XI CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA - SÃO PAULO, SP - BRASIL (DEZEMBRO, 1991)
11th ABCM MECHANICAL ENGINEERING CONFERENCE – SÃO PAULO, SP – BRAZIL (DECEMBER, 1991)

ON THE EFFECTS OF VECTORIZATION FOR EFFICIENT COMPUTATION OF THREE DIMENSIONAL SEGREGATED FINITE VOLUME SOLUTIONS

A.F.C. SILVA, C.H. MARCHI, M.A. LIVRAMENTO
Universidade Federal de Santa Catarina
Departamento de Engenharia Mecânica
88049 — Florianópolis — SC

J.L.F. AZEVEDO
Centro Técnico Aeroespacial
Instituto de Aeronáutica e Espaço
12225 — São José dos Campos — SP

SUMMARY

Segregated finite volume methods have become quite an attractive tool for the solution of complex fluid flow problems. For large, three dimensional CFD problems, a very significant portion of the total computational time is spent in the solution of linear systems. The present work studies the effect of vectorization on the efficiency of a few linear system solvers typically used in the context of these segregated finite volume flow computations. The results here obtained indicate that strongly implicit procedures, which are typically the most efficient in scalar mode, can be less efficient than fully explicit algorithms when vectorization is taken into account. The influence of the size of the problem on the gains obtained from vectorization is also investigated and discussed.

INTRODUCTION

Computational Fluid Dynamics has typically been an area of work which has taken advantage of all recent progresses in computer architecture and hardware in a continuous search for improved computational performance. One aspect of the problem is the use of vectorization. It is possible to show that for large, complex, three-dimensional CFD problems, a very significant portion of the total computational time is spent in the solution of linear systems. Therefore, the use of linear system solvers that can benefit from vectorization can have a substantial impact on the computational cost of these CFD solutions. This has motivated the authors to study the effects of vectorization on the performance of a few methods used for the solution of the linear systems that arise in the computational treatment of fluid flow problems.

The physical problems considered here concern the three dimensional, supersonic, flow over the first Brazilian satellite launcher, the VLS. A segregated finite volume technique in generalized coordinates is used to discretize the governing equations. In the three dimensional case, this leads to the need of "inverting" heptadiagonal matrices with non-adjacent diagonals. Three methods were used in the present work for the solution of these linear systems. These were the modified strongly implicit (MSI) procedure (Zedan and Schneider, 1983), the alternating direction implicit (ADI) algorithm, and the point Jacobi scheme. The MSI algorithm is characterized by strongly recursive procedures, therefore allowing very little vectorization. The ADI algorithm cannot be vectorized in the direction being swept, but it can always be vectorized in the other two directions. The Jacobi scheme is fully vectorizable since it is a completely explicit method.

The concept of efficiency used in the present work is associated with the computational time necessary in order to obtain the solution of a given problem to a desired accuracy. Among the factors investigated, we included the size of the problem, i.e., the number of computational volumes considered, and its effect on the relative efficiency of the various methods. Moreover, the allowable range of time-steps that each method could stand was also considered.

PROBLEM STATEMENT AND SOLUTION METHOD

The inviscid, three dimensional, supersonic flow of a perfect gas over the nose fairing of the VLS vehicle is used as the physical test problem for the present work. The flow is, then, assumed to be governed by the Euler equations, which can be written for a general curvilinear coordinate system (ξ, η, ζ) as

$$\frac{1}{J}\frac{\partial}{\partial t}(\rho\phi) + \frac{\partial}{\partial \xi}(\rho U\phi) + \frac{\partial}{\partial \eta}(\rho V\phi) + \frac{\partial}{\partial \zeta}(\rho W\phi) = \hat{S}^{\phi} \qquad (1)$$

The general form given in Eq. (1), with the appropriate source term $\hat{S}^{\phi}$, can recover the continuity equation, the three momentum equations, and the energy equation. For that, ϕ must be chosen as 1, u, v, w, and T, respectively. J is the Jacobian of the coordinate transformation, and U, V and W are the contravariant velocity components. The system of 5 partial differential equations represented by Eq. (1), plus an equation of state, form a closed set of 6 equations for the 6 unknowns (ρ, u, v, w, p, and T).

Since we are considering low enough angles of attack, the flowfield is symmetric about the pitch plane. Therefore, the computational solution domain spans only 180° in the circumferential direction around the vehicle, going from the leeward to the windward plane. Flow symmetry conditions are, then, imposed at both lee- and windward planes. At the body surface, a flow tangency condition is enforced and the wall is assumed to be adiabatic. Freestream conditions are prescribed at the computational entrance surface. At the exit plane, since we are considering supersonic flow cases, all properties are obtained by extrapolation of interior information. At the upstream stagnation line, it turns out that no numerical boundary conditions are actually required. Freestream conditions are used as initial conditions for the simulations performed here.

Eq. (1) is discretized using a control volume method (Patankar, 1980). The mass conservation equation is linearized in such a way that maintains both density and velocity as unknowns (van Doormall, 1985), therefore allowing the solution of incompressible as well as compressible flows. Through the use of the SIMPLEC method (van Doormall and Raithby, 1984) for the pressure-velocity coupling, the continuity equation is used for the calculation of pressure, the equation of state is used to obtain the density, and the three momentum equations plus the energy equation are used in order to obtain the other quantities (u, v, w and T). A colocated variable arrangement is employed in the present work. Further details of the numerical methodology used here can be seen in Marchi et al. (1989, 1990).

Once initial values for the six state variables are known, the solution procedure adopted in the present work takes the following steps. **1.** Estimate of the u, v, w, p, T and ρ fields at instant $t + \Delta t$. **2.** Computation of the coefficients for the three momentum equations. **3.** Computation of the coefficients for the continuity equation. **4.** Computation of the $\hat{S}^{\phi}$ source terms for u, v and w. **5.** Solution of the momentum equations. This step determines new velocity components u^*, v^* and w^* which do not necessarily conserve the mass. **6.** Evaluation of the contravariant velocity components U^*, V^* and W^*. **7.** The error, or the residue, in the continuity equation is computed using the available contravariant velocity components and density field. **8.** A correction to the pressure field is determined using the coefficients evaluated in step (3) and the residues determined in step (7). **9.** Velocity components and densities are corrected by the new pressure field. The resulting fields conserve mass. **10.** Computation of the coefficients and source term for the energy equation. **11.** Calculation of a new temperature field. **12.** Computation of the density as function of pressure and temperature. **13.** Return to step (1) and iterate until the steady state is reached.

The solution process, as presented above, does not involve any iteration cycle within each time interval. However, due to the type of linearization adopted and due to the coupling scheme implemented, some steps must be executed more than once within each time step. In the present work, the computations associated with steps (3) through (9) were usually executed twice for each time interval. We shall refer to this situation stating that $ITM = 2$, i.e., two evaluations, or two iterations, of the continuity equation coefficients for each time interval. This inner iteration cycle, which has no meaning for incompressible problems, allows the use of larger time steps. Finally, we must emphasize that steps (5), (8) and (11) involve the solution of linear systems.

The iterative procedure is performed until

$$\frac{|p - p^0|}{p_{max} - p_{min}} \leq TOLP \tag{2}$$

for all volumes in the computational domain. Here, p is the pressure at the present time instant, p^0 is the pressure at the previous instant, p_{max} and p_{min} are, respectively, the maximum and minimum values of pressure throughout the field, and $TOLP$ is the specified tolerance. Two values of $TOLP$ were used in the present work. For the tests using a computational mesh with 7200 volumes, we adopted $TOLP = 10^{-5}$. The convergence criterion was relaxed to 2×10^{-5} for the cases involving meshes with 28800 volumes.

METHODS FOR LINEAR SYSTEM SOLUTION

Before describing the different methods which will be used for the solution of the linear systems, it is convenient to discuss the structure of these system coefficient matrices. We can start by observing that the solution domain assumes the form of a parallelepiped in computational space. In this transformed space, the solution domain is discretized into NI volumes in the ξ-direction, NJ volumes in the η-direction and NK volumes in the ζ-direction. The total number of computational volumes is $NIJK = NI \times NJ \times NK$. Volumes are numbered as indicated in Figs. 1 and 2, where the $K = 1$ and $K = 2$ planes are shown.

The process of discretization of the governing equations, when applied to a volume centered at point P, yields

$$a_P\phi_P + a_e\phi_E + a_w\phi_W + a_n\phi_N + a_s\phi_S + a_d\phi_D + a_f\phi_F = b \tag{3}$$

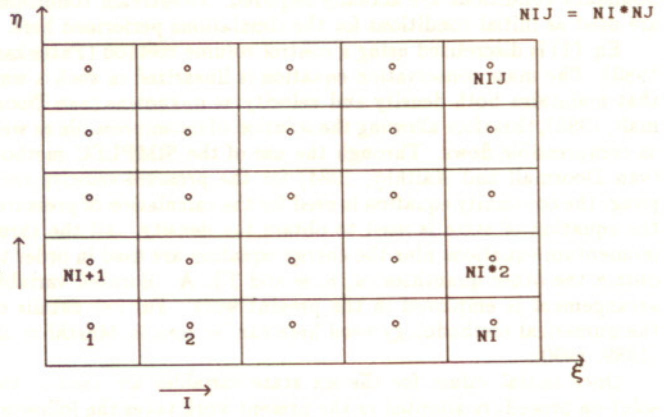

Figure 1: Plane $K = 1$ in computational space.

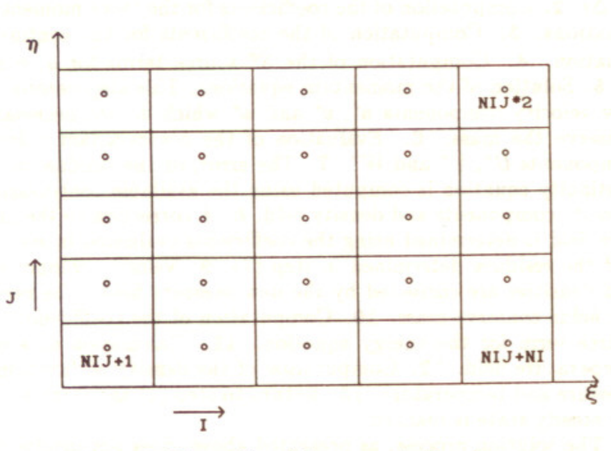

Figure 2: Plane $K = 2$ in computational space.

where the subscripts E (for "east"), N (for "north") and F (for "forward") indicate the neighboring volumes in the positive ξ, η and ζ-directions, respectively. In a similar fashion, the subscripts W, S and D indicate the corresponding volumes in the negative ξ, η and ζ-directions. Eq. (3), when applied to the $NIJK$ volumes, can be written in matrix notation as

$$[A]\,\{\phi\} = \{b\} \tag{4}$$

The form of the $[A]$ matrix is shown in Fig. 3. It is clear, therefore, that the $[A]$ matrix assumes a heptadiagonal structure with a bandwidth equal to $2 \times NIJ + 1$. Its solution by direct, i.e., non-iterative, methods is typically unfeasible for "real life" problems.

The Jacobi Scheme. The Jacobi method is a point iteration scheme in the sense that the new values of the variables for a given P volume are explicitly calculated using the values of the 6 neighboring volumes at the previous iteration level. Using Eq. (3), we can obtain that a generic variable ϕ at volume P can be computed from the following expression

$$\phi_P^{n+1} = \frac{-\sum a_{nb}\phi_{NB}^n + b}{a_P} \tag{5}$$

where

$$\sum a_{nb}\phi_{NB}^n = a_e\phi_E^n + a_w\phi_W^n + a_n\phi_N^n + a_s\phi_S^n + a_d\phi_D^n + a_f\phi_F^n \tag{6}$$

The Jacobi method is easy to implement and it requires only one additional auxiliary array with dimension $NIJK$, which is used to store the property value at the previous iteration, i.e., ϕ^n. Since it is fully explicit, the Jacobi scheme allows for total vectorization.

The ADI Scheme. The alternating direction implicit method consists in the cyclic alternate application of the tridiagonal matrix algorithm (Patankar, 1980) in the three spatial directions. When applied in the ξ-direction, Eq. (3) should be written as

$$a_P\phi_P + a_e\phi_E + a_w\phi_W = d_P \tag{7}$$

where

$$d_P = -a_n\phi_N - a_s\phi_S - a_d\phi_D - a_f\phi_F + b \tag{8}$$

The right-hand side of Eq. (8) must be evaluated using the available values of ϕ. The solution of Eq. (7) involves initially the evaluation of arrays $\mathcal{P}$ and $\mathcal{Q}$ defined as

$$\mathcal{P}_P = \frac{-a_e}{a_P + a_w\mathcal{P}_W}\,, \qquad \mathcal{Q}_P = \frac{d_P - a_w\mathcal{Q}_W}{a_P + a_w\mathcal{P}_W} \tag{9}$$

Thus, ϕ_{NI} is equal to $\mathcal{Q}_{NI}$ and the other values of ϕ are calculated as

$$\phi_P = \mathcal{P}_P\phi_E + \mathcal{Q}_P \tag{10}$$

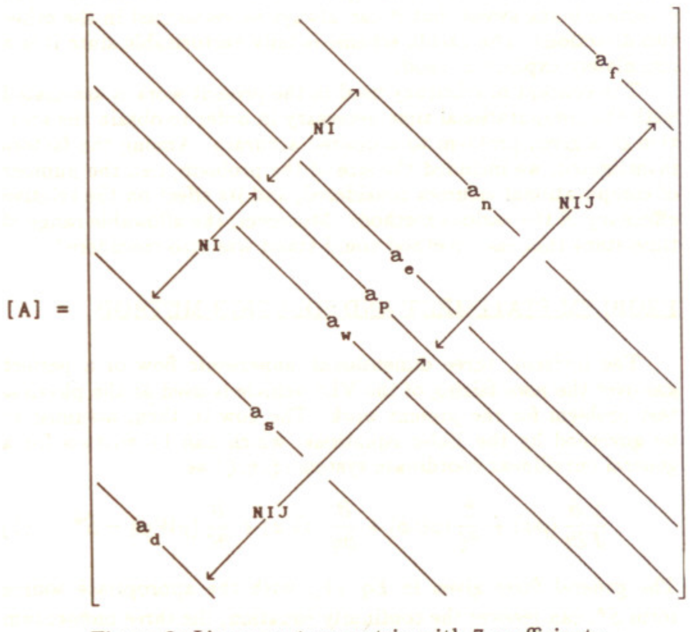

Figure 3: Linear system matrix with 7 coefficients.

We observe that those are very recursive relations, which implies that the direction being swept cannot vectorize. It is possible, however, to vectorize along the plane normal to the sweep direction, i.e., in the other two coordinate direction. In terms of memory usage, 4 additional arrays of dimension $NIJK$ are required, one for $\mathcal{P}$, another for $\mathcal{Q}$ and two others to aid in the vectorization process.

The MSI Procedure. The $[A]$ coefficient matrix, in the modified strongly implicit procedure (Zedan and Schneider, 1983), undergoes an approximate LU decomposition such that

$$[L][U] = [A] + [B] \tag{11}$$

Matrix $[B]$ has 12 non-zero diagonals adjacent to the diagonals of the matrix of coefficients $[A]$. Since the decomposition process is an approximate one, the solution of Eq. (3) involves an iterative process which recurrence relation is given by

$$[A+B]\{\phi\}^{n+1} = [A+B]\{\phi\}^{n} - \{[A]\{\phi\}^{n} - \{b\}\} \tag{12}$$

Here, n simply denotes the iteration level, and it is not the time step counter. When the process converges, $\{\phi\}^{n+1} = \{\phi\}^{n}$, and Eq. (12) then yields $\{[A]\{\phi\}^{n} - \{b\}\} = 0$. Therefore, if the iteration process described by Eq. (12) converges, it will converge for the solution of Eq. (3).

In order to actually apply Eq. (12), it must be rewritten in a more appropriate form. If we define a correction to the ϕ field as

$$\{\delta\}^{n+1} = \{\phi\}^{n+1} - \{\phi\}^{n} \tag{13}$$

and a residue, or error, as

$$\{R\}^{n} = [A]\{\phi\}^{n} - \{b\} \tag{14}$$

with the help of Eq. (11), we could finally obtain

$$[L][U]\{\delta\}^{n+1} = \{R\}^{n} \tag{15}$$

Since the $[L]$ and $[U]$ matrices are lower and upper triangular matrices, the solution of the above equation can be obtained by a forward sweep followed by a backward sweep. The method, as implemented here, requires 4 arrays with dimension $6 \times NIJK$, 4 arrays with dimension $7 \times NIJK$ and 5 arrays with dimension $NIJK$.

Storage Requirements and Convergence Criterion. A good estimate of the amount of core memory that a code requires to run can be obtained by considering the large arrays it needs. If we do not consider the arrays used by the linear system solution method, the code here implemented requires 68 arrays with dimension $NIJK$ and 4 arrays with dimension $7 \times NIJK$. This gives, therefore, a total of 96 arrays of dimension $NIJK$, where we recall that $NIJK$ is the total number of interior computational volumes in the problem. From the foregoing discussion, we can see that the Jacobi scheme requires one additional array with dimension $NIJK$. The ADI scheme requires 15 additional arrays with the same dimension, and the MSI procedure needs 57 of these additional arrays.

The Euclidean norm of the residues is defined as

$$NER = \sqrt{\sum\left(b - \sum a_{nb}\phi_{NB} - a_P\phi_P\right)^2} \tag{16}$$

For the cases considered here, all three methods were iterated until the Euclidean normal dropped to 0.005 of its initial value.

RESULTS

CPU usage curves, representing the time required to achieve steady state, were constructed for all cases analyzed here as a function of the dimensionless time step. This nondimensional time step, DT, is defined as

$$DT = \Delta t^* \, R/\mathcal{U}_\infty \tag{17}$$

Here, Δt^* is the dimensional time step, R the radius of the cylindrical forebody region, and $\mathcal{U}_\infty$ is the freestream velocity. Since we are only interested in steady state problems, the objective would be to make DT as large as possible, because this typically leads to faster convergence. Since how much one gains in performance by going from scalar to vector mode is typically dependent on the size of vectors used, it is important to consider the effect of the size of the problem in order to fully characterize the relative efficiency of the various methods. To that end, test with two different meshes were performed. The smaller one uses a total of 7200 volumes ($NI = 30$, $NJ = 24$ and $NK = 10$), and the other mesh requires 28800 volumes ($NI = 60$, $NJ = 24$ and $NK = 20$). A longitudinal plane of the larger mesh is shown in Fig. 4.

For all test cases considered in the present work, we assumed $M_\infty = 3.75$, $T_\infty = 168$ K and $\rho_\infty = 0.3175$ kg/m^3. For the cases with the 7200 volume mesh, we assume $\alpha = 0°$. The cases using the 28800 volume mesh, the angle of attack was set at 5°. Fig. 5 presents the performance curves otained for the tests with the mesh with 7200 volumes. The overall behavior of these curves is in good agreement with what one should expect. For very small DT's, a large number of iterations is required in order to reach steady state. Therefore, CPU time is large. The computational effort has a minimum and, afterwards, it raises again with increasing DT.

We can observe that all 6 curves have a very similar shape. This is an indication that the other phases of the problem solution process are not sensitive to the differences in the linear system solutions. We can also observe from Fig. 5 that, in scalar mode, MSI is by far the cheapest method, and Jacobi is the most expensive one. The reasons for this behavior are very well understood, and usually can be explained in terms of the fact that information in the Jacobi method propagates at a rate of one volume per iteration. Hence, even though one Jacobi iteration is much faster than one MSI iteration, the implicitness of the MSI procedure allows convergence in a much smaller number of iterations. The efficiency of the three methods becomes much more comparable in vector processing mode. Now, Jacobi is the most efficient method. Although the

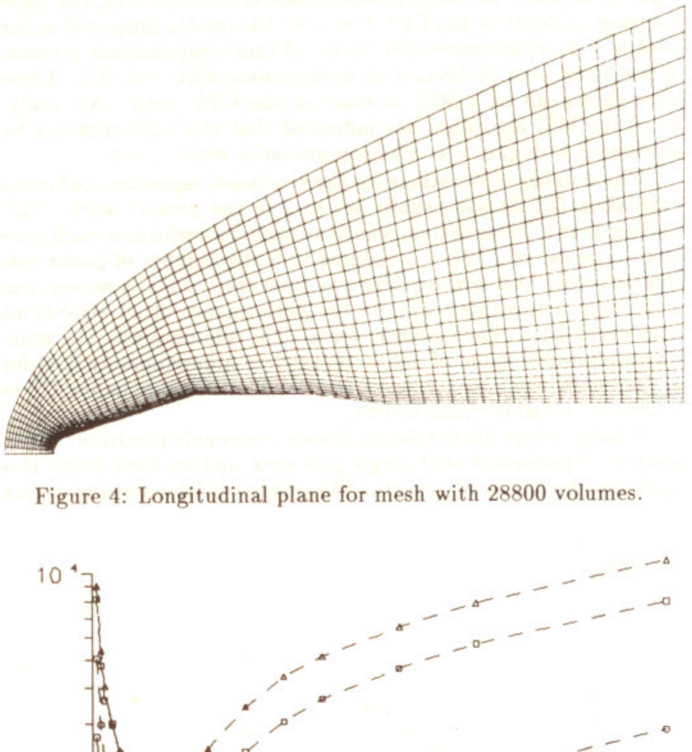
Figure 4: Longitudinal plane for mesh with 28800 volumes.

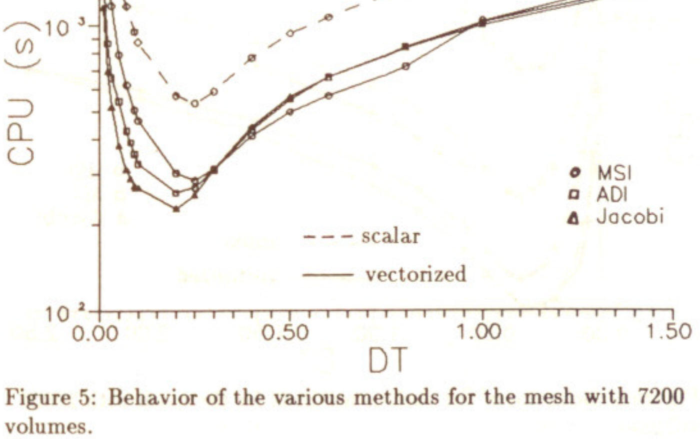

Figure 5: Behavior of the various methods for the mesh with 7200 volumes.

authors must admit that they did not expect such behavior, this result is not entirely surprising since the Jacobi procedure is fully vectorizable while the amount of vectorization possible with MSI is very limited.

Results for the mesh with 28800 volumes are shown in Fig. 6. Here, the behavior of the Jacobi scheme is rather odd. The Jacobi scheme presents performance similar to ADI up to $DT = 0.2$. For $0.2 \leq DT \leq 0.5$, the solution does not converge in less than 20000 s with the Jacobi scheme, even with vector processing. From $DT = 0.5$ up to $DT = 1.5$, the Jacobi method again produces converged solutions, but with a performance much poorer than ADI. For larger time steps, the solution diverges. Aside from this atypical behavior of the Jacobi scheme, the results in Fig. 6 do not present any significant differences with respect to those shown in Fig. 5. Of course, the CPU time grows approximately proportional to the number of volumes in the mesh.

Table 1 presents average values for the ratio between CPU times for scalar and vector processing, for all three methods and for the two meshes considered. As expected, the MSI procedure extracts the smallest performance improvement from vectorization. The Jacobi algorithm, on the other hand, obtains the largest performance boost from vectorization for the cases considered. It is interesting to observe that the relative benefits of vector processing increased with mesh size only for the ADI method. Since we are measuring the overall CPU time here, other phases of the solution process are involved besides linear system solution. Therefore, the full meaning of this result is not completely clear.

The MSI procedure involves a process which tries to partially compensate for the approximate decomposition errors. We have found that the α parameter, which controls this process, can have a strong influence in the CPU time. All the results presented so far used $\alpha = 0$, which essentially turns off this compensation process. A few tests were performed in vector mode with $\alpha = 0.5$. These have shown up to a 50% increase in the CPU time. An analysis of the MSI algorithm has indicated that this difference can be attributed to higher vectorization potential when $\alpha = 0$.

The number of iterations for the continuity equation coefficient evaluation (ITM) was usually set to 2 in the present work. This implies that 9 linear system solutions must be performed each time step. The use of $ITM = 1$ reduces to 5 the number of linear system solutions that are required per time step. This, however, has a strongly detrimental effect in the performance of all algorithms considered here. For instance, for a 7200 volume mesh, the maximum allowable value of DT is reduced to approximately 0.09 for all methods. Such an stringent limitation on the maximum DT is a very undesirable characteristic.

Finally, all results previously shown use double precision. A few tests were performed with single precision, and we have found this to be detrimental in all aspects. The number of time steps to obtain

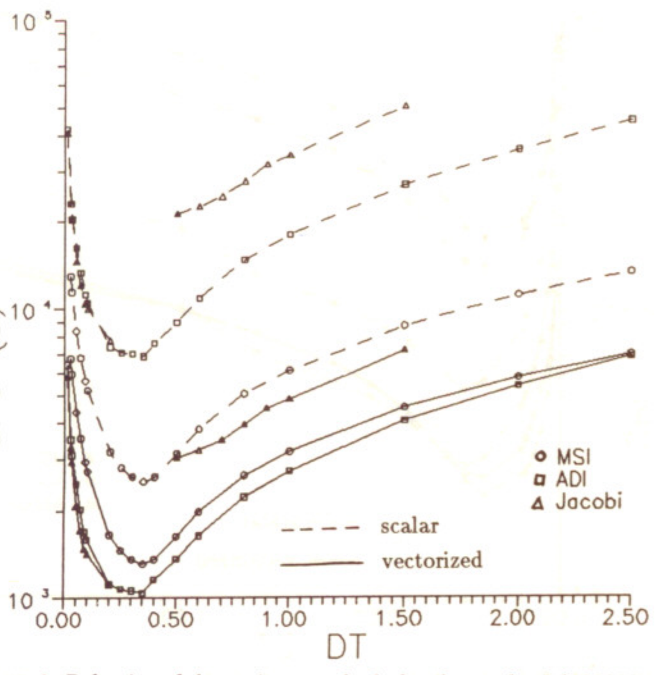

Figure 6: Behavior of the various methods for the mesh with 28800 volumes.

Table 1: Ratio of scalar CPU time over vector CPU time.

Method	Mesh Size	
	7200	28800
Jacobi	7.8	7.0
ADI	5.5	6.6
MSI	1.9	1.9

convergence to steady state is typically increased and, at most, remains unchanged. The number of iterations in the linear system solution increases for all methods. Maximum allowable values of DT for which there is convergence are reduced to very low values. Actually, a combination of $ITM = 1$ and single precision makes the use of the Jacobi scheme practically impossible.

CONCLUSIONS

The results demonstrate that strongly implicit procedures, such as MSI, which are very efficient in scalar processing mode, can have poorer performance than fully explicit schemes in vectorized mode. Moreover, explicit schemes such as the Jacobi algorithm imposed almost no additional penalty in terms of memory requirements for linear system solution. These observations can be enough justification to consider the replacement of strongly implicit procedures by, apparently, older explicit linear system solver algorithms when the possibility of vectorization is available. We further observed that the use of $\alpha = 0$ makes the MSI scheme at least partially vectorizable and, hence, more efficient. The use of single precision must be avoided and, for the kind of finite volume methodologies here implemented, the use of $ITM = 2$ is recommended. It is clear that there are quite a few aspects of computational performance improvement which were not addressed here. Nevertheless, the authors believe that the present results can be relevant for those interested in the numerical solution of complex fluid flow problems.

Moreover, the particular details of how one goes about in order to obtain vector code were not discussed in the present work, because these could be machine dependent. Some general rules are presented by Olson (1990). More specific details can generally be found at the vector optimization guide of the particular installation (in the present case, "Convex Fortran ...", 1989). The authors would like to emphasize, however, that the overall conclusions of the present work are thought to be general and independent of installation details.

REFERENCES

- Marchi, C.H., Maliska, C.R., and Bortoli, A.L., "The Use of Co-located Variables in the Solution of Supersonic Flows," *Proceedings of the 10th Brazilian Congress of Mechanical Engineering*, Vol. 1, Rio de Janeiro, Dec. 1989, pp. 157-160.
- Marchi, C.H., Maliska, C.R., and Silva, A.F.C., "A Boundary-Fitted Numerical Method for the Solution of Three Dimensional All Speed Flows Using Co-located Variables," *Proceedings of the 3rd Brazilian Thermal Sciences Meeting*, Vol. 1, Itapema, SC, Dec. 1990, pp. 351-356.
- Olson, R.W., "Introduction to Vector Processing," Notes for short course presented at the 3rd Brazilian Thermal Sciences Meeting, Itapema, SC, Dec. 1990.
- Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Co., Washington, 1980.
- van Doormall, J.P., "Numerical Methods for the Solution of Incompressible and Compressible Fluid Flows," Ph.D. Thesis, University of Waterloo, Ontario, Canada, 1985.
- van Doormall, J.P., and Raithby, G.D., "Enhancements of the Simple Method for Predicting Incompressible Fluid Flows," *Numerical Heat Transfer*, Vol. 7, 1984, pp. 147-163.
- Zedan, M., and Schneider, G.E., "3-D Modified Strongly Implicit Procedure for Finite Difference Heat Conduction Modelling," *AIAA Journal*, Vol. 21, No. 2, Feb. 1983, pp. 295-306.
- "Convex Fortran Guide to Vector and Parallel Optimization," Doc. No. 720-000930-200, 1st Edition, Convex Computer Corporation, Richardson, TX, May 1989.