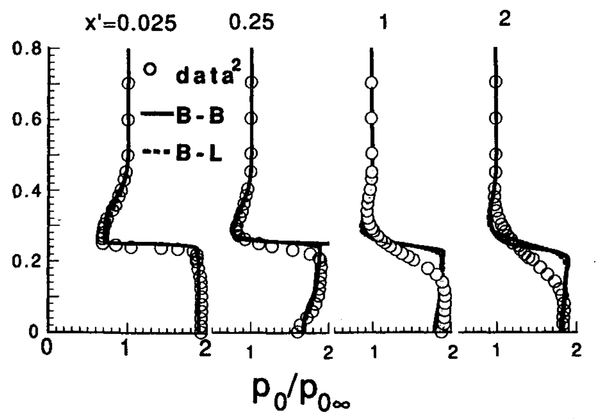

Fig. 3 Comparison of jet radial pitot pressure.

predicted locations of separation are $x' = -0.298$ and -0.251 for the Baldwin-Barth and Baldwin-Lomax models, respectively. In the attached region, both Baldwin-Barth and Baldwin-Lomax models agree well with the data. In the separated region, the Baldwin-Barth model compares considerably better with the data, as compared to the Baldwin-Lomax model. However, there still appears to be some overprediction of c_p by the Baldwin-Barth model and, hence, the underprediction of the drag.

Mixing Layer Predictions

The mixing layer predictions are compared in Fig. 3a–3d with the pitot pressure data at various streamwise stations in the jet ($x' = 0.025$, 0.25, 1.0, and 2.0). In general the predictions from the Baldwin-Barth model are close to those from the Baldwin-Lomax model in describing mixing layers. Both turbulence models clearly underestimate the mixing and, hence, entrainment of fluid from the external flow region. The deviations of the models with the data is seen to increase with increasing x'.

The Baldwin-Barth model, being built upon the k-ϵ model, should be expected to predict too small a wake growth, since the k-ϵ model is known to underpredict the far-wake growth. The assumption of equality of turbulence production and dissipation rates (equilibrium turbulence) over the entire shear layer/mixing layer is also not realistic. The flow under consideration undergoes abrupt changes in the turbulence structure, i.e., from a boundary-layer flow to a free shear flow with quite different length scales.

Conclusion

The present study of an afterbody/exhaust jet flowfield has shown that the Baldwin-Barth one-equation turbulence model offers a definitive improvement in the prediction of surface pressure in the pressure-gradient induced separation region over the Baldwin-Lomax algebraic model. However, there still is some overprediction of the data by the one-equation model. In the mixing layer, the Baldwin-Barth model considerably underpredicts the growth of the mixing layer, as does the Baldwin-Lomax model. The Baldwin-Barth model needs improvement for the prediction of pressure-gradient induced separation and mixing layer growth. Higher order turbulence models including multiscale models should be investigated for an accurate description of separated flows and mixing layer growth.

Acknowledgment

This work was supported by the Aerosciences Branch of NASA Johnson Space Center under Contract NAS 9-17900, with F. W. Martin Jr. as Technical Monitor.

References

[1]Reubush, D. E., and Runckel, J., "Effect of Fineness Ratio on Boattail Drag on Circular-Arc Boattail Afterbodies Having Closure Ratios of 0.50 with Jet Exhaust Mach Numbers Up to 1.3," NASA TN D-7192, May 1973.

[2]Mason, N. L., and Putnam, L. E., "Pitot Pressure Measurements in Flow Fields Behind Circular-Arc Nozzles with Exhaust Jets at Subsonic Freestream Mach Numbers," NASA TM 80169, 1979.

[3]Deiwert, G. S., Andrews, A. E., and Nakahashi, K., "Theoretical Analysis of Aircraft Afterbody Flows," *Journal of Spacecraft and Rockets*, Vol. 24, No. 6, 1987, pp. 496–503.

[4]Baldwin, B. S., and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, Jan. 1978.

[5]Peace, A. J., "Turbulent Flow Predictions for Afterbody Nozzle Geometries Including Base Effects," *Journal of Propulsion and Power*, Vol. 7, No. 3, 1991, pp. 396–403.

[6]Chien, K. Y., "Predictions of Channel and Boundary Layer Flows with a Low-Reynolds-Number Turbulence Model," *AIAA Journal*, Vol. 20, No. 1, 1982, pp. 33–38.

[7]Baldwin, B. S., and Barth, T. J., "A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows," NASA TM 102847, Aug. 1990.

[8]Kandula, M., and Buning, P. G., "Evaluation of Baldwin-Barth Turbulence Model with Thin-Layer Navier-Stokes Computation of an Axisymmetric Afterbody-Exhaust Jet Flowfield," AIAA Paper 93-0418, Jan. 1993.

[9]Buning, P. G., Chen, W.M., Renze, K. J., Sondak, D., and Chiu, I. T., "OVERFLOW / F3D User's Manual, Version 1.6n," NASA Ames Research Center, Moffett Field, CA, Oct. 1991.

Parallelization of the Factored Implicit Finite Difference Technique

Abraham N. Varghese*
Naval Undersea Warfare Center, Newport, Rhode Island 02841
and
Peter E. Raad†
Southern Methodist University, Dallas, Texas 75275

Introduction

PHYSICAL phenomena governed by differential equations give rise to systems of coupled algebraic equations, requiring the solution of matrices. The coefficient matrices that result from finite difference implementations in fluid dynamics are banded. In the last two decades, several solution techniques have been developed to take advantage of the diagonal nature of the resulting system of equations. All were developed though in the context of a serial computer, seeking to minimize the total work needed for a complete solution while maintaining a high level of accuracy and stability. This paper summarizes a research effort in the parallelization of the Beam and Warming factored implicit technique.[1] The parabolic heat equation and the nonlinear equations of curvilinear grid mapping were chosen as model equations. Additional information may be found in Varghese.[2]

Parallel Implementations and Results

The factored implicit (FI) technique was implemented in Fortran on the Sequent 20-processor Symmetry computer which has a tightly coupled multiple instruction stream–multiple data stream (MIMD), shared memory, multiprocessor architecture. Speedup calculations were made based on the definition of the speedup S being equal to the ratio of the most efficient serial algorithm time to the parallel algorithm time with P processors running. Parallel compilers such as the Sequent's provide parallelization directives such as *C$DOACROSS* (hereafter referred to in shorthand as *C$DO*), which automatically handles do loops, and *M_FORK* (hereafter referred to as *M_F*) which provides the programmer with selective con-

Received March 2, 1992; revision received Feb. 4, 1993; accepted for publication March 10, 1993. Copyright © 1993 by Abraham N. Varghese and Peter E. Raad. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Mechanical Engineer, Code 8232, Building #108.

†J. Lindsay Embrey Trustee Associate Professor, Department of Mechanical Engineering.

trol over specific portions of the code. What follows are three different parallel implementations of the FI technique. The first uses the automatic parallel directives whereas the next two use manual directives to assign work to each available processor.

C$DOACROSS Implementation

In the *C$DO* implementation, each processor is assigned to an x sweep along a row of grid points starting with $j = 2$ and ending with $j = NY - 1$. Each sweep involves calculating the right-hand side of the system of algebraic equations D_{ij} and the recursion coefficients of the tridiagonal solution α_{ij} and β_{ij}, followed by a recursive back substitution to calculate the unknowns at the intermediate time level ΔT^*_{ij} (see Refs. 1 and 2 for methodology and terminology). Once the x sweep is completed, each processor is assigned to a y sweep over a column of grid points, starting with $i = 2$ and ending with $i = NX - 1$. The y sweep involves the identical sequence of steps outlined for the x sweep except that the D_{ij} calculations are replaced with a read-only access to ΔT^*_{ij}. At the completion of this step, all processors are assigned to calculate the new temperature field T^{n+1}_{ij}, as well as to check for steady state by calculating the sum of the residuals ΔT^n_{ij}, both of which are insensitive to the order of execution. Figure 1 shows the actual speedups obtained for the FI solution of the heat equation with *C$DO*. Here, P represents the number of processors and S represents the speedup. A desirable characteristic in parallel implementations is the scalability of the method to computers with large numbers of processors. The deficiency that manifests itself in the steps observed in Fig. 1 would prevent this implementation from taking advantage of additional processors. It is even possible that with 10 or 20 additional processors, the speedup slope may turn negative due to additional processor overhead.

Therefore, the existence of the "steps" is an interesting phenomenon that deserves an explanation. Consider the run through the x sweep, for example, where each processor is assigned a row of grid points. As soon as a processor completes a row, it claims the next available one, and if none is available, it has to spin idly until all the other processors have finished the x sweep. Only then can the processors start the y sweep. It is thus possible that, say, 10 processors become available, but only nine rows remain. Therefore, a relationship between the number of rows and processors exists which determines the total number of row sweeps required. For the x integration, the number of total sweeps required is equal to the integer value of $[(NY - 2)/P + 0.5]$. To further elucidate this point, consider a specific example with $NY = 9$ and $P = 3$. Let each x sweep require a time of Δs. This example would then require a total of three sweeps with three processors going each time for a total wall clock time of $3\Delta s$. Now, let P increase to 4. First, the four processors proceed through the first four rows, then through the next four, but during the next overall sweep, work for only one processor is available and the other three must spin idly. Since all single x sweeps require the same amount of time Δs, the four-processor run would require $3\Delta s$ again. If P were increased to 5, however, only two sweeps would be needed for a total wall clock time of $2\Delta s$, and so on.

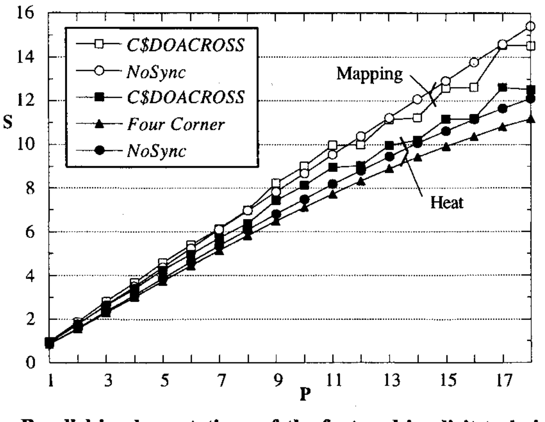

Fig. 1 Parallel implementations of the factored implicit technique.

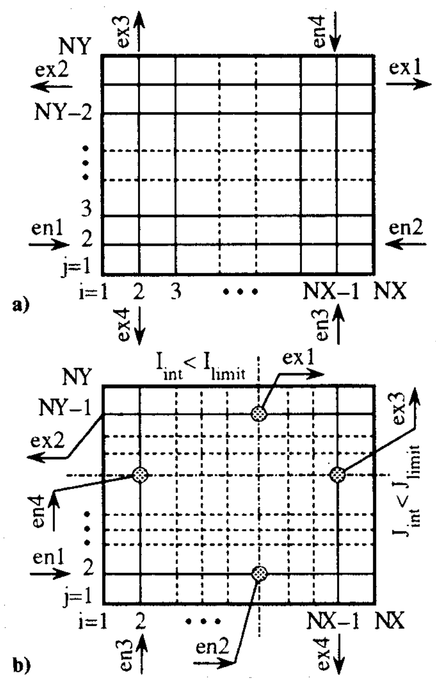

Fig. 2 Flow of execution of the a) four corner and b) NoSync implementations.

Four Corner Implementation

A new parallel implementation, called the four corner implementation, was investigated, where sweeps were subdivided into phases with an eye on eliminating processor synchronization between the x sweep and the y sweep and thus eliminating the steps. Elimination of synchronization without proper division of the x sweep into phases, however, will give rise to a race condition since the CPU time required by a single processor to complete the x sweep is larger than the CPU time required by a single processor to complete the y sweep. Therefore, the elimination of this synchronization without the introduction of any race condition motivated subdividing each of the x and y sweeps into two phases. The flow of execution of the four corner implementation is presented in Fig. 2a. First, the right-hand side D_{ij} and the recursion coefficients α_{ij} and β_{ij} along constant y coordinates are calculated from bottom left (en1) to top right (ex1). Here, en and ex refer to entry and exit, respectively, and the number suffix refers to the phase. When no rows remain unassigned, available processors proceed to the x back substitution phase starting from the bottom right (en2) toward the top left (ex2). After the penultimate row has been claimed, the next free processor starts the forward y sweep from the bottom right corner (en3) and ends in the top left corner (ex3). Finally, the y backward substitution phase starts from the top right (en4) and ends in the bottom left corner (ex4). Synchronization becomes necessary at this point to avoid a race condition at the bottom left corner since otherwise any free processor would begin the next time level solution by attempting to calculate $\alpha_{2,2}$ and $\beta_{2,2}$ before $T_{2,2}$ has been updated.

The speedups obtained for the FI technique by use of the four corner implementation for a 101 × 101 grid point domain are shown in Fig. 1. By comparison to the *C$DO* results, it can be seen that although the steps disappeared, lower speedups were achieved. Additional overhead was generated by the creation, allocation, and control of processors. Also, the *C$DO* directive is implemented at a lower machine level with predetermined processor control and, therefore, is more efficient than higher level directives such as *M_F*.

NoSync Implementation

The two preceding parallel implementations of the FI technique required a synchronization of processors between consecutive time integration steps, (n) and $(n + 1)$. Elimination of this synchronization should increase the speedup, but requires very careful restructuring of the different computational phases and appropriate means of accessing them. An algorithm, hereafter referred to as NoSync, was successfully designed with no synchronization. As shown in Fig. 2b, the flow of execution for this four-phase implementation begins with the calculation of the right-hand side D_{ij} and the recursive coefficients α_{ij} and β_{ij}, from bottom left (en1) to a location (ex1) set at an intermediate i value, $I_{\text{int}} < I_{\text{limit}} = NX - P$. Then, available processors start from $i = I_{\text{int}} + 1$ and $j = 2$ (en2), calculate D_{ij}, α_{ij}, and β_{ij} in the remaining forward x sweep, and perform the backward x sweep on the same row. Phase 2 terminates at the top left corner (ex2). Free processors begin the forward y sweep from bottom left (en3) up to an intermediate j value, $J_{\text{Jint}} < J_{\text{limit}} = NY - P$, and complete the third phase at the right end of J_{int} (ex3). In the fourth and last phase, unassigned processors start at the left end of $J_{\text{int}} + 1$ with the remainder of the forward y sweep, as well as the entire y back substitution, ending at the bottom right corner of the computational domain (ex4).

First, the load distribution in the NoSync implementation increased the work load in phase 1 to more equitably balance the loads between phases 4 and 1. Of course, care must be taken not to alter the load balance between phases 1 and 2 to the point that the race condition is only migrated from one phase interface to another rather than diminishing the possibility of its occurrence. Second, keeping $I_{\text{int}} < I_{\text{limit}} = (NX - P)$ ensures that processors in phase 1 do not interfere with any processor working in phase 4. Furthermore, choosing J_{limit} such that $P - 1$ rows are inaccessible to processors in phase 3 eliminates the possibility of a race condition between phases 2 and 3. Finally, it becomes possible to move the starting point of phase 3 to the bottom left corner and to allow phase 4 to exit on the right side of the computational domain as opposed to the left side in the four corner implementation. As a result, at the next time level the forward x sweep can begin more safely at the left bottom corner without the need for synchronization. Since the NoSync implementation involves four phases, processor interference is possible at any one of the four phase interfaces. By setting phase boundaries at $I_{\text{int}} = NX - P$ and $J_{\text{int}} = NY - P$, the occurrence of processor interference is avoided at the interfaces between phases 4 and 1 and between phases 2 and 3. Proper load balancing and efficient processor routing removed the possibility of processor interference along the other two interfaces.

Parallelization of the Mapping Equations

One may ask how will these implementations perform with more complex equations such as those solved in elliptic grid generation[3]? In this work, the time-dependent version of the two familiar, quasilinear, two-dimensional, curvilinear mapping equations[4] was solved. The solution at every time level represented a larger computational task than that of the heat equation and, as expected, yielded higher speedups. A comparison of resulting speedups from *C$DO* implementations of the heat and mapping equations can be made by examining Fig. 1. The NoSync implementation looks promising for large size problems running with a large number of processors. A continued advantage for the NoSync implementation on computers equipped with more processors is expected, therefore, when solving more complex problems on higher resolution grids (as is typically the case in CFD).

Conclusions

This work was aimed at investigating the suitability of parallel processing for the class of alternating direction implicit techniques on a tightly coupled, shared-memory, parallel architecture. One such technique, that of Beam and Warming, was parallelized on a Sequent Symmetry S81 parallel computer. A significant impediment in achieving near-theoretical speedups with implicit techniques is the requirement for processor synchronization at one or more stages during a time integration. The need for synchronization was eliminated by the careful use of load balancing, phase splitting, and processor routing. This in turn enabled the factored implicit technique to achieve higher speedups than possible with standard parallel compiler implementations. The highest speedups resulted from more complex equations, larger size grids, and more stringent convergence criteria. The principles presented in this work are general and should be applicable to other implicit techniques with minor technique-dependent alterations.

Acknowledgment

Funding for this work by the Department of Defense under DARPA Project 5-25085 of Contract MDA-093-86-C-0182 is gratefully acknowledged.

References

[1]Beam, R. M., and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393–402.

[2]Varghese, A. N., "Surface Texture Effects on Ultra-Thin Finite-Width Gas Bearings," Ph.D. Thesis, Civil and Mechanical Engineering Dept., Southern Methodist Univ., Dallas, TX, May 1991.

[3]Thompson, J. F., "A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure," *Computer Methods in Applied Mechanics and Engineering*, Vol. 64, Nos. 1–3, 1987, pp. 377–411.

[4]Raad, P. E., and White, J. W., "A 'Transient' Automated Mapping Procedure for Complex Geometries," *Numerical Grid Generation in Computational Fluid Mechanics*, edited by S. Sengupta, J. Häuser, P. R. Eiseman, and J. F. Thompson, Pineridge Press Limited, Swansea, United Kingdom, 1988, pp. 237–245.

Effects of a Rear Stagnation Jet on the Wake Behind a Cylinder

R. Duke,* B. Shrader,* and J. Mo†
Memphis State University, Memphis, Tennessee 38152

Introduction

THE control of the wake flow behind bluff bodies such as cylinders and the subsequent improvement of the aerodynamic performance of the cylinder are interesting subjects for both fundamental and applied research because they have practical applications and illustrate the basics of modern fluid mechanics. It is known that when the flow Reynolds number based on the cylinder diameter exceeds some critical value, the cylinder will experience an oscillating lateral force. This oscillating force is caused by the asymmetric flow pattern in the downstream wake, which is known as the von Kármán vortex street. Because of the massively separated wake, the cylinder also experiences a considerable drag. Experimental evidence shows that if the flow Reynolds number is over 100, the averaged drag coefficient on the cylinder is about 1.0 (Ref. 1). Of the two parts of drag, pressure or form drag is directly related to the wake behavior, whereas the friction drag is very small compared to the form drag. In fact, for a cylinder, the

Received Oct. 17, 1992; revision received Feb. 24, 1993; accepted for publication March 9, 1993. Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Graduate Assistant, Department of Mechanical Engineering. Student Member AIAA.

†Assistant Professor, Department of Mechanical Engineering. Member AIAA.