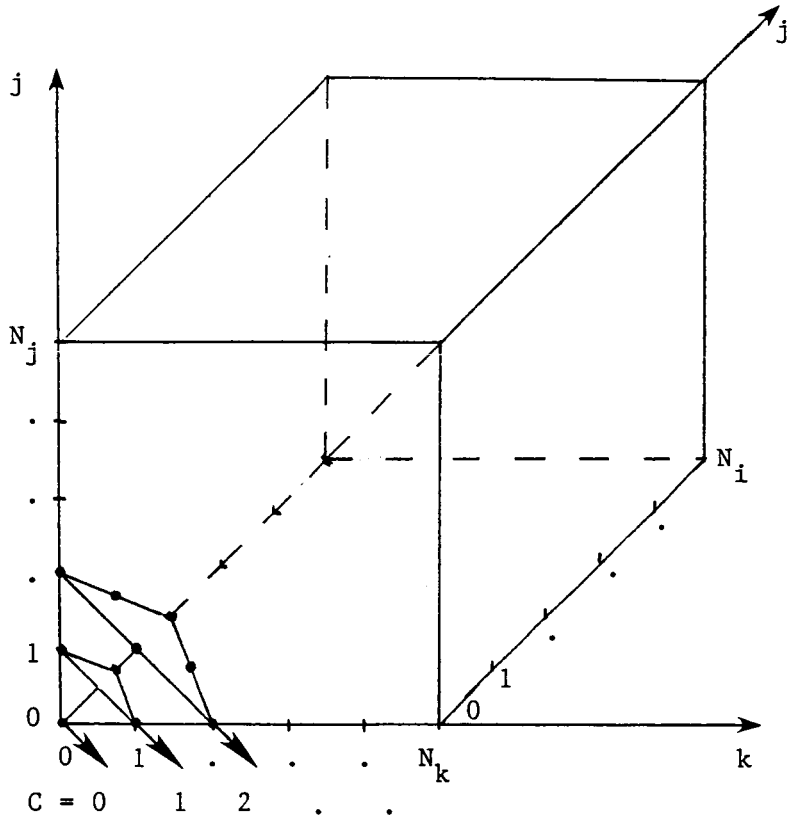


Fig. 5 Three-Dimensional Independent Computational Planes

ORIGINAL PAGE IS
OF POOR QUALITY

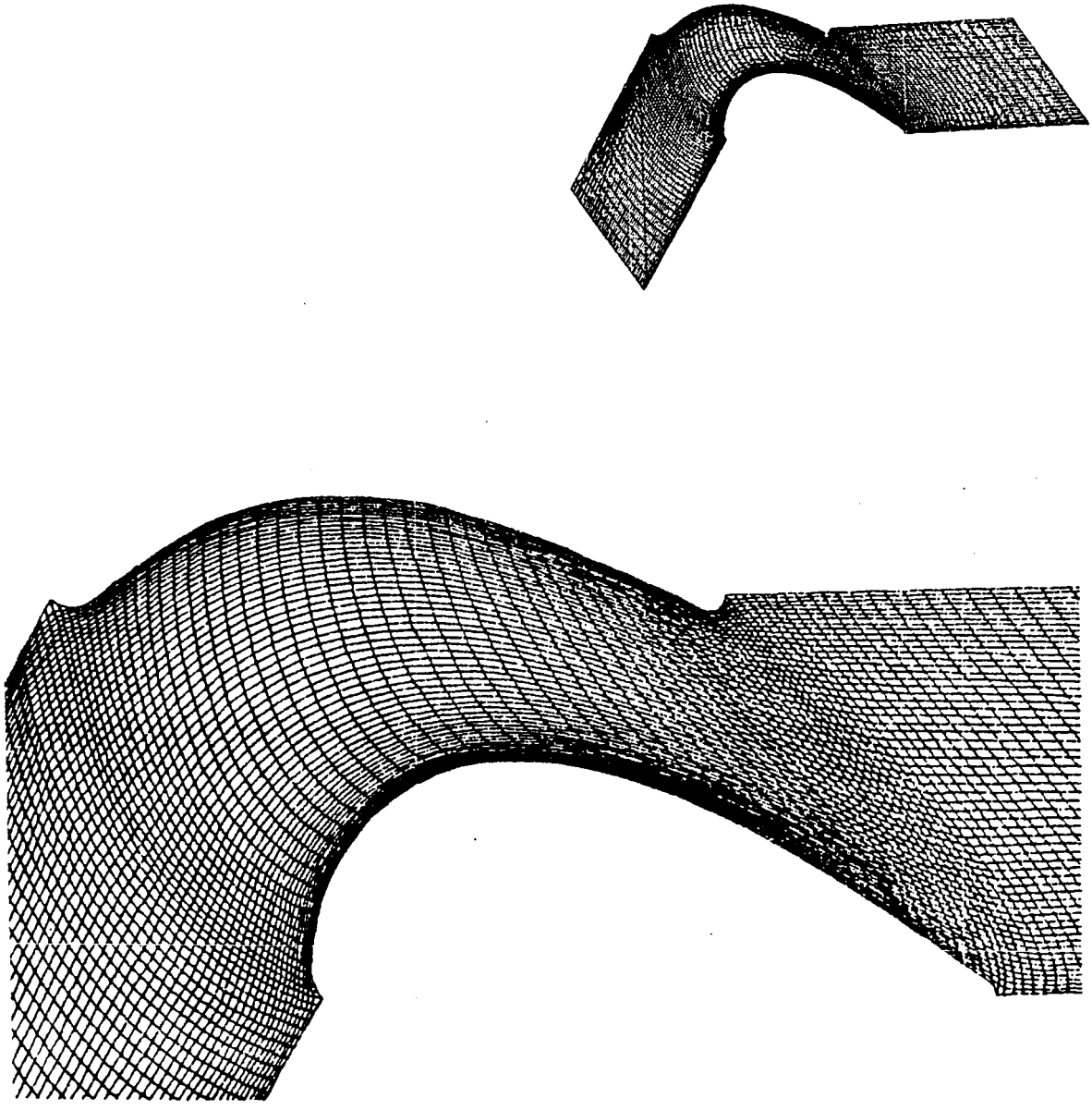

Fig. 6 Typical Cascade H-Mesh [11]

ORIGINAL PAGE IS
OF POOR QUALITY

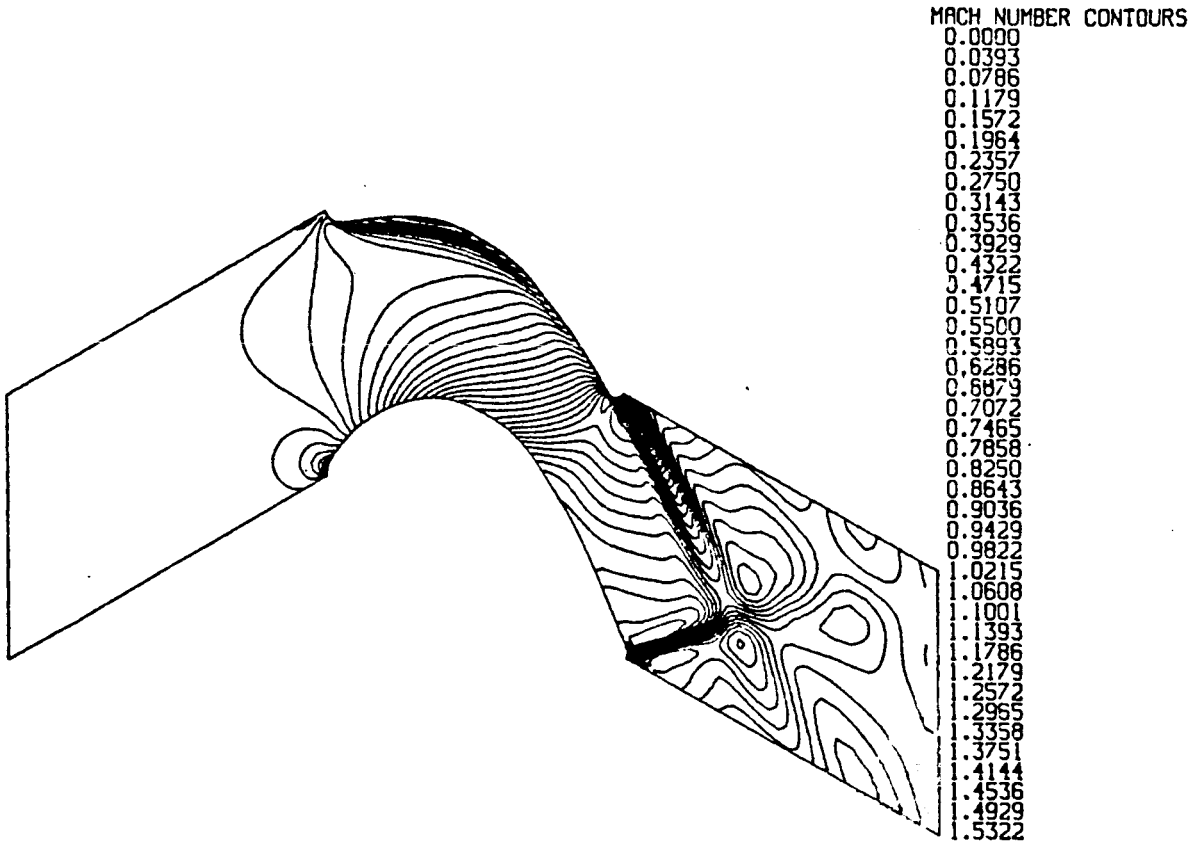


Fig. 7 Mach Contours[11]

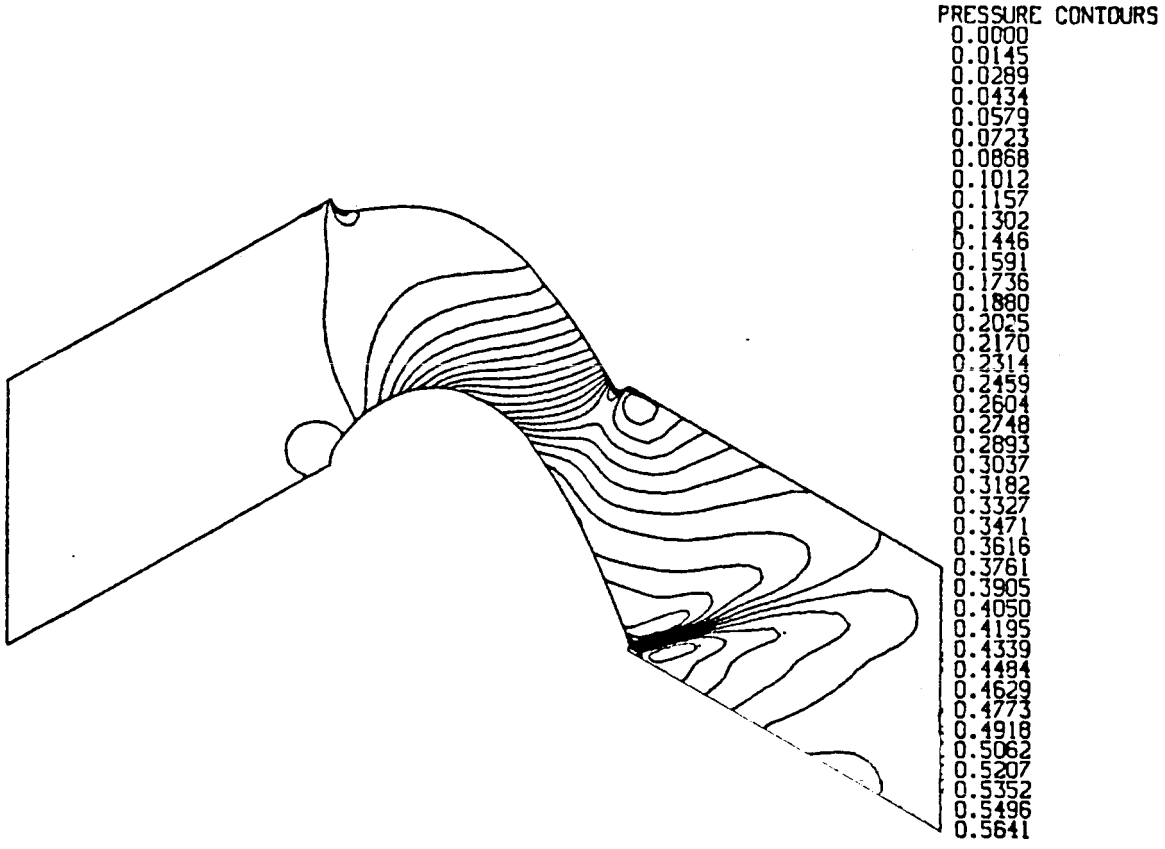


Fig. 8 Pressure Contours [11]