JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 33, No. 5, September–October 1996

Engineering Notes

ENGINEERING NOTES are short manuscripts describing new developments or important results of a preliminary nature. These Notes cannot exceed 6 manuscript pages and 3 figures; a page of text may be substituted for a figure and vice versa. After informal review by the editors, they may be published within a few months of the date of receipt. Style requirements are the same as for regular contributions (see inside back cover).

Hypersonic Nonequilibrium Parallel Multiblock Navier–Stokes Solver

Salvatore Borrelli,* Antonio Schettino,† and Pasquale Schiano‡
Centro Italiano Ricerche Aerospaziali, Capua 81043, Italy

Introduction

TO respond to the always increasing needs of the aerospace community and the scientific world in general, in recent years there has been a proliferation of new computer architectures that are different from classical serial computers: the massively parallel computers. These machines seem to be the most promising for achieving order-of-magnitude increases in computational power at reasonable cost.

One of the most effective and widely adopted techniques for using these kinds of machines efficiently consists in a decomposition of the topological domain of the problem, so that the calculations and the data are spread among different processors. In this case, additional treatment of boundary conditions is necessary, and data must be exchanged among neighboring subdomains.

In this Note, an application of this technique to a solver for hypersonic flows is made. After a brief description of the code, the masked multiblock technique (MBT) is discussed in more detail. Finally, the parallelized code is tested on different machines to show the efficiency that can be obtained with an increasing number of processors.

Flow Solver

The flow solver is based on a finite volume technique, with the fluxes at the interfaces calculated using a flux difference splitting solver (Pandolfi formulation) with a high-order essentially not oscillatory scheme.[1,2] Viscous two-dimensional plane and axisymmetric flowfields can be solved on multiblock grids for hyperenthalpic flows: in fact, chemical and vibrational nonequilibrium are taken into account, using several chemical models.[3] Local grid refinement is also possible; however, this feature is under testing and is not considered here.

The evaluation of the chemical source terms in the species conservation equations is deactivated for temperatures lower than a fixed level, either because at low temperatures some chemical models do not work well or because the influence of the chemistry does not justify the higher computational load; as will become clear in the following, this feature is very important for applying the MBT.

Received Feb. 10, 1995; revision received Feb. 26, 1996; accepted for publication Feb. 29, 1996.

*Aerothermodynamics Supervisor, Aerothermodynamics, Computational Laboratory Department, P.O. Box 81043.

†Researcher, Aerothermodynamics, Computational Laboratory Department, P.O. Box 81043.

‡Supercomputing, Computational Laboratory Department, P.O. Box 81043.

Masked MBT

For a multiblock structured grid-based code, such as our H2NS code, splitting the domain and assigning one or more entire blocks to each processor would not be efficient enough to guarantee each processor an identical or nearly identical computational workload. Sometimes it is necessary to use blocks with very different numbers of points; furthermore, because of fluid-dynamics nonlinearities, the computational load of each block can vary during the evolution of the solution. To override these difficulties an approach has been introduced, called masked multiblock,[4,5] that is capable of guaranteeing optimal computational load balancing among processors. This strategy meets the two following requirements: 1) the technique itself does not affect the accuracy of the converged solution; and 2) the technique itself does not have to modify the grid topology, which is user-defined and represents an input for the sequential–parallel code. Basically the idea consists in creating a virtual topology, which can be considered a sort of mask that may be placed on the real topology, completely independently of the grid blocks. Therefore, to each processor can be assigned one or more grid blocks or parts thereof.

In any case, for reactive fluxes a static partitioning can turn out to be inadequate. In fact, as mentioned before, the source terms are very small at low temperatures, and therefore their computation can typically be avoided in the parts of the flowfield with temperatures lower than 1000–1500 K; this means that in these zones the computation time is lower than in the rest of the flowfield, and therefore a fixed domain decomposition produces inferior computational load balancing. This is not the only case in which a dynamic load balancing is useful; for example, it can also be used for multizone calculations (mixed Euler and Navier–Stokes) or for turbulent flows with a dynamically computed transition point; another important case is when a check is made on the asymptotic flowfield to avoid useless calculations in the zones still distant from the shock wave. In all of these cases, a procedure to balance the computational load in a dynamic way among the processors can be used. A brief description of it is shown in the following.

Each N iterations, a check is made to verify the computational load of each processor. If the unbalancing is higher than a fixed percentage, the slave processes send the grid and the flowfield to the master process, which makes a new domain subdivision changing the limits of the masked blocks. Finally, the master resends the data to the slaves. Generally, this operation is needed only at the beginning of the computation, when the flow is changing rapidly. This is the reason why it is made only after having checked whether the load balancing has sensibly varied. However, this procedure allows for having a final domain subdivision that guarantees the best load balancing.

Results

The above-described methodologies, used to develop the parallel implementation of the H2NS code, have been tested by using a double-ellipse multiblock topology. The geometry is the same used in the Antibes Workshops; however, a 0-deg angle of attack has been chosen to emphasize the canopy shock that is one of the most critical points of this problem. The target parallel machines are summarized in Table 1; the message passing among processors is handled by using the PVM 2.4 (on the Convex Metaseries) and PVM 3.1 (on the Intel Paragon).[6] Obviously the solution has been verified to be the same in all of the cases, independently of the machine used and of the number of processors.

Table 1 Characteristics of machines used

Machine	No. of N processors	Peak rate per processor, M flops
Convex 3420	2	100
Convex 3860	6	240
Convex metaseries	8	198
Intel paragon	98	60

Table 2 Navier–Stokes calculations on coarse grid

Machine	No. of processors	Time, s	Speedup	Efficiency
CX 34	1	93,181	—	—
CX 38	1	33,002	—	—
CX MPP	1	25,601	—	—
CX MPP	2	13,157	1.94	0.95
CX MPP[a]	2	12,988	1.97	0.97
CX MPP	4	6,690	3.82	0.95
CX MPP[a]	4	6,674	3.83	0.96
CX MPP	6	5,104	5.01	0.83
CX MPP[a]	6	4,990	5.13	0.85
CX MPP	8	4,898	5.22	0.65
CX MPP[a]	8	4,685	5.44	0.68

[a]Run with dynamic balancing.

Table 3 Navier–Stokes calculations on fine grid

Machine	No. of processors	Time, s	Speedup	Efficiency
CX 38	1	81,810	—	—
CX MPP	1	67,217	1	1
CX MPP	2	34,787	1.93	0.96
CX MPP[a]	2	34,023	1.98	0.99
CX MPP	4	19,129	3.51	0.88
CX MPP[a]	4	17,453	3.85	0.96
CX MPP	6	12,933	5.19	0.86
CX MPP[a]	6	12,455	5.40	0.9
CX MPP	8	10,047	6.69	0.84
CX MPP[a]	8	9,387	7.16	0.89
Intel	1	27,500	1	1
Intel	2	14,074	1.95	0.975
Intel	4	7,249	3.8	0.95
Intel	8	3,757	7.32	0.915
Intel	15	2,250	12.22	0.81

[a]Run with dynamic balancing.

In Tables 2 and 3, results are shown in terms of time, speedup, and efficiency. Two grid levels have been used: a 4879-point coarse grid and a 9758-point fine grid. In both cases, the grid is composed of seven blocks; note that the blocks have quite different numbers of points. In particular, the blocks near the nose and the canopy region are finer, these being the most critical regions; that is why assigning each grid block to each processor leads to inferior load balancing.

The results, in terms of speedup and efficiency, are good in all cases. Obviously, in the fine-grid case, where the amount of calculation is greater than with the coarse grid, the speedup is better on account of the decrease of the overall overhead introduced by message passing among processors. Also, the gain from the use of dynamic load balancing is significant over the fine grid but is not very important with the coarse grid. The main reason is that this gain is greater at the beginning of the computation, when the points involved in the evaluation of production terms increase continuously; in the first case, the flowfield is still evolving with the chosen number of iterations, whereas in the second case a large number of iterations have also been performed after the stabilization of the chemistry.

Moreover, it must be said that the chemical model used here does not generate large ratios of computational load among cells where the production terms must be evaluated and the others. However, it is expected that using more sophisticated chemical models (e.g., taking into account also the ionization), or in the other cases discussed in the previous section, the benefits coming from the use of DLB algorithm should be notable.

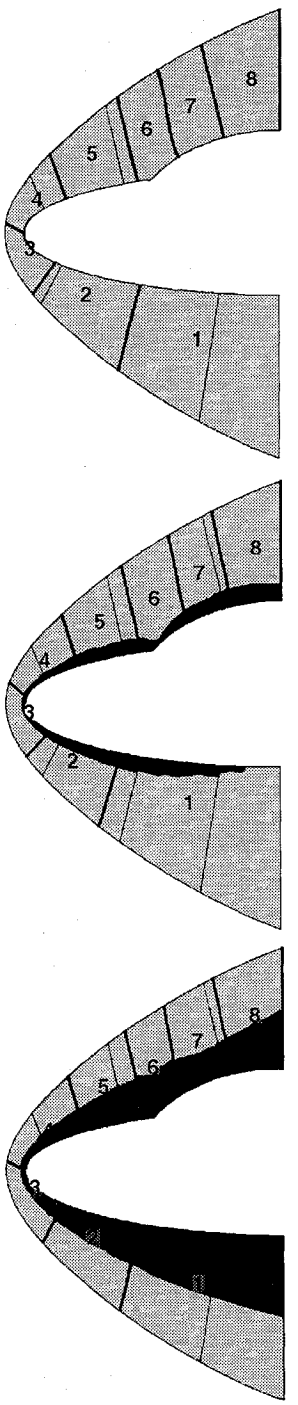

Fig. 1 Evolution of the masked blocks during the calculation: eight processors. The bold lines are the interfaces between processors; the thin lines are the interfaces between grid blocks.

Because the Intel Paragon is much slower than the Convex Metaseries, performing all of the 10,000 iterations on it as for the Convex was not possible; thus only 1000 iterations were made.

In Fig. 1 an example of dynamic load balancing is reported for the case with eight processors. The dark zone is the one in which the temperature is higher than T_{min}, and therefore the chemical production terms are calculated. In the first plot the initial masked topology, computed simply by assigning each processor the same number of points, can be clearly seen. In the second figure, the evolution of the chemistry, following the shock wave, is emphasized with the dark zone; it can be seen that the parts with less chemistry become greater because their computational load (for each point) is lower.

Obviously, the balancing cannot be done every few iterations, because this requires some maneuvers, particularly for message passing, that need time and delay the process. In this case, the balancing is checked every 200 iterations; in any case, the update of the

Table 4 Scalability test: cold test, one-block grids, 1500 iterations

No. of processors	Time, s		
	1025 × 65 points	513 × 65 points	257 × 65 points
1	—	—	917
2	—	926	466
4	944	480	262
8	506	267	226
16	342	203	—
32	296	—	—

masked blocks is made only if the difference between the averaged computational load and the one performed by one of the processors is higher than a fixed percentage (20% for these calculations).

Finally, some more tests have been made to check the scalability. Such tests cannot be made easily on the hot case, because on increasing the numbers of points, the computational load does not rise proportionally, since the ratio between the numbers of points with a temperature higher and lower than the limit temperature generally varies. Therefore, the test has been made only on the cold case, on a one-block grid. The computations have been performed at three grid levels: grid 1, 257 × 65; grid 2, 513 × 65; and grid 3, 1025 × 65. The results are shown in Table 4, in which the scalability can be seen to be generally good.

Conclusions

A hypersonic Navier–Stokes nonequilibrium code has been parallelized using the masked MBT with an automatic load-balancing algorithm. A number of tests have been made over a double ellipse; the original grid consisted of seven blocks, each one with a different number of points. Balancing is automatically controlled during the computation to take into account the different load due to the chemical terms. The results show good efficiency, particularly over the fine grid. The effect of dynamic balancing is also significant for these cases.

Other applications of the algorithm are under test: in particular, local grid refinement is being extended to the parallel version. Furthermore, it could be convenient to make a check to recognize the points where the computation can be avoided (i.e., the freestream points far enough from the shock wave); in that case the automatic load balancing would be very useful. Finally, it can be said that the procedure used seems to be fundamental in all the future applications of this code, if flowfields over greater grids (e.g., three-dimensional cases) are to be computed.

References

[1]Borrelli, S., and Pandolfi, M.,"An Upwind Formulation for the Numerical Prediction of Nonequilibrium Hypersonic Flows," *Proceedings of 12th International Conference on Numerical Methods in Fluid Dynamics* (Oxford, England, UK), edited by J. W. Morton, Springer–Verlag, 1990, pp. 416–420.

[2]Borrelli, S., and Schettino, A., "Influence of Chemical Modelling on Hypersonic Flow," *Proceedings of First European Symposium on Aerothermodynamics for Space Vehicles*, edited by B. Battrick, SP-318.23, European Space Agency, European Space Research and Technology Centre, 1991, pp. 335–341.

[3]Schettino, A., Borrelli, S., and De Filippis, F., "Influence of Transport and Thermokinetic Models in Free-Flight and in Plasma Wind Tunnel Tests," *Proceedings of the 5th International Symposium on Computational Fluid Dynamics* (Sendai, Japan), edited by H. Daiguji, Japan Society of Computational Fluid Dynamics, 1993, pp. 75–80.

[4]Borrelli, S., and Schiano, P., "A Nonequilibrium Hypersonic Flow Calculation on a Massively Parallel Computer," *Proceedings of Parallel CFD 1991* (Stuttgart, Germany), edited by K. G. Reinsch et al., Elsevier, Amsterdam, 1991, pp. 59–73.

[5]Borrelli, S., Matrone, A., and Schiano, P., "A Multiblock Hypersonic Flow Solver for a Massively Parallel Computer," *Proceedings of Parallel CFD 1992* (New Brunswick, NJ), edited by R. B. Pelz, A. Ecer, and J. Hauser, Elsevier, Amsterdam, 1992, pp. 25–37.

[6]Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., *PVM, Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, MITT Press, London, 1994.

I. D. Boyd
Associate Editor

Lower–Upper Symmetric Gauss–Seidel Scheme Exhibiting Asymmetric Vortices over Slender Bodies

Soo Jung Hwang* and Oh Hyun Rho†
Seoul National University,
Seoul 151-742, Republic of Korea

Nomenclature

$\hat{A}, \hat{B}, \hat{C}$	= Jacobian matrix of convective flux vectors
C_Y	= side-force coefficient, side force/$[q\pi D^2/4]$
D	= cylinder diameter
$D_\xi^+, D_\eta^+, D_\zeta^+$	= forward difference operators
$D_\xi^-, D_\eta^-, D_\zeta^-$	= backward difference operators
$\hat{E}, \hat{F}, \hat{G}$	= convective flux vectors
$\hat{G}_v$	= viscous flux vector in radial coordinate ζ
I	= identity matrix
J	= Jacobian of transformation
M_∞	= freestream Mach number
$\hat{Q}$	= conserved variable vector
q_∞	= freestream dynamic pressure
$\hat{R}$	= residual vector
Re_D	= Reynolds number based on cylinder diameter
t, τ	= time
x, y, z	= Cartesian coordinates
α	= angle of attack
Δ	= correction $()^{n+1} - ()^n$, where n denotes time level
ξ, η, ζ	= generalized curvilinear coordinates in the streamwise, circumferential, and radial direction.

Introduction

A PERFECT numerical simulation of asymmetric vortical flow around slender bodies at high angle of attack, which was observed experimentally,[1,2] is extremely difficult to obtain accurately. One of the difficulties lies in that it suffers from numerical errors inherently induced in the process of computation. Earlier researchers found out experimentally,[1,2] as well as numerically,[3–9] that minute asymmetric disturbance might cause large asymmetry in the entire flowfield. Siclari and Marconi[5] demonstrated asymmetric flow solutions by solving Navier–Stokes equations with conical assumptions obtained without preimposing any perturbation. A recent numerical study by Levy et al.[9] indicated that asymmetric errors introduced in the diagonalization process developed a spurious asymmetry in the flow.

A consequence of the previous results has been the assumption that the steady-state solution that depends only on the right-hand side of the algorithm, which has been generally accepted, does not hold for the high angle-of-attack flows. In this Note, a recently developed lower–upper symmetric Gauss-Seidel (LU-SGS)[10] algorithm, has been applied without introducing any perturbation intentionally. The scheme does not symmetrically factorize the flux Jacobian of the left-hand side. Asymmetric vortical flows have been obtained numerically[11,12] simply through direct application of LU-SGS scheme. The origin of the numerical asymmetry has also been investigated.

Received May 30, 1995; presented as Paper 95-1799 at the AIAA 13th Applied Aerodynamics Conference, San Diego, CA, June 19–22, 1995; revision received March 15, 1996; accepted for publication May 3, 1996.

*Graduate Research Assistant, Department of Aerospace Engineering.
†Professor, Department of Aerospace Engineering. Member AIAA.