ARTICLE NO. 79-1457R

J80-149

# Vector Processor Algorithms for Transonic Flow Calculations

00001
20005
20018

Jerry C. South Jr.* and James D. Keller†
*NASA Langley Research Center, Hampton, Va.*
and
Mohamed M. Hafez‡
*Flow Research Co., Kent, Wash.*

**This paper discusses a number of algorithms for solving the transonic full-potential equation in conservative form on a vector computer, such as the CDC STAR-100 or the CRAY-1. Recent research with the "artificial density" method for transonics has led to development of some new iteration schemes which take advantage of vector-computer architecture without suffering significant loss of convergence rate. Several of these more promising schemes are described and two- and three-dimensional results are shown comparing the computational rates on the STAR and CRAY vector computers and the CYBER-175 serial computer. Schemes included are: 1) checkerboard SOR, 2) checkerboard leapfrog, 3) odd-even vertical line SOR, and 4) odd-even horizontal line SOR.**

## Introduction

In the last decade, significant progress has been made in the analysis of three-dimensional (3-D) aerodynamic configurations at transonic speeds. The key element in this progress has been the development of reliable relaxation methods for solution of the nonlinear potential equation in various forms. In many cases, useful results have been obtained with these methods on standard "serial-type" computers such as the CDC 6000 series and the IBM 360 series machines. However, production-type runs of 3-D transonic codes use excessive amounts of computer resources. An example is the Caughey-Jameson[1] transonic wing code FLO 22 used extensively throughout the government and aircraft industry: A medium parabolic-coordinates grid of $192 \times 16 \times 32$ (chord × normal × span) takes about 1700 s on a CYBER-175 computer, with an accounting cost of about $500. It is estimated that the inclusion of a sophisticated 3-D boundary-layer interaction scheme might triple the cost.

Furthermore, the finest grid which can be operated in the CYBER-175 (with $131{,}000_{10}$ word central memory) with the FLO 22 code is $192 \times 24 \times 32$, which has been found to provide inadequate resolution of certain features of the flow such as swept, oblique, weak shock waves where the flow before and after the shock is supersonic. A more desirable grid for resolving such features seems to be about $300 \times 32 \times 32$, well beyond the capacity of most existing high-speed serial computers. There is thus an increasing interest in operating these codes on more powerful computers in terms of both memory and speed, such as the CRAY-1, CDC STAR-100, or ILLIAC-IV.

The current relaxation methods used in the 3-D transonic codes are not suitable for the vector-architecture computers just mentioned. For example, one of the most reliable and widely used methods is line successive over-relaxation (LSOR); its convergence rate depends partly upon using the latest available values from the previous line while solving for the current line; hence, the lines cannot be solved simultaneously. Use of old values of the preceding line would reduce the method to line Jacobi and (assuming stability) allow simultaneous solution of all lines; however, the convergence rate would be intolerably slow, negating the advantages of the fast computational rate of the vector or parallel computer. It is clearly desirable to search for methods that can take advantage of the long vector operational capabilities of such machines as STAR without suffering significant loss in convergence rate compared to established serial-computation methods.

Some progress has been made in this direction already. Smith et al.,[2] carried out a preliminary study of "vectorizing" the LSOR algorithm in FLO 22. The serial version of the code has the option to use either vertical or horizontal line relaxation; the vector version used the horizontal line algorithm to achieve the longer vector length (192 to 300). Speed gains of the STAR-100 over the CYBER-175 ranged from 2 to 3, indicating that far greater vector lengths and less serial, or scalar, operations were required for the STAR-100 to approach its potential efficiency. Keller and Jameson[3] studied a three-iteration level, explicit, point relaxation scheme for the 2-D small disturbance equation, with vector lengths equal to the number of points in a plane. Their results indicated that such a code, with vector operations of length $I \times J$ of order $10^3$ or more approached the full potential of the STAR-100 capability; however, the explicit scheme took about three times as many cycles on a given mesh as LSOR, so some of the benefits were lost. Their net gain over a good serial LSOR version on CYBER-175 was 1.8, although the vector-code computational rate was 250,000 points/s. Redhed et al.[4] discussed an alternating line SOR method for a 3-D transonic wing code, wherein the odd and even vertical lines in a cross-flow ($Y$-$Z$) plane were solved separately; the computation of the equations (residual and tridiagonal coefficients) was a vector length equal to half the cross-plane dimension $J \times K/2 = 28 \times 20/2 = 280$, while the even- and odd-line tridiagonal solutions were carried out with $J/2 = 14$ length vectors. This procedure represented a compromise in that it retained the essentials of the relatively successful LSOR algorithm, while achieving an increase in vector length for a significant part of the calculation. The net result for this algorithm on STAR-100 was a speed gain of 3.4 over the

Presented as Paper 79-1457 at the AIAA 4th Computational Fluid Dynamics Conference, Williamsburg, Va., July 23-24, 1979; submitted Aug. 10, 1979; revision received Dec. 13, 1979.  Reprints of this article may be ordered from AIAA Special Publications, 1290 Avenue of the Americas, New York, N.Y. 10104. Order by Article No. at top of page. Member price $2.00 each, nonmember, $3.00 each. **Remittance must accompany order.**

Index categories: Transonic Flow; Computational Methods; Aerodynamics.

*Head, Theoretical Aerodynamics Branch, Subsonic-Transonic Aerodynamics Division. Associate Fellow AIAA.

†Aero-Space Technologist, Theoretical Aerodynamics Branch, STAD. Member AIAA.

‡Senior Scientist. Member AIAA.

CYBER-175 LSOR code. Obviously, a larger speed ratio would have been obtained using a finer mesh; the $64\times28\times20$ mesh employed in Ref. 4 should be considered as a coarse 3-D mesh. Reference 4 rightly concluded that more study is needed to eliminate the short vector operations. Hotovy and Dickson[5] presented a "three-color checkerboard" algorithm for use on STAR-100 for a 2-D transonic small disturbance equation. The third "color" in the checkerboard pattern of solution was required in their algorithm because of the update of the $(i\text{-}2, j)$ point in the upwind difference for $\phi_{xx}$ at supersonic points. Their algorithm required that point to be at the same iteration level as the $(i,j)$ point; and, hence, they could not both belong to the same vector or "color." The vector lengths in the algorithm were one-third of the mesh points in the $I\times J$ plane. The algorithm suffered from the same defect as did the explicit scheme of Ref. 3, in that it required about three times as many cycles as a good LSOR version; the net speed ratio was about 2 when the three-color checkerboard on STAR was matched against LSOR on CYBER-175.

In this paper, we present some new algorithms which solve the 2-D and 3-D full-potential equation in conservation form. All of the schemes are based upon the artificial density formulation of the potential equation studied in Refs. 6-8. With this formulation, transonic full-potential calculations are rather easily vectorized. In the following section, the artificial density method is discussed briefly. Improved procedures are given which avoid erratic behavior at shocks. Subsequent sections discuss some properties of vector-processor machines, iteration algorithms, and numerical results.

## Artificial Density Method

The artificial density method has been proposed in Refs. 6-8 as a simple method for introducing dissipation into the difference analog of the conservative full-potential equation. The idea is to solve the equation

$$(\tilde{\rho}\phi_x)_x+(\tilde{\rho}\phi_y)_y+(\tilde{\rho}\phi_z)_z=0$$

where

$$\tilde{\rho}=\rho-\mu\Delta s\rho_s,\quad \rho=(M_\infty^2a^2)^{1/\gamma-1},\quad a^2=\frac{1}{M_\infty^2}+\frac{\gamma-1}{2}(1-q^2)$$

$$q^2=\phi_x^2+\phi_y^2+\phi_z^2,\quad \mu=\max(0,1-a^2/q^2)$$

The term $\Delta s\rho_s$ in the $\tilde{\rho}$ equation is the product of the step length along a streamline and the streamwise density gradient. The use of $\tilde{\rho}$ in the potential equation instead of the isentropic density $\rho$ produces a dissipative difference scheme when correct differencing is used. In Refs. 7 and 8, the isentropic density was calculated at the node points using the velocity components centered there. The difference operator requires the density values at midpoints of line segments connecting node points; hence, these midsegment values were obtained by averages of the adjacent node point values. In Ref. 7, this convenience was carried through to the calculation of the artificial density $\tilde{\rho}$,

$$\tilde{\rho}_{ij}=\rho_{ij}-\mu_{ij}(\rho_s\Delta s)_{ij}$$

where upwind formulas were used for the approximation to $\rho_s\Delta s$. The midsegment values of $\tilde{\rho}$ were calculated as averages. A similar procedure was used in Ref. 8. In both Refs. 7 and 8, it was noted that overshoots (nonphysical expansions) occurred just ahead of a shock wave and that it was usually necessary to multiply the switch function $\mu$ by a constant or variable factor to increase the amount of dissipation. This procedure tends to cause more smearing of the shocks, and a reliable prescription for selecting the factor from one case to the next is not known. South and Jameson[9] relate this erratic behavior to two sources: 1) the mislocation of the switch function $\mu$, and 2) calculation of the midsegment densities by averaging nodal point values. In Ref. 9, the difficulty is cured if the switch function $\mu_i$, defined at the node point $i$ rather than at the midsegment $i+\frac{1}{2}$, is used in the expression for $\tilde{\rho}_{i+\frac{1}{2}}$ and if the midsegment values of the density are calculated using the midsegment values of the velocities as in Refs. 6, 10, or 11. Two-dimensional results illustrating these points are given in Figs. 1-3. Figures 1 and 2 compare the nodal point and midcell evaluation of artificial density for a nonlifting NACA 0012 airfoil at $M=0.85$ and 0.92, respectively. These results, and all others in this paper, use the planar, small-disturbance boundary condition at the airfoil mean chord line. The $M_\infty=0.92$ case exhibits a "fishtail" shock pattern, with an oblique shock at the trailing edge

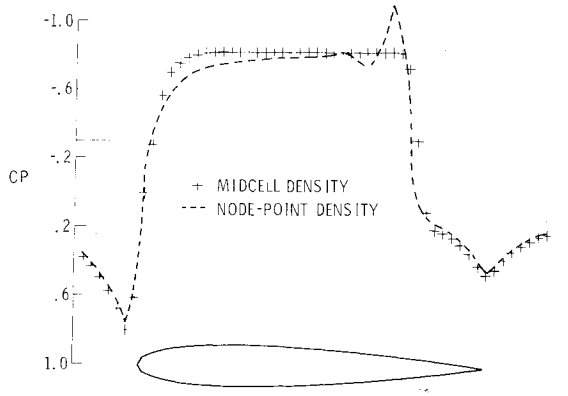


**Fig. 1 Pressure distribution for NACA airfoil, $M_\infty=0.85$, $\alpha=0$, $121\times25$ mesh.**

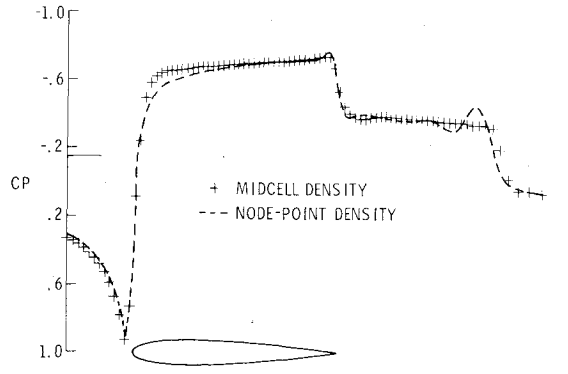


**Fig. 2 Pressure distribution for NACA airfoil, $M_\infty=0.92$, $\alpha=0$, $121\times49$ mesh.**

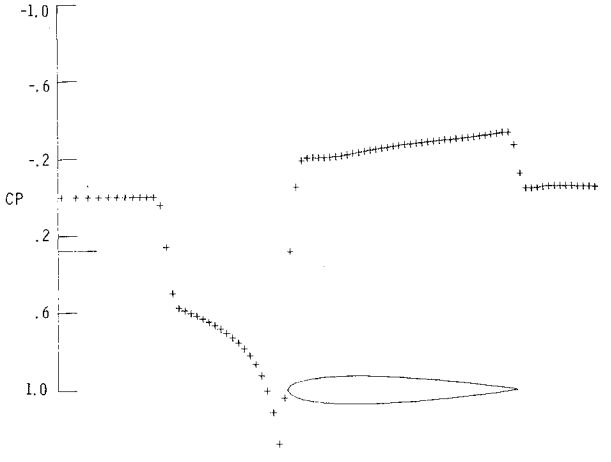


**Fig. 3 Pressure distribution for NACA 0012 airfoil, midcell density, $M_\infty=1.2$, $\alpha=0$, $121\times61$ mesh.**

followed by a normal shock in the wake. The nodal point density evaluation gives nonphysical overshoots (which could be diminished at the expense of more smearing), while the midcell density yields sharp shocks with no discernible overshoots. Figure 3 shows the midcell method alone for $M_\infty = 1.2$, with a bow shock ahead and an oblique shock at the tail. Again the results show a sharp shock without overshoots.

It has been found that the question of iterative schemes for solving the artificial density difference equations is an issue almost separate from the artificial density formulation. Indeed, fairly simple iteration schemes could be applied to the difference equations regardless of how the artificial viscosity is introduced. Based upon these simplified iteration schemes, new methods for transonics are especially effective on a vector or parallel computer as discussed in a later section. First, calculations on vector computers are discussed in the following section.

## Vector Processor Properties

The advent of the new vector processor type of large-scale computers (CDC STAR-100, CRAY-1, ILLIAC IV, and Texas Instrument ASC) will have a significant impact on iterative algorithms used for transonic flow calculations. There are currently two types of vector processors. The first type gets its speed by having a number of parallel processors which can perform the same operation on different sets of operands simultaneously. An example of this type is the ILLIAC IV. The second type is the pipeline processor which does operations on streams of operands in assembly line fashion. CDC STAR-100 and CRAY-1 are examples of this second type. Vector computers are most efficient when the algorithms used call for the same operations to be done on a number of different pairs of operands (or vectors) independently. The parallel processors operate at their fastest rate when the vector length is equal to the number of processors or a multiple thereof. The CRAY-1 pipeline-type processor also operates most efficiently on vectors which are of length 64 or a multiple of 64. This is because the vector registers hold 64 words which are sent to the pipeline. The STAR-100 pipeline-type processor is unique in that it continues to operate more efficiently as the vector length increases. It is much more efficient on long vectors (say 1000 or more) than on short vectors (200 or less). It is this feature of the STAR-100 which has motivated the search for algorithms which can be done with long vector instructions. Another feature of such machines is that as the ratio of vector speed to scalar speed increases, it becomes even more important to avoid scalar operations. For example, if vector operations can be done 20 times faster than scalar operations and a code does 90% of its operations in vector mode and 10% in scalar mode, then the 10% of the operations which are scalar will take 69% of the time.

As discussed in the Introduction, the methods in wide use for transonics today are not very suitable for vector computers; they involve short vector operations and over 10% scalar work. The following section discusses transonic algorithms which use moderate to long vectors and which appear to minimize the scalar work.

## Iteration Algorithms

So far the search for efficient iterative schemes for vector processors has developed into a study of compromises, as we will discuss in the following paragraphs. Several explicit, implicit, and semi-implicit schemes are described, together with their advantages and disadvantages for existing vector computers. We will concentrate on the one we have found to be the most efficient so far, which we call ZEBRA II.

### Checkerboard SOR (CKBSOR)

One way to construct an algorithm in terms of vectors whose elements are independent is to use the familiar black-white point ordering; namely, if $i+j+k$ is odd, the point is called black and if $i+j+k$ is even, it is white. Variants of this approach have been studied in Refs. 4 and 5, as already discussed. With the artificial density formulation, we need only two colors, however, compared to the multicolor scheme of Ref. 5. A complete iteration cycle consists of a sweep of the black points, followed by a sweep of the white points, using the latest available values of the potential $\phi$ in the computation of the residuals, but not in the artificial density. At the end of both sweeps, the density is updated. Experiments have been made with updating the density less frequently than every cycle, with some gain in efficiency, as will be discussed later. The checkerboard SOR scheme is "volume vectorizable," i.e., the vector operations are of length $\sim 0(I \times J \times K)$. On the other hand, experiments on a serial machine show it to be somewhat slower than LSOR.

### Checkerboard Leapfrog (CKBLF)

A close cousin of the checkerboard SOR is the checkerboard leapfrog scheme. It is a special case of the three-level Richardson method, where the modified equation is a damped wave equation. Variants of this were tried in Refs. 3 and 7 without the checkerboard ordering. The standard leapfrog (DuFort-Frankel[12]) scheme as applied to the 2-D potential equation, but with black-white ordering is:

$$(\phi_{ij}^{n+1} - \phi_{ij}^{n-1}) = \bar{\rho}_{i+\frac{1}{2},j}(\phi_{i+1,j}^{\nu} - \hat{\phi}_{ij}) - \bar{\rho}_{i-\frac{1}{2},j}(\hat{\phi}_{ij} - \phi_{i-1,j}^{\nu})$$

$$+ \left(\frac{\Delta x}{\Delta y}\right)^2 [\bar{\rho}_{i,j+\frac{1}{2}}(\phi_{i,j+1}^{\nu} - \hat{\phi}_{ij}) - \bar{\rho}_{i,j-\frac{1}{2}}(\hat{\phi}_{ij} - \phi_{i,j-1}^{\nu})]$$

where

$$\hat{\phi}_{ij} = \tfrac{1}{2}(\phi_{ij}^{n+1} + \phi_{ij}^{n-1})$$

and where

$$\nu = n \text{ for } i+j \text{ odd}, \quad \nu = n+1 \text{ for } i+j \text{ even}$$

The extension to the third dimension is obvious. As in Ref. 7, to improve convergence for supersonic points, a term $\beta\phi_{xt}$ is added in the form:

$$\phi_{xt} \propto (\phi_{ij}^{n+1} - \phi_{ij}^{n-1}) - (\phi_{i-1,j}^{n+1} - \phi_{i-1,j}^{n-1})$$

The advantage of CKBLF is that it requires storage for only two levels of $\phi$ at each point, rather than three levels as in Refs. 3 and 7. Numerical experiments show CKBLF to have nearly the same convergence rate as CKBSOR, as one might expect; it is likewise volume vectorizable.

### ZEBRA I

An interesting two-color *line* scheme is one we call ZEBRA I; so-called ZEBRA because alternate lines, rather than points, are colored black and white. The idea is to perform LSOR at alternating vertical lines, where the $(X\text{-}Y)$ plane (the plane containing the wing planform) has a checkerboard pattern and tridiagonal systems of equations are solved simultaneously along the vertical $(Z)$ lines. That is, if vertical lines with $i+j$ odd are black and $i+j$ even are white, we solve all the black tridiagonals simultaneously, followed by all the white tridiagonals, using in the residual calculation the latest values of $\phi$ from the opposite vector (black or white) family, as appropriate. The necessary amount of $\phi_{xt}$ can be added to the operator by including a term involving $\Delta\phi_{i-1,jk}$, since it is always of the opposite family. We should point out that ZEBRA I is not the same as the alternating line SOR scheme of Ref. 4, which solved alternating vertical lines in each $i$ = constant cross-plane with vector instructions of length $J/2$ for each color. ZEBRA I has vector length $I \times J/2$ for the $\phi$ update, while the density and residual calculations can be

volume vectorized. This scheme was tested in both a scalar and vector 2-D code, and it has a convergence rate identical to our artificial density implementation of LSOR. Another version of ZEBRA I was tested in a 3-D code. This version performs on alternating horizontal lines, where the $(Y\text{-}Z)$ plane has a checkerboard pattern and tridiagonal systems are solved simultaneously along horizontal $(X)$ lines.

#### ZEBRA II

The scheme which has proved to be most efficient on STAR-100 is a point scheme which mimics a full-plane SOR. It is a cross-plane checkerboard, where the horizontal lines $j+k$ odd are black and the lines $j+k$ even are white. Each plane $i=$constant is thus a checkerboard pattern. The algorithm is, in its simplest form for an unstretched grid, with $\Delta X=\Delta Y=\Delta Z$,

$$B\Delta\phi_{ijk}=\frac{\Delta X^2 R_{ijk}}{\bar{\rho}_{\text{avg}}}+\beta\Delta\phi_{i-1,jk}$$

where

$$\Delta\phi=\phi^{n+1}-\phi^{n}$$

$$\Delta X^2 R_{ijk}=\bar{\rho}_{i+\frac{1}{2},jk}(\phi^{n}_{i+1,jk}-\phi^{n}_{ijk})-\bar{\rho}_{i-\frac{1}{2},jk}(\phi^{n}_{ijk}-\phi^{n+1}_{i-1,jk})$$

$$+\bar{\rho}_{i,j+\frac{1}{2},k}(\phi^{\nu}_{i,j+1,k}-\phi^{n}_{ijk})-\bar{\rho}_{i,j-\frac{1}{2},k}(\phi^{n}_{ijk}-\phi^{\nu}_{i,j-1,k})$$

$$+\bar{\rho}_{ij,k+\frac{1}{2}}(\phi^{\nu}_{ij,k+1}-\phi^{n}_{ijk})-\bar{\rho}_{ij,k-\frac{1}{2}}((\phi^{n}_{ijk}-\phi^{\nu}_{ij,k-1})$$

$$\nu=n \text{ for } j+k \text{ odd}; \nu=n+1 \text{ for } j+k \text{ even}$$

$$B=\frac{2}{\omega_x}+\beta+\frac{2}{\omega_y}+\frac{2}{\omega_z}$$

The term $\beta$ is the coefficient of the added $\phi_{xt}$; the algorithm possesses "natural" $\phi_{xt}$ is that the $\phi$ values at the $i+1$ plane are old, while the values at $i-1$ are new. We use a simple scheme for automatically increasing or decreasing $\beta$ as needed. Generally, an estimate for the spectral radius based upon both the maximum residual and the average residual is calculated. Let

$$\text{SRM}=|R|^{n}_{\max}/|R|^{n-1}_{\max}$$

$$\text{SRA}=|R|^{n}_{\text{avg}}/|R|^{n-1}_{\text{avg}}$$

Then $\beta$ is changed according to:

$$\left.\begin{array}{l}\text{if (SRM + SRA)} > \text{SRMAX} \\ \text{and } \beta^n < \text{BMAX}\end{array}\right\} \rightarrow \beta^{n+1}=1.2\beta^{n}$$

$$\text{if (SRM + SRA)} < 2.0 \qquad \rightarrow \beta^{n+1}=0.98\beta^{n}$$

Typically, the parameters which have been quite successful for stretched Cartesian meshes are:

$$1.9\le\omega_x<2.0,\ \omega_y=\omega_z=2.0$$

$$\text{SRMAX}=2.1,\ \text{BMAX}=2.0,\ \beta^{0}=0.25$$

The usual behavior of the convergence process is a transient adjustment of the supersonic region and shock movement, followed by a fairly smooth decrease of the error with an attendant reduction in $\beta$. The three cases of Figs. 1-3 were solved easily by the 2-D version of ZEBRA II; the latter two should be considered as rather severe tests of an iterative scheme. So far, the ZEBRA II algorithm has been the most reliable method tested of those considered herein.

ZEBRA II has two features which are advantageous for the STAR-100 computer. First, the algorithm uses vector instructions most of which are of length 0 $(J\times K)$ or longer; typical vector lengths range 400-2000. Second, the storage occurs in a natural order so that a virtual memory system will have no excessive paging. The $\phi$ array is the only 3-D array stored as consecutive cross-planes; all other arrays are 2-D. For example, STAR-100 is a virtual memory machine, and the code and all 2-D arrays were stored on one large page. The other large pages available in core were left for the $\phi$ array. On the STAR-100 computer at Langley Research Center, which has about 1/2 million word (64 bit) storage, it would be possible to run cases with as many as 10,000 points in a cross-plane and still have a small number of page faults per iteration (depending upon how many points are in the $x$ direction). When all of the values of the potential function will not fit in core, it is possible to have the number of page faults per iteration equal to 2 plus the number of large pages (65,000 words) which will not fit in core.

#### Approximate Factorization

In Refs. 6-8, the artificial density difference equations were solved by implicit approximate factored methods. These methods solve systems of equations which are relatively easy to invert in alternating directions. We have temporarily sidetracked the study of vectorizing these methods for several reasons:

1) For each new directional sweep, the data must be rearranged (transposed) so it is in contiguous form for vector processing, and this is basically a scalar operation on STAR-100. Future vector processors may have a transpose network included in the hardware to accommodate the implicit factored schemes.

2) We found that on stretched Cartesian meshes at high Mach numbers, the methods given in Refs. 7 and 8 were not as reliable as hoped, and optimal sets of acceleration parameters were difficult to determine in general.

## Numerical Results

In this section, 2-D and 3-D results are presented. The 3-D results pertain to the node-point evaluation of the artificial density, as discussed in an earlier section. Time did not permit recoding for the midcell density evaluation, which would affect mainly the quality of results, with at most secondary effects on the convergence rates and vector-processor speeds.

#### 2-D Results

Comparison of speed of computations for a 2-D case using the three-level explicit algorithm of Ref. 7 and alternating line SOR (ZEBRA I) is given in Table 1. The three-level explicit

**Table 1 Timing comparison for two artificial density iterative algorithms for 2-D transonic flow**

| | Three-level algorithm, | Alternating line SOR algorithm | |
|---|---|---|---|
| | $61\times 26$ grid[a] | $61\times 26$ grid[b] | $121\times 26$ grid[c] |
| Machine | Rate, points/s | Rate, points/s | Rate, points/s |
| CYBER 175 | 20,000 | 18,000 | 19,000 |
| STAR-100 (64 bits) | 192,000 | 83,000 | 139,000 |
| STAR-100 (32 bits) | 450,000 | 136,000 | 228,000 |

[a] All vector lengths = 1500. [b] Shortest vector length = 30, longest vector length = 1500. [c] Shortest vector length = 60, longest vector length = 3000.

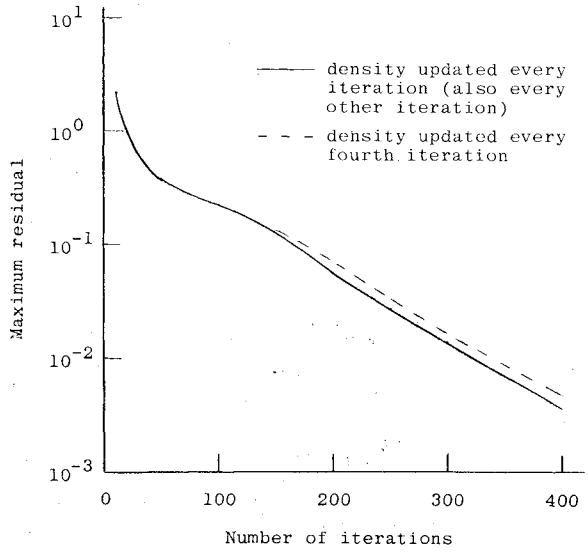


Fig. 4 Iteration history of LSOR ($\omega = 1.9$).

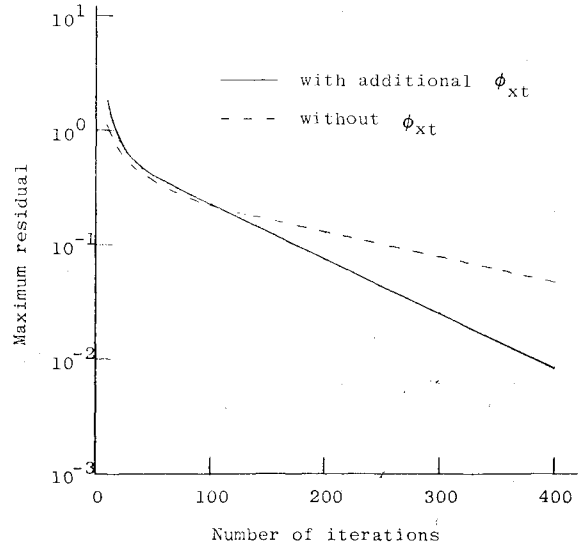


Fig. 6 Iterative history of SOR checkerboard ($\omega = 1.6$).

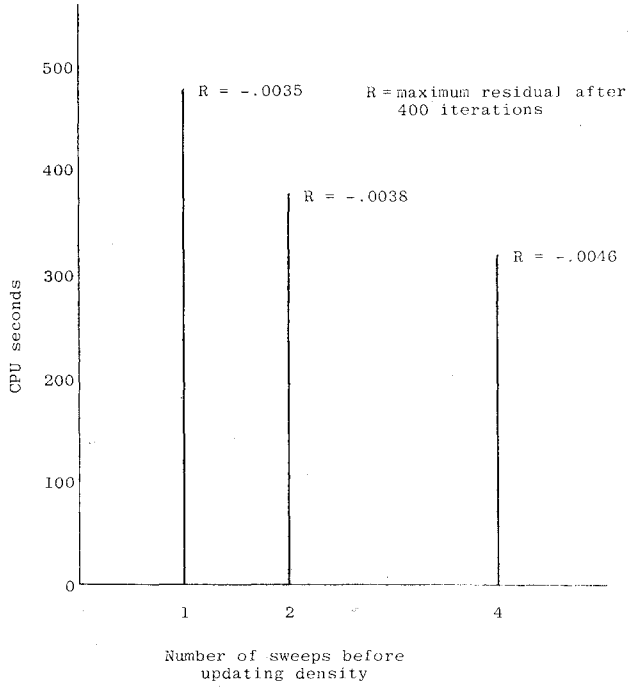


Fig. 5 CPU times for different density update frequency.

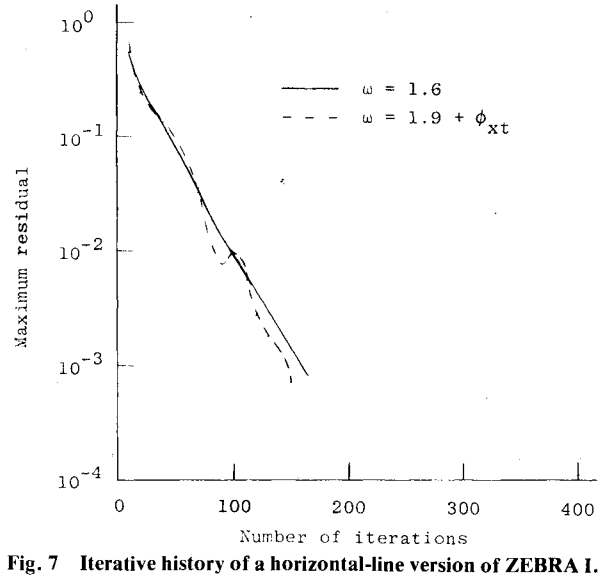


Fig. 7 Iterative history of a horizontal-line version of ZEBRA I.

scheme is completely vectorizable with long vectors while the ZEBRA I scheme uses short vectors of length $\sim 0\,(I/2)$ in 2-D. The three-level scheme has, however, a slower rate of convergence. We did not optimize parameters in either method, but we can safely assume that the three-level scheme takes about three times as many cycles as the ZEBRA I for equivalent convergence. Thus, for the $61 \times 26$ mesh, the two methods would require about the same CPU time to convergence, while for the $121 \times 26$ mesh, the minimum ZEBRA I vector length is doubled to 60 with a large increase in computational speed; it should be about 50% faster to converge than the three-level scheme. The 3-D ZEBRA I, it should be recalled, has minimum vector lengths $\sim 0(I \times J/2)$, and it will far exceed the efficiency of the three-level scheme on any reasonable 3-D mesh.

### 3-D Results

Three-dimensional calculations of transonic flows around a wing at an angle of attack are described next. The calculations are for $\alpha = 2$ deg, $M_\infty = 0.85$, NACA 0012 airfoil section, leading-edge sweep angle = trailing-edge sweep angle = 15 deg, on a grid $I \times J \times K = 70 \times 16 \times 30$. All runs were 400 cycles.

Figure 4 shows the rate of convergence for LSOR, $\omega = 1.9$, where the density is updated every sweep, every other sweep, and every four sweeps. Figure 5 shows the corresponding CPU times.

Figure 6 shows the rate of convergence for SOR checkerboard, $\omega = 1.6$, with and without a $\phi_{xt}$ term. It can be seen that the added $\phi_{xt}$ improves convergence.

Figure 7 shows the rate of convergence for a ZEBRA I scheme. The version of ZEBRA I tested here is based upon horizontal line relaxation with a checkerboard cross-plane. A $\phi_{xt}$ term is needed in order to use a large relaxation factor ($\omega = 1.9$). Notice that this version of ZEBRA I is faster than LSOR. This is because it is semi-implicit in the horizontal direction which has more points along each line. The shock position is also settled in fewer iterations.

Figure 8 shows the rate of convergence for ZEBRA II, $\omega = 1.9$, updating the density every sweep and every four sweeps.

The ZEBRA II code was operated without any vectorization on different machines and the CPU times are given in Table 2. Notice that STAR-100 is slow (about the speed of a

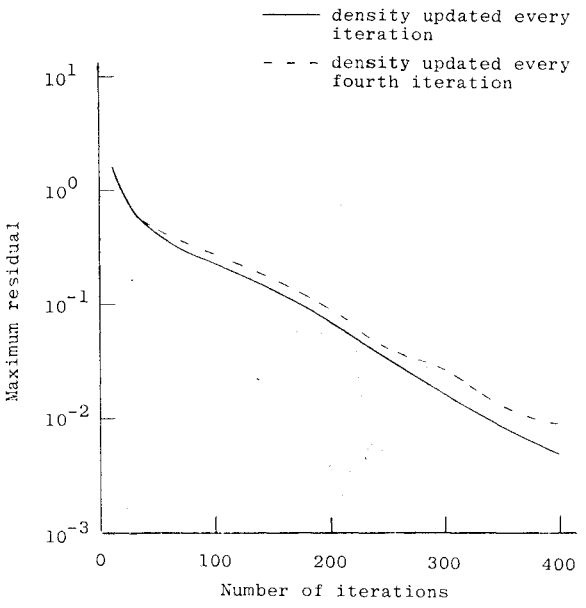


**Fig. 8 Iterative history of ZEBRA II ($\omega = 1.9$).**

CDC 6600) if scalar operations are used. The vectorized code runs, however, in 80 s.

The results of another case, where density is updated every four sweeps are given in Table 3.

The ZEBRA II code was originally written for use on a conventional-type computer and then modified for the CRAY-1 and the STAR-100. The modifications for the CRAY-1 were minor. The only changes necessary were to rearrange loops by introducing temporary vectors so that the compiler could generate vector instructions. For the CRAY-1 results in this paper, only a portion of the code was vectorized. The calculations of the density and the maximum residual were not vectorized. It is estimated that a completely vectorized code on the CRAY-1 would run in about 45 s. The modifications on the STAR-100 were not as easy to make. The code was rewritten using explicit vector instructions.

The two types of vector processor computers used in this study have different architectures. The CRAY-1 reaches its full speed with short vector lengths while the speed of the STAR-100 increases as the vector length increases. The results shown in this paper were computed on a grid with $(30 \times 16)$ 480 points in each cross-plane. On this size problem, the STAR-100 computational rate is 144,000 points/s. When the number of points in the cross-plane is increased to $(60 \times 32)$ 1920, the computational rate increases to 208,000 points/s. Thus, the STAR-100 has an additional advantage as the problem size increases. Finally, it should be emphasized that the 32-bit arithmetic capability of STAR-100 can be utilized to push the larger problem computational rate to 460,000 points/s and double the available core size.

**Table 2 Timing comparison for scalar codes of ZEBRA II algorithm for 3-D transonic flow**

| Machine | CPU time, s |
|---|---|
| CDC STAR-100 | 2272 |
| CYBER-175 | 724 |
| CDC 7600 | 443 |
| CRAY-1 | 222 |

**Table 3 Timing comparison for ZEBRA II algorithm for various code versions and computers with density updated every fourth cycle**

| Machine | CPU time, s |
|---|---|
| CYBER-175 | 456 |
| CDC 7600 | 282 |
| CRAY-1 without vectorization | 144 |
| CRAY-1 with partial vectorization | 69 |
| STAR-100 vectorized code | 58[a] |

[a] STAR-100 has 32-bit arithmetic capability which can be used via the SL/1 language; such codes run about 2.2 times faster than with 64-bit arithmetic. In this case the CPU estimate is 26 s.

## Conclusion

The artificial density method for computational transonics has opened the door to the development of new algorithms which are more efficient on vector computers such as STAR-100 and CRAY-1 than current "serial" methods such as LSOR. The architecture of the new computers suggests compromises which must be made in order to best utilize the capabilities of these machines. A good vector algorithm for the STAR-100 is one in which the data are processed in a natural contiguous pattern, the vectors are of moderately long length (400-2000, say), and the convergence rate is not much slower than good standard methods. One method which fits these requirements is ZEBRA II, which is essentially a cross-plane, point relaxation, checkerboard method.

The numerical results illustrate some of the compromises that must be made. The ZEBRA II scheme works efficiently on both CRAY-1 and STAR-100, with run times of about 1 min for a 3-D lifting wing with planar boundary conditions, Cartesian coordinates, and the full-potential equation. STAR-100 32-bit arithmetic capability would yield run times of 26 s. This run time represents a speed gain of 10 over CDC 7600 and 17 over the CYBER-175. Doubling the number of mesh points in each cross-plane direction increases the relative speed gain of STAR-100 by 30%.

It remains to be studied as to how well the ZEBRA II scheme, or some variant of it, can be made to work in other coordinate systems, such as the parabolic coordinates used in the Jameson-Caughey FLO 22 wing code. Further, it would be valuable to obtain some hard data on vectorization of some of the new implicit approximte factorization methods.

## Acknowledgment

The third author wishes to express his appreciation to Donald Lovell of Flow Research Co. and Frank Chism of United Computing Systems, Inc., for their help during the development of the CDC 7600 and CRAY-1 computer codes.

## References

[1] Jameson, A. and Caughey, D. A., "Numerical Calculation of the Transonic Flow Past a Swept Wing," ERDA Res. and Dev. Rept. C00-3077-140, New York University, June 1977.

[2] Smith, R. E., Pitts, J. I., and Lambiotte, J. J., "A Vectorization of the Jameson-Caughey NYU Transonic Swept-Wing Computer Program FLO-22-VI for the STAR-100 Computer," NASA TM-78665, March 1978.

[3] Keller, J. D. and Jameson, A., "Preliminary Study of the Use of the STAR-100 Computer for Transonic Flow Calculations," AIAA Paper 78-12, Huntsville, Ala., Jan. 1978.

[4] Redhed, D. D., Chen, A. W., and Hotovy, S. G., "New Approach to the 3-D Transonic Flow Analysis Using the STAR-100 Computer," *AIAA Journal*, Vol. 17, Jan. 1979, pp. 98-99.

[5] Hotovy, S. G. and Dickson, L. J., "Evaluation of a Vectorizable 2-D Transonic Finite Difference Algorithm," AIAA Paper 79-0276, New Orleans, La., Jan. 1979.

[6] Holst, T. L. and Ballhaus, W. F., "Conservative Implicit Schemes for the Full Potential Equation Applied to Transonic Flows," NASA TM-78469, March 1978.

[7] Hafez, M. M., South, J. C., and Murman, E. M., "Artificial Compressibility Methods for Numerical Solution of Transonic Full

Potential Equation," AIAA Paper 78-1148, Seattle, Wash., July 1978.

[8]Holst, T. L., "An Implicit Algorithm for the Conservative Transonic Full Potential Equations Using an Arbitrary Mesh," AIAA Paper 78-1113, Seattle, Wash., July 1978.

[9]South, J. C., and Jameson, A., "Recent Advances in Computational Transonics," Conference on Computers in Aerodynamics, Polytechnic Institute of New York, Farmingdale, N.Y., June 4-5, 1979, (to appear in *Computers and Fluids*).

[10]Jameson, A., "Transonic Potential Flow Calculations Using Conservation Form," *Proceedings of the 2nd Computational Fluid Dynamics Conference*, Hartford, Conn., June 1975, pp. 148-161.

[11]Jameson, A., "Numerical Computation of Transonic Flows with Shock Waves," Symposium Transonicum II, *Proceedings International Union of Theoretical and Applied Mechanics*, Göttingen, Sept. 8-13, 1975, K. Oswatitsch and D. Rues, Eds., Springer-Verlag, 1976.

[12]DuFort, E. C. and Frankel, S. P., "Stability Conditions in the Numerical Treatment of Parabolic Differential Equations," *Mathematical Tables and Other Aids to Computation*, Vol. 7, 1953, p. 135.