Conf-950130--4
SAND 94-1402C

AIAA-95-0572

A DOMAIN DECOMPOSITION STUDY OF MASSIVELY PARALLEL COMPUTING IN COMPRESSIBLE GAS DYNAMICS

C. Channy Wong, Fred G. Blottner, and Jeffrey L. Payne
Sandia National Laboratories
Albuquerque, NM

Moeljo Soetrisno
Amtec Engineering, Inc.
Bellevue, WA

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

**33rd Aerospace Sciences
Meeting and Exhibit**
January 9-12, 1995 / Reno, NV **MASTER**

For permission to copy or republish, contact the American Institute of Aeronautics and Astronautics
370 L'Enfant Promenade, S.W., Washington, D.C. 20024

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
DT

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

A DOMAIN DECOMPOSITION STUDY OF MASSIVELY PARALLEL COMPUTING IN COMPRESSIBLE GAS DYNAMICS

C. Channy Wong^{*}, Fred G. Blottner[†], and Jeffrey L. Payne^{*}

Sandia National Laboratories, Albuquerque, NM 87185-0825

Moeljo Soetrisno[‡]

Amtec Engineering, Inc., Bellevue, WA 98009

Abstract

The appropriate utilization of massively parallel computers for solving the Navier-Stokes equations is investigated and determined from an engineering perspective. The issues investigated are: (1) Should strip or patch domain decomposition of the spatial mesh be used to reduce computer time? (2) How many computer nodes should be used for a problem with a given sized mesh to reduce computer time? (3) Is the convergence of the Navier-Stokes solution procedure (LU-SGS) adversely influenced by the domain decomposition approach? The results of the paper show that the present Navier-Stokes solution technique has good performance on a massively parallel computer for transient flow problems. For steady-state problems with a large number of mesh cells, the solution procedure will require significant computer time due to an increased number of iterations to achieve a converged solution. There is an optimum number of computer nodes to use for a problem with a given global mesh size.

I. Introduction

When applying massively parallel (MP) processing technology in the area of computational fluid dynamics (CFD), there are at least two popular approaches to formulate an engineering problem. The first approach is to decompose the spatial domain into many subdomains and load each computer node with a different subdomain. Then each computer node will be responsible to process information within the assigned spatial subdomain. When solving the flow field within a subdomain, each node, from time to time, requires exchanging information with the neighboring subdomains using the message passage programming calls to update the boundary conditions

throughout the computation process. Thus this approach requires generating many fictitious computational boundary cells, also known as communication cells. Hence the disadvantage of this approach is that it requires modifying the computer program to include many message passing calls (*send, receive,*). Many times, however the work of adding and testing the message passing call is a labor intensive task. Also there are no standardized subroutine calls for message passing. Different MP computers use different subroutine calls. The modified program may only work for a specific MP machine, making it difficult to justify the investment of time and effort to modify the program. Moreover the massively parallel computing technology is not yet mature and is still evolving. Changes made in today's codes may not be applicable tomorrow. However, a standard message passing call, MPI is emerging.

The other approach is to work with the kernels of the discretized governing equations and apply a distributed sparse matrix solver developed specifically for a MP computer.¹ The latter approach has an advantage of requiring a less demanding code modification effort by adopting a generic parallelized sparse matrix solver. However it has the drawback of overloading the computing work load in the host node while under-utilizing all other acquired computer nodes. This happens because only the solver portion of the program is operated in the parallel mode. Then majority of the acquired computer nodes will be idle when the code is performing other work. This could be a significant part of the elapsed time for large-scale problems.

Our interest is to solve large scale problems in a reasonable length of time and this requires a MP computer with a large number of computer nodes. The spatial domain decomposition approach then becomes very attractive because it can provide a well-balanced loading and an excellent computational speedup. This section addresses the technical issues and accomplishments obtained with the domain decomposition technique in order to achieve

^{*} Senior Member of Technical Staff

[†] DMTS, Fellow AIAA

[‡] Senior Engineer, Member AIAA.

good performance with massively parallel computing from an engineering perspective.

The computer time per time step on a parallel computer depends on the usual computational time plus the communication time required to transmit information between the nodes. The communication time is influenced by the decomposition scheme. In addition, the convergence of the solution to a steady-state is also influenced by decomposition approach. These issues will be addressed in this paper.

II. Influence of Multi-subdomains on Computer Time

A. Decomposition Schemes and Computer Time

There are many ways to decompose a computational domain and map these subdomains into MP computers. For the simplicity of algorithm implementation, this study will only consider those decomposition schemes which produce subdomains with simple and regular geometries. For a two-dimensional (2D) rectangular computational domain, one can use either (a) a strip decomposition, either column-oriented (Figure 1a) or row-oriented (Figure 1b), or (b) a patch decomposition (Figure 1c). Since each subdomain needs to exchange information with its neighboring subdomains, there will be a communication overhead cost. To simplify the analysis, this study will first consider a fixed-sized problem with a 2D domain as in Figure 1. Consider a domain of interest that has m^2 total mesh cells. For each communication with its neighboring subdomain, the number of single precision data exchanged in each message will be directly related to the number of mesh cells adjacent to the boundary.

For a strip decomposition (Figure 1a and Figure 1b), each subdomain will exchange with its neighboring subdomain a message of length of mk , where k is the number of state variables in the calculation. The transfer of information between the various nodes is performed simultaneously. Thus the communication time for each exchange message at all the nodes will be

$$t_{comm} = \gamma + 4\phi mk \quad (1)$$

where γ is the communication start-up time and ϕ is the time required to send one byte of data. For each interior subdomain, it requires communicating with two neighboring subdomains. For those subdomains adjacent to the physical boundary, each will only communicate with one neighboring subdomain. Besides communicating with the neighboring subdomains, a global communication may occur at each time step to broadcast the necessary information such as the size of time step (Courant-Friedricks-Levy condition). The communication time for a global broadcast

Figure 1: Domain Decomposition Schemes for 2D Case.

is highly dependent on the parallel computer architecture. For the Intel Gamma machine, this message passing can be accomplished via nearest-neighbor data exchanges along each dimension of the hypercube. Thus this global communication can be achieved in only $\log_2 P$ time.

If only considering the message passing to the neighboring subdomains, one can easily identify the disadvantage of this strip decomposition scheme. For a fixed-sized problem, the communication time remains constant no matter how many computer nodes are used. However increasing the number of computer nodes does slightly increase overall global communication time. On the other hand, an increasing number of computer nodes does speed up the computation process. More computer nodes help because the total tasks will be divided up among the computer nodes. Each node will then have less tasks to perform. Hence the amount of computation in the computer program is proportional to the number of mesh cells m^2/P , where P is the number of computer nodes. The computational time per time step becomes

$$t_{comp} = cm^2/P \quad (2)$$

An interesting parameter for determining overhead cost and for comparing different decomposition schemes is the communication-to-computation ratio. The ratio is defined as follows:

$$Ratio \sim n_a \left[\left(\frac{\gamma}{m^2} \right)^P + \left(\frac{4\phi k}{m} \right)^P \right] + n_b \left[P \frac{\log P}{\log 2} \right] \quad (3)$$

where n_a is the total number of communication with its neighbors and n_b is the total number of global broadcast inside the computer program. Thus Eq (3) shows that the communication-to-computation ratio increases proportionally to the number of computer nodes P . Hence communication overhead becomes a large portion of the cost when utilizing a strip decomposition scheme and a large number of computer nodes.

For the patch decomposition (Figure 1c), consider a case with a 2D computational domain being decomposed into many square subdomains, each with a dimension of $m/\sqrt{P}$. With this setup, the message length in each information exchange is $mk/\sqrt{P}$. The communication time for each message exchange becomes:

$$t_{comm} = \gamma + 4\phi mk/(\sqrt{P}) \quad (4)$$

Unlike the strip decomposition, the patch decomposition will have each interior subdomain communicate with four neighboring subdomains. For the subdomain adjacent to the physical boundary, it will only communicate with two neighboring subdomains. Hence the communication time with the neighboring subdomains for the patch decomposition decreases as the number of computer nodes increases. In addition to communicate with the neighboring subdomains, there always exists global communications which requires $\log_2 P$ time. The new overhead cost, a ratio to compare communication time to computation time becomes:

$$Ratio \sim n_a \left[\left(\frac{\gamma}{m} \right) P + \left(\frac{4\phi k}{m} \right) \sqrt{P} \right] + n_b \left[P \frac{\log P}{\log 2} \right] \quad (5)$$

For long messages, the second term dominates the ratio and is a very slowly increasing function of P , number of computer nodes used.

In summary, for any problem, estimating the total communication cost is not a simple and straight forward matter. It is highly dependent on the characteristics of the problem, the decomposition schemes, and the number of message passing calls in the computer program. This functional analysis intends to provide some insight of the dominant parameters so that an engineer can establish some guidelines to approach a massively parallel computer for solving a technical problem. So far, according to the functional analysis given above, the patch decomposition is better than the strip decomposition for a 2D problem. The total time that the patch decomposition scheme takes to communicate with its neighboring subdomains decreases when an increasing number of computer nodes are used. Thus, the patch decomposition scheme has potential to de-

liver a better speed-up and turn-around time than the strip approach.

B. A Case Study

Results from a numerical simulation of a hypersonic blunt body flow problem is consistent with the analytical analysis and also demonstrates that the patch decomposition scheme is better than the strip decomposition scheme. The patch scheme has a relatively faster elapsed time and a better convergence rate (Table 1). These results were generated by a parallel version of the INCA code, known as PINCA.² The PINCA code was used to calculate a hypersonic flow field over a generic blunt conic body. The freestream flow conditions are: velocity = 6.2 km/s, pressure = 461 N/m², temperature = 241K. For these PINCA calculations, a perfect gas approximation was assumed.

In this study, the global grid system is constant with 80 by 60 computational cells. The spatial domain is decomposed according to the number of computer nodes used in the simulation and the strip or patch decomposition scheme is used. Table 1 shows that the patch decomposition scheme with the aspect ratio of subdomain close to 1 has a faster elapsed time and better convergence rate (i.e. the case with a local mesh of 20x20). If the concern is to achieve a convergence criterion, those simulations using a strip decomposition scheme (i.e. the third, sixth, and eighth cases) will require a much longer elapsed time than the results given in Table 1

Table 1: Comparison of Code Performance Between Strip and Patch Decomposition Schemes

Nodes	idiv	jdiv	Local mesh	Elapse time	log (min L ²)
16	4	4	20x15	2367	-7.78
16	8	2	10x30	2341	-8.11
16	16	1	5x60	2571	-4.03
12	4	3	20x20	2212	-8.09
12	2	6	40x10	2370	-7.41
12	1	12	80x5	2396	-4.88
6	2	3	40x20	2628	-8.10
6	1	6	80x10	2906	-3.81
PINCA Performance on Intel/Gamma for 2000 steps; idiv and jdiv are the number of subdomains in x and y direction, respectively.					

For a fixed-sized two-dimensional problem, both analysis and the case study show that the patch decomposition scheme has a better elapsed time performance and convergence rate than the strip decomposition scheme. However we need to apply caution when generalizing this finding to any problem and any solution technique used in other codes. Another problem that has different dominant flow physics may produce different results. For example, the case studied is a steady state, perfect gas problem and PINCA uses a point LU-SGS solution technique to obtain a solution. This setup leads to a weak coupling between subdomains. Thus, if using a point LU-SGS solution technique, spatial domain decomposition has a relatively small effect on any timing performance or convergence rate. For cases such as chemical reactive flow problems that use line and plane iteration techniques, coupling between subdomains may be strong. Decomposing a strongly coupled region may decrease the convergence rate and degrade the performance. Hence, the efficiency of the decomposition scheme may be different depending on the characteristics of the problem. In some cases, a dynamic decomposition scheme may be justified.

III. Influence of Multi-subdomains on Steady-State Convergence

A. Convergence Rate

The domain decomposition approach is a popular approach on parallel computers. The Schwarz alternating procedure³ established the convergence of the domain decomposition approach with overlapping subdomains. This approach is known to have slow convergence with certain solution methods when many subdomains are used as discussed in the work of Braaten.⁴

Solution performance of various *transient* fluid dynamic problems on parallel computers have been previously evaluated by many authors. This part of the investigation is concerned with solution performance of solving *steady-state* fluid dynamic problems on a parallel computer. The steady-state solution of a problem requires a sufficient number of time steps, N , to reduce the solution residual to a small value. Number of time steps to obtain the steady-state solution depends on the solution technique, number of global mesh cells, the number of subdomains used in the problem, and the desired level of convergence of the solution. In an initial study, a boundary layer code has been used to evaluate the expected solution performance on a parallel computer. In addition, two problems have been solved with the INCA Navier-Stokes code. The steady-state solution convergence properties of the two problems have been determined for various number of subdomains.

B. Similar Boundary Layer Solutions with Domain Decomposition

The present investigation determines the influence of the amount of overlap between the subdomains on the solution convergence for the Falkner-Skan boundary layer flow problem. The governing Navier-Stokes equations are reduced to a steady one-dimensional problem in terms of the similarity coordinate and two dependent variables for the velocity components. The interface coupling approach employed in the boundary conditions for a subdomain is investigated. The solution of the difference equations within a subdomain is solved directly with a block tridiagonal solver and with two iterative techniques. The Jacobi and SSOR iterative methods are used. An iterative approach is required for multi-dimensional problems as direct solvers are not practical for large problems. For the one-dimensional boundary layer problem, a direct solver can be used to determine the influence of the subdomain interface coupling method on the solution convergence. The solution convergence properties of the three solvers is determined for the problem with multi-subdomains. In addition, idealized parallel computer times are estimated for steady-state fluid dynamic problems with the three solvers. The desired properties of the difference equation solution technique are determined.

Estimates of Solution Times for Steady-State Problems

The solution time required on a parallel computer is estimated for steady-state fluid dynamic problems. The present estimates are overly optimistic as the communication times required between processors are not considered and it is assumed that the complete solution can be performed in parallel. Also it is assumed that the computational domain is divided into subdomains and the solution in each subdomain is performed on a separate node of the parallel computer.

The number of steps required to obtain a converged steady-state solution depends on various parameters and is determined from numerical solutions. For the present domain decomposition approach with weak coupling between the subdomains, the number of iteration steps N to reduce the solution residual to a specified value can be a function of the number of mesh points and the number of subdomains and is expressed as

$$N = C_1 J_d^m P^n = C_1 J_{sd}^m P^{n+m} \quad C_1 \text{ is a constant}$$

$$J_d = \text{Number of mesh cells in domain} = PJ_{sd} \quad (6)$$

$$J_{sd} = \text{Number of mesh cells in subdomain}$$

The exponents m and n are parameters determined by the solution method and the coupling utilized between the

subdomains. The time required to obtain the solution on a parallel computer is

$$t(P) = \phi N/P = C_0 J_d^{m+1} P^{n-1} \quad \text{with } \phi = C_0 J_d \quad (7)$$

and ϕ = Time per step required by the solution method.

The values of the parameters for several methods are given in Table 2 and the justification for these values are given later.

Table 2: Behavior of Solution Techniques for Steady-State Problem

Solution Technique	m	n	Parallel Computer Time	
			Fixed	Increasing
Jacobi	2	0	$\sim P^{-1}$	$\sim P^2$
SSOR	1	1	Constant	$\sim P^2$
Tridiagonal	0	2	$\sim P$	$\sim P^2$
Ideal	0	0	$\sim P^{-1}$	Constant

For a *fixed size problem* with J_d held constant, the serial and parallel computation times are

$$\begin{aligned} t(1) &= C_0 C_1 J_d^{m+1} \\ t(P) &= C_0 C_1 J_d^{m+1} P^{n-1} \sim P^{n-1} \end{aligned} \quad (8)$$

For an *increasing size problem* with the number of mesh points per subdomain held constant, the serial and parallel computation times are

$$\begin{aligned} t(1) &= C_0 C_1 J_{sd}^{m+1} P^{m+1} \\ t(P) &= C_0 C_1 J_{sd}^{m+1} P^{m+n} \sim P^{m+n} \end{aligned} \quad (9)$$

The computer times are given in Table 2 for the various solution techniques.

Ideal Properties of Solution Technique

The Jacobi solution technique has properties that result in the desired speedup on a parallel computer. However, the technique is still too slow even on a massively parallel computer to be useful. Another solution technique is required.

For a *fixed size problem* with J_d held constant, the ideal performance should have parallel computation time inversely proportional to the number of processors as given by Eq (8). This requires $n = 0$ and the number of itera-

tion steps for the steady-state solution should be independent of the number of subdomains that are used. For an *increasing size problem* with the number of mesh points per subdomain held constant, the ideal performance should have parallel computation time constant as given by Eq (9). This requires $m + n = 0$ and the number of iteration steps for the steady-state solution should be independent of the number of mesh points in the complete domain.

Results for Fluid Dynamic Problem

The difference equations for the Falkner-Skan boundary layer flow problem have been solved with the Jacobi, symmetric Gauss-Seidel with over relaxation (SSOR), and tridiagonal methods. The first item investigated is the influence of the overlap between the subdomains. The solution has been obtained with the tridiagonal solver with 4 subdomains and 16 mesh point per subdomain. The residual variation with iteration has been determined with various overlap. The residual is determine from

$$\begin{aligned} \text{Residual} &= L_2 \text{ norm of } R = \sqrt{\sum_{j=j_s}^{j_e} |R_j|^2} \\ |R_j|^2 &= R_m^2 + R_c^2 \end{aligned} \quad (10)$$

The variables R_c and R_m are residuals for the continuity and momentum equations. The results of this study are given in Figure 2. Increasing the overlap gives a significant improvement in the solution convergence. However in the following discussion, the results in this study use an overlap of 4 unless indicated otherwise. Flores⁵ has obtained solutions to the Euler and Navier-Stokes equations with a zonal approach and has investigated the effects on convergence rate due to changing the zonal boundary overlap region. He concluded that the amount of overlap had no significant ef-

Figure 2: Influence of subdomain overlap on solution convergence.

fects on convergence rates. This issue needs further evaluation.

Next we will determine the exponents m and n for those cases with overlapping subdomains by analyzing the convergence rate. First the convergence properties of the tridiagonal solver have been obtained with 4 subdomains used with various number of mesh points per subdomain. The number of steps to obtain a converged solution is approximately proportional to the number of mesh points per subdomain ($N - J_{sd}$). The exponent m in Eq (6) becomes $m \approx 1$. Next for the case of 16 mesh points per subdomain and various number of subdomains, the number of steps to obtain a converged solution is approximately proportional to the number of subdomains squared ($N \sim P^2$). The exponents for P in Eq (6) is approximately 2 and $n \approx 1$.

The number of iterations required to reduce the residual to 10^{-10} has been determined. These results with one subdomain for the three solvers are presented in Figure 3. The number of steps with the tridiagonal solver is independent of the total number of mesh points in the domain. The number of steps with the symmetric Gauss-Seidel solver is approximately proportional to the number of mesh points. The number of steps with the Jacobi solver is proportional to the number of mesh points squared. Solutions have been obtained also with 4 subdomains. The number of steps with the tridiagonal solver is now approximately proportional to the number of mesh points. The number of steps with the symmetric Gauss-Seidel solver is still nearly proportional to the number of mesh points. The number of steps with the Jacobi solver is still proportional to the number of mesh points squared and independent of the number of subdomains.

The number of iterations required to reduce the residual to 10^{-10} has also been determined for these cases with a fixed number of points per subdomain with various number of subdomains. These results for the three solution

Figure 3: Influence of number of mesh points on number of iteration steps.

methods are given Figure 4. The iteration steps required for a converged solution for these solution methods are approximately proportional to the number of subdomains squared, $N \sim P^2$. This shows that $m + n \approx 2$ in Eq (9) for the three solution methods. However, there is a significant difference in the number of iteration steps required with the various solution methods for a given number of subdomains.

Figure 4: Influence of number of subdomains on the solution convergence with a fixed number of mesh points in the subdomains.

C. Nozzle Flow Problem

The inviscid flow in a supersonic nozzle is used to investigate the convergence properties of the Lower-Upper Symmetric Gauss-Seidel (LU-SGS) solution technique. PINCA uses the LU-SGS technique to solve the difference form of the governing equations. The nozzle flow problem is interesting because the steady form of the governing equations are hyperbolic and a marching solution approach could be used to obtain the flow prediction with one sweep along the nozzle. The LU-SGS solution technique requires many sweeps down and back, along the nozzle to obtain a converged solution to the steady flow. The inflow to the nozzle is supersonic and has the velocity, temperature, and pressure specified as follows:

$$\begin{aligned} u &= 4000 \text{ m/s} & v &= 0 & w &= 0 \\ T &= 9000 \text{ K} & p &= 99610 \text{ N/m}^2 \end{aligned} \quad (11)$$

In addition, a perfect gas model is assumed. The gas is assumed to be air with a molecular weight of 28.966 and $\gamma = 1.4$. The density from the equation of state is 0.03856 kg/m^3 and the Mach number is 2.103 at the inflow location.

The nozzle is axisymmetric with an increasing radius which is specified by the relation

$$r = 0.125L \left[1 + \sin\left(\frac{\pi x}{2L}\right) \right] \quad (12)$$

The entry to the nozzle is at $x = 0$ and the exit is at $x = L$ with the nozzle length L equal to 2 meters. A uniform mesh is used along the nozzle and a stretched mesh is used across the nozzle. The stretching is that developed by Roberts (see transformation 1 on page 249 of Anderson et al.⁶). The present calculations use a stretching with $\beta = 1.5$. Downstream in the nozzle an oblique shock wave structure develops. This flow is similar to the nozzle flow investigated experimentally by Back and Cuffel⁷ where oblique shocks occur.

The number of time steps to reduce the residual to 10^{-10} is given in Table 3 for various meshes investigated.

Table 3: Time steps required for converged solution of nozzle flow problem with one domain.

Grid	Global Mesh, M	Time Steps, N
36 x 36	1,296	265
72 x 72	5,184	440
144 x 144	20,736	820
40 x 10	400	168
144 x 10	1,440	194
1280 x 10	12,800	731

Only one domain is used. As the number of mesh cells M increases, the number of time steps increases approximately proportional to $M^{0.4}$. The impact of domain decomposition on the number of time steps required to obtain converged solution for the nozzle flow problem is illustrated in Figure 5 and Figure 6. These solutions indicate that the present solution method for the nozzle flow problem is not significantly influenced by the domain decomposition approach. For the cases investigated (as many as 64 subdomains), the number of time steps for a converged solution is increased less than 20%. However, the number of time steps is increasing as the number of subdomains is increasing. This behavior is shown in Figure 7 where the linear variation is obtained from

$$N/N_{P=1} = 1 + C(P - 1) \quad (13)$$

The constant C has been specified such that this approximation is a reason fit to the data points. Even with a massively parallel computer with 1000 nodes, the number of time steps is expected to be perhaps double due to the large number of subdomains.

Figure 5: Residual behavior in subdomain indicated for nozzle flow problem with various number of subdomains and global mesh is 145 x 145.

Figure 6: Residual behavior in subdomain indicated for nozzle flow problem with various number of subdomains and for two global meshes.

D. Hypersonic Flat Plate Boundary Layer Problem

It is known that some solution techniques with physical domain decomposition are adversely influenced when there are subdomains within the boundary layer flow.⁸ The flat plate boundary layer problem is considered a significant test of the convergence properties of the present Navier-Stokes solution technique with domain decomposition. The hypersonic flow of a perfect gas over a flat plate of length $L \approx 1m$ and at a temperature of 1000K has been used as a sample problem. The freestream Mach number is 15. The other flow condition are

Figure 7: Increase in time steps to obtain converged solution to the nozzle flow problem with various meshes and various number of subdomains.

$$\begin{aligned} u_{\infty} &= 6723.5 \text{ m/s} & p_{\infty} &= 114.04 \text{ N/m}^2 \\ T_{\infty} &= 500 \text{ K} & Re &= \rho_{\infty} u_{\infty} L / \mu_{\infty} \approx 200,000 \end{aligned} \quad (14)$$

A parabolic mesh system has been used and is described in Wong, et al.⁹ The Mach number in the flow field is shown in Figure 8. The flow is from left to right and due to the displacement of the boundary layer, a shock wave is generated. The boundary layer fills nearly half of the shock layer for these flow conditions.

The results for a one domain solution of the hypersonic flat plate problem with various size meshes is given in Figure 9. Four of the curves show the residuals for different mesh systems as a function of time steps with JMAX

Figure 8: The Mach number distribution for the hypersonic flow over a flat Plate.

Figure 9: Residual as a function of the number of time steps for the hypersonic flat plate problem with different size global meshes and with one subdomain.

(number of mesh cells across boundary layer) = 40, 80, 160, and 320 for these curves respectively. Additional results have been obtained with JMAX held constant and with various values of IMAX (number of mesh cells along flat plate). The convergence is nearly independent of IMAX for this problem; for example, when IMAX = 20, 40, and 80 with JMAX = 160 the residual curves have very nearly the same variation with time steps. These results show that the solution convergence for this problem is dominated and determined by the small mesh spacing near the surface.

When the global mesh is divided into subdomains, the influence on convergence is shown in Figure 10. These results indicate the number of time steps to obtained a steady-state solution is increased somewhat. With 64 subdomains,

Figure 10: Influence of the number of subdomains on the solution convergence for the hypersonic flat plate problem.

the number of time steps required to obtain a converged steady-state solution is increased approximately 34%. The number of time steps to obtain a steady-state solution with the $\log(\text{residual})$ equal -10 has been determined from Figure 9 where one domain is used. These results have been extrapolated to larger meshes with the relation

$$\text{Number of time steps for converged solution} = N = 2M^{3/4} \quad (15)$$

This prediction is compared to the hypersonic flat plate problem results in Figure 11. These estimates of the number of time steps should be increased when solutions are obtained on a parallel computer due to dividing the problem into subdomains. For the following estimates of results on a parallel computer, this increase in number of time steps is neglected.

Steady-State Convergence Results

The Navier-Stokes code INCA (without any message passing) is used for the computer results in this part of the investigation. The number of time steps to obtain converged solutions with one solution domain is given in Figure 12 for the problems investigated in this study plus two other problems that use the basic LU-SGS solution technique. The authors¹⁰ of the INCA code have obtained the solution for the flow of nitrogen over a cylinder at Mach 6.13. The gas is treated as a binary gas mixture of nitrogen atoms and molecules with finite chemical reactions. Three grids 22x22, 34x34, and 50x50 were used and the L_2 residual was reduced 11.55, 6.69, and 8.21 orders of magnitude for the three cases, respectively. The 3D flow over the Apollo command module has been obtained with a modified form of the INCA code.¹¹ A mesh with 110 x 26

Figure 11: Number of time steps required for steady-state solution to the hypersonic flat plate problem with various size meshes and one domain.

Figure 12: Number of time steps required to obtain converged solutions to various problems with one domain.

x 70 cells was used for this solution with a perfect gas model and with a finite rate air model. The L_2 residual was reduced 3 orders of magnitude and required 7000 iterations for the perfect gas case and 13,000 iterations for the real gas case. Only the real gas result is presented in the figure. This figure shows there is a significant increase in the number of time steps required to obtain a converged solution as the problem size increases. Also there is an increase in time steps required as the gas complexity increases.

The influence of using multi-subdomains on steady-state convergence with this code has been investigated. The supersonic inviscid flow in a diverging nozzle has been used as one test case. With up to 64 subdomains, the number of iterations for convergence is increased by less than 20%. The second problem investigated is the hypersonic flow over a flat plate where the complete flow field of shock wave and boundary layer is obtained. With as many as 32 subdomains normal to the flat plate, the number of iterations required for a converged solution is increased by less than 34%. The results of this investigation indicate that the LU-SGS scheme is weakly influenced by the number of subdomains.

IV. Choosing Number of Computer Nodes

A. Introduction

Estimates of the performance of the PINCA code on a parallel computer can be made if the communication times for the code on the parallel computer and the convergence properties of the solution technique are known. From the work described in section IIB, estimates of the computer communication times can be made. The convergence properties of this code have been investigated in section III for

the inviscid supersonic flow in a divergent nozzle and for the hypersonic flow over a flat plate. The hypersonic flat plate case is used as a sample problem for the illustrations presented in this section. The performance of this sample problem on a parallel computer and the Cray computer is estimated.

Initially estimates are made for the communication and computational times required for the solution for one time step. These include the conditions (the number of nodes and global mesh size) required to minimize execution time. In addition, the optimum conditions are considered to minimize the computational cost. These initial results are only important for transient flow problems. Then the number of nodes is determined to minimize the computer execution time, lapse time or turn-around time to obtain the steady-state solution to the sample problem with a given mesh size. This is the optimum operating conditions from the user point of view.

B. Estimated Computer Times for Transient Problems

The parallel computer time required to solve the Navier-Stokes equations for one time step can be represented as

$$t_s = t_{comm} + t_{comp} \quad t_{comm} = t_{exch} + t_{glob} \quad (16)$$

- t_{comm} Time to communicate all information to the nodes
- t_{comp} Time to calculate one step of the solution
- t_{exch} Time to exchange overlap information between domains
- t_{glob} Time to send global information to all the nodes

The times in Eq (16) are a function of the number of cells in the global mesh M , the Navier-Stokes solution technique and gas model employed, and the properties of the parallel computer. For the Intel Gamma computer, the nodes are connected with a hypercube network and the times are expected to be of the following forms:

$$\begin{aligned} t_{exch} &= a(M)P & t_{glob} &= b \ln P \\ t_{comp} &= cM/P \end{aligned} \quad (17)$$

The coefficients a , b , and c in Eq (17) are determined from solutions obtained on the parallel computer. The time for exchange of information is neglected in this part of the study as it is usually small relative to the global time and this assumption simplifies the results. For solutions on the Intel Gamma computer, initially the coefficients are assumed to have the following values for the logarithmic communication time:

$$a = a(M) = 0 \quad b = 0.5 \quad c = 0.00125 \quad (18)$$

For a fixed sized mesh, the computer time per step will initially decrease as the number of subdomains or nodes is increased but with a sufficiently large number of nodes the time per step starts to increase. This behavior is estimated for the hypersonic flat plate problem and these results are given in Table 4. The efficiency in this table is defined as

$$\begin{aligned} \text{Speed up} &= S = t_s(P=1) / t_s(P) \\ \text{Efficiency (\%)} &= \eta = \\ 100S/P &= 100(t_{comp}/t_s) \end{aligned} \quad (19)$$

The node-time is an indication of the computational cost of a solution as it determines the length of time the computer is used and the fraction of the parallel computer being utilized. The node-time is defined as

$$\text{Node-time per step} = \tau = P t_s \quad (20)$$

Table 4: Fixed Mesh Problem - Global Mesh 80 x 160

P	Mesh on Node	t_{comp} (s)	t_{comm} (s)	t_s (s)	η (%)	Node Time (s)
1	80x160	16.0	0.00	16.0	100	16.0
2	80x80	8.0	0.35	8.35	95.8	16.7
8	40x40	2.0	1.04	3.04	65.8	24.3
32	20x20	0.5	1.73	2.23	22.4	71.5
128	10x10	0.125	2.43	2.55	4.90	326
512	5x5	0.031	3.12	3.15	0.99	1613

C. Minimizing the Elapsed Time

With a fixed-sized global mesh for the hypersonic flat plate problem, the behavior of the various computer times for one time step is illustrated in Figure 13 as the number of nodes is increased. The computation time decreases as the number of nodes increases while the communication time increases. Therefore, the minimum computational time per step occurs when the number of nodes is given by

$$aP^2 + bP - cM = 0 \quad (21)$$

When the time to exchange information between subdomains can be neglected, then the optimum number of nodes to minimize computer time is

Figure 13: Computer time per step as function of number of nodes for the hypersonic flat plate problem with logarithmic communication time.

$$P_{opt} = \frac{c}{b}M = 0.0025M \quad (22)$$

The optimum node mesh for minimum computer time is determined for the case with the same number of mesh cells m_{node} in each direction (2D problem) and becomes with the parameters given in Eq (18)

$$M_{node} = M/P_{opt} = b/c = 400 \quad (23)$$

$$m_{node} = \sqrt{M_{node}} = 20$$

The optimum node mesh (number of cells in mesh on a node) is independent of the global mesh size or the number of nodes being used. The optimum number of meshes cells for various number of nodes is given in Figure 14 for flow problems with a perfect gas model and with the logarithmic communication time of Eq (17). The results obtained with two global meshes with the Intel/Gamma parallel computer are also presented in this figure. The predicted behavior with logarithmic communication time does not match the actual behavior on the parallel computer. An analysis of the computational time results on parallel computers indicated that the communication time behavior is not as expected. The communication time has been determined for the Intel Gamma computer with a fixed mesh (two meshes used) with the number of nodes varied. These results for a perfect gas problem (Figure 15) indicate the communication time is proportional to the number of nodes and is independent of the global mesh size. The solid lines in this figure indicate the communication time behavior when it is proportional to the number of

Figure 14: Optimum number of computer nodes for given number of mesh cells.

Figure 15: Communication time on parallel computers as a function of the number of computer nodes for perfect gas model.

nodes. An additional solution on the Intel Gamma computer is shown in this figure for a variable mesh where the node mesh size is held fixed and the number of nodes is varied. Again the communication time is proportional to the number of nodes. The communication time on the Intel Paragon computer has also been determined for a variable mesh problem (Figure 15). The Intel Paragon computer uses a two-dimensional mesh to connect the nodes and the global communication time is expected to be proportional to the square root of the number of nodes. For all results on the Intel parallel computers, the communication time is proportional to the number of nodes. Further work is required to determine why this behavior is occurring. The communication time is expressed as

$$t_{comm} = bP^n \quad (24)$$

The results obtained on the Intel parallel computers show $n = 1$ while the expected behavior on the Paragon should have $n = 1/2$. Solution to flow problems with real gas effects have also been obtained¹² and the linear behavior of the communication time with the number of nodes is observed.

For the case given in Eq (24), the optimum conditions to minimize computer time is obtained from the computer time per step and this time is expressed as

$$t_s = bP^n + c(M/P) \quad (25)$$

The coefficients in Eq (25) have been obtained from computations on two Intel parallel computers and the results are given in Table 5.

Table 5: Coefficients for Linear Communication Time

Gas Model	Intel Gamma		Intel Paragon	
	b	c	b	c
Perfect Gas	0.028	0.0012	0.018	0.00073
Real Gas	0.032	0.0036	0.017	0.0024

For the communication time given by Eq (24), the optimum number of nodes or mesh size to minimize computer time is determined to be the following:

$$P_{opt} = \left[\frac{cM}{nb} \right]^{\frac{1}{1+n}} \quad M_{opt} = \left(\frac{nb}{c} \right) P^{1+n} \quad (26)$$

The optimum mesh on each node becomes

$$M_{node} = \frac{M_{opt}}{P_{opt}} = \left[\frac{nbM^n}{c} \right]^{\frac{1}{1+n}} \quad (27)$$

The estimated performance with a parallel computer for solving the hypersonic flat plate problem with various global meshes for the linear communication time is given in Table 6. The efficiency is 50.3% for all cases in this table. The estimated performance with a parallel computer for solving the hypersonic flat plate problem with various global meshes for the logarithmic communication time is given in Table 8. The appropriate scaling of the problem varies with the behavior of the communication time. For the linear communication time and a 2D problem, the node

Table 6: Conditions for Minimum Execution Time for the Hypersonic Flat Plate Problem with Linear Communication Relation

Nodes P	Node Mesh	t_{comp} (s)	t_{comm} (s)	t_s (s)	Node Time (s)
1	5x5	0.018	0.018	0.036	0.036
4	10x10	0.073	0.072	0.145	0.580
16	20x20	0.292	0.288	0.580	9.280
64	40x40	1.168	1.152	2.32	148.5
256	80x80	4.672	4.608	9.28	2376
1024	160x160	18.69	18.43	37.12	38011

Table 7: Conditions for Minimum Execution Time for the Hypersonic Flat Plate Problem with Logarithmic Communication Relation (Node mesh 20x20)

P	Global Mesh	t_{comp} (s)	t_{comm} (s)	t_s (s)	η (%)	Node Time (s)
1	20x40	1.0	0.0	1.0	77.6	1.55
2	20x80	0.5	0.35	0.85	63.4	3.79
8	40x80	0.5	1.04	1.54	53.6	8.96
32	80x160	0.5	1.73	2.23	40.9	46.9
128	160x320	0.5	2.42	2.92	33.1	232
512	320x640	0.5	3.12	3.62	27.8	1106

mesh doubles in size in each coordinate direction while the number of nodes increases by a factor of 4 and the global mesh increases by a factor of 16. For this case, there is a quadratic relationship between the number of nodes and the number of mesh cells. For the logarithmic communication time and a 2D problem, the node mesh remains of fixed size while the number of nodes increases by a factor of 4 and the global mesh increases by a factor of 4. For this case, there is a linear relationship between the number of nodes and the number of mesh cells.

D. Minimizing the Node-Time

The operating conditions on a parallel computer can also be chosen to minimize the cost or minimize the node-time. The node-time for the linear communication time relation is

$$\tau = Pt_s = bP^2 + cM \quad (28)$$

Eq (28) shows that the number of nodes P should be as small as possible and just large enough to accommodate the desired problem. Since $M = PM_{node}$, the node mesh size to minimize cost is determined from

$$P = M / (M_{node})_{max} \quad (29)$$

$(M_{node})_{max}$ = Maximum node mesh possible

The mesh size on a node is made as large as the node memory can handle in order to minimize the number of parallel computer nodes required. The estimated performance for solving the hypersonic flat plate problem on the Paragon parallel computer with conditions to minimize the cost is given in Table 8. In these results it has been as-

Table 8: Computer Times per Step for Hypersonic Flat Plate Problem with Various Size Meshes and Conditions to Minimize Cost (Node mesh 100x100)

P	M	t_{comp} (s)	t_{comm} (s)	t_s (s)	η (%)	Node Time (s)
1	10^4	7.3	0.018	7.32	99.8	7.32
10	10^5	7.3	0.18	7.48	97.6	74.8
100	10^6	7.3	1.8	9.1	80.2	910
1000	10^7	7.3	18	25.3	28.8	25,300

sumed that the maximum mesh on a node is 100 x 100 due to the memory size available on the nodes. The variation of the total mesh cells with the number of nodes to minimize cost or node-time is given in Figure 14. For small problems, there is a significant difference between the number of nodes required to minimize cost and minimize computer execution time. For large problems, the appropriate operating conditions correspond to the minimum cost curve as the computer execution time curve can never exceed the cost curve as there is insufficient memory to operate in this region. For the computer execution time given by Eq (25), the number of nodes and the number of mesh cells must satisfy the following relations for the minimize computer execution time curve:

$$P < \left[\left(\frac{c}{nb} \right) (M_{node})_{max} \right]^{\frac{1}{n}} \quad M < P (M_{node})_{max} \quad (30)$$

For the minimized cost case, there is a linear relationship between the number of nodes and the number of mesh cells. The parallel computer time per step is compared to an estimate of the time required on a Cray computer. The Cray time for one domain is estimated from the relation

$$t_{Cray} = 2.5 \times 10^{-5} M \quad (31)$$

This comparison is given in Figure 16. The computer time obtained from Eq (25) with $n = 1$ and with the Paragon computer coefficients in Table 6 is given in this figure when the node mesh is held constant with 60 x 60 cells. This curve is close to the times measured on the Paragon computer which are given as the solid squares. The parallel computer times are smaller than the Cray results when the global mesh is greater than approximately 100,000 mesh cells. The curves for optimum conditions to minimize the execution time for both linear and logarithmic communication time are also given.

Figure 16: Estimate of optimum computer time per step for the hypersonic flat plate problem as the global mesh is changed.

E. Estimated Computer Times for Steady-State Problems Using the Optimum Number of Mesh Cells on a Node

The steady-state solution of a problem requires a sufficient number of time steps, N , to reduce the solution residual to a small value. The total computer time to obtain the steady-state solution becomes

$$t_T = t_s N \quad (32)$$

Number of time steps to obtain the steady-state solution depends on the solution technique, number of global mesh

cells, the number of subdomains used in the problem, and the desired level of convergence of the solution. For the sample hypersonic flat plate problem, the convergence properties of the INCA code have been investigated in an earlier section. When the global mesh is divided into subdomains, the results indicate the number of time steps required to obtain a steady-state solution is increased somewhat. The number of time steps to obtain a steady-state solution with the $\log(\text{residual})$ equal -10 has been determined from Eq (15). These estimates of the number of time steps should be increased when solutions are obtained on a parallel computer due to dividing the problem into subdomains. For the following estimates of results on a parallel computer, this increase in number of time steps is neglected. The estimates for the hypersonic flat plate problem with different size meshes is given in Table 9. In Figure 17, the parallel computer results are compared to the Cray computer results. For a large problem the parallel computer is faster than the Cray computer. However as the problem size increases, it requires more time steps to achieve a steady state solution. Hence large scale problems takes a significant amount of computer time.

Table 9: Total Computer Time for Sample Problem with Various Meshes

Nodes, P	Global Mesh, M	Time Steps, N	Computer Time, t_T (hour)
2	800	453	0.118
8	3,200	890	0.410
* Estimated

Figure 17: Estimate of total computer time for obtaining the steady-state solution to the hypersonic flat plate problem.

Table 9: Total Computer Time for Sample Problem with Various Meshes

Nodes, P	Global Mesh, M	Time Steps, N	Computer Time, t_T (hour)
32	12,800	2,414	1.55
128	51,200	7,084	5.69
512	204,800	19,254*	19.4
2048	819,200	54,459*	65.2
* Estimated			

F. Computer Time, Node-time and Efficiency with Non-Optimum Number of Mesh Cells on a Node

In the previous section, the number of mesh cells on a node has been chosen to minimize the parallel computer time required to obtain the solution of a problem with a specified number of global mesh cells. One also might want to choose the number of mesh cells on a node such that the computer cost (node-time) is minimized. The computer time has been estimated for four different values of the number of mesh cells on a node and these results are given in Figure 18. As predicted, the computer time is a minimum when the node mesh is 20×20 for the problem with any number of global mesh cells. The computer time is only significantly larger when the mesh is 80×80 .

The node-time has also been determined for different values of the number of mesh cells on a node and these results are presented in Figure 19. The parallel computer cost is reduced by minimizing the node-time which indicates that the node mesh should be 80×80 or as large as can be

Figure 18: Influence of node mesh size on the computer time required for the steady-state solution of the hypersonic flat plate problem.

Figure 19: Influence of node mesh size on the node-time required for the steady-state solution of the hypersonic flat plate problem.

accommodated by the memory available on the node. This is the typical situation where you can have fast computer turn-around time but it cost more. For the present problem, the node mesh of 40x40 is a good compromise where the computer time is only slightly larger than the minimum possible and the cost has been reduced significantly.

The speed-up or the efficiency of the solution process is one measure of how successful a code has been run on a parallel computer and has not been considered as yet. The efficiency as defined in Eq (19) is not important from the point of view developed in this analysis. The efficiency is given in Figure 20 for different node mesh sizes. Higher efficiencies are obtained as the mesh size on the nodes is increased. For the cases with high efficiency, the computational cost is reduced but the computer turn-around time can be significantly increased.

V. Conclusions

Maximizing the usage of each computer node is an important strategy to achieve good performance in massively parallel computing, especially when solving a large scale problem. In order to achieve a close-to-perfect scaleup and to determine the maximum problem size that the computer can handle, engineers should maximize the computing usage for each node first.¹² This also implies that before performing any calculation of a large scale problem, the computer code has been optimized for the single node performance. Figure 21 illustrates a preliminary guideline for simulating a large-scale fluid dynamics problem on a massively parallel computer. For a fixed-sized problem, it is better to start with the least number of computer nodes for the best node-time performance. If the total elapsed time is an important criterion, add a small number of computer

Figure 20: Efficiency of solutions on parallel computer with various mesh sizes on a node for the steady-state solution of the hypersonic flat plate problem.

nodes to enhance the performance. Too many computer nodes will produce a less satisfactory result because communication overhead becomes dominant.

Spatial domain decomposition is a natural approach to massively parallel computing in computational fluid dynamics. For the cases studied, patch decomposition scheme works better because it leads to less communication overhead. However to generalize this finding to any problem and any code is difficult because it depends on the characteristics of the problem and the numerical algorithm used.

The results from the convergence investigation indicate that properties of the solution method used for solving the difference equations for steady-state fluid dynamic problems is a critical issue for obtaining the desired performance on a parallel computer. None of the methods investigated in the initial boundary layer problem have the desired steady-state convergence properties. Solution meth-

Log of Problem Size

Figure 21: Ensemble Computing Performance Pattern (see Gustafson, et al.¹³)

ods are required with convergence rates that are nearly independent of the number of mesh points in the domain and are not greatly influenced by the number of subdomains. The results of this investigation indicate that the LU-SGS scheme is weakly influenced by the number of subdomains. The weakness in the LU-SGS solution method is the significant increase in the number of time steps required to obtain a steady state solution as the problem size increases.

A procedure has been developed for estimating the performance of the Navier-Stokes code PINCA on a parallel computer. This analysis has resulted in a performance code that can be used to determine the appropriate number of computer nodes or the number of node mesh cells to obtain the minimum computer turn-around time. The analysis requires solutions be obtained on a parallel computer which allows the constants in the computer time per step and the communication time per step relations to be determined. The optimum number of nodes to minimize computer execution time has been determined and is shown to be proportional to the number of mesh cells used in the problem when the expected communication time is logarithmic. Also the optimum number of node mesh cells is shown to be independent of the global mesh size and the number of nodes being used. Solutions with the PINCA code on parallel computers indicate the communication time is not as expected but is a linear function of the number of nodes. For this case, the optimum number of nodes to minimize computer execution time has been determined and shown to be proportional to the number of mesh cells used in the problem squared. The optimum number of nodes to minimize computation cost requires that the number of nodes should be the minimum number that can be used to handle the problem being solved. This requires that the node mesh be as large as can be accommodated by the nodes. For large problems, the computer execution time can not be reduced below the optimum computer time for minimizing cost.

Studies have been performed to determine the time steps required to obtain a converged steady-state solution for two problems. The number of time steps required to obtain a converged solution increases significantly as the number of global mesh cells increases. The introduction of a large number of subdomains to distribute the problem to the nodes of the parallel computer has a modest impact on the solution convergence properties. When the number of node mesh cells is increased above the optimum number, the computer turn-around time is increased but the node-time (cost) is reduced. The performance code can be used to evaluate this trade-off.

VI. References

1. D. Bailey, et al., "The NAS Parallel Benchmarks," NASA Technical Report RNR-91-002, NASA Ames Research Center, Moffett Field, CA, August 1991.
2. PINCA: A Massively Parallel Version of the Integrated Code Analysis, a joint-development project between Sandia National Laboratories, Albuquerque, NM, and Amtec Engineering Inc., Bellevue, WA, May 1994.
3. H. Schwarz, Gesammelte Mathematische Abhandlungen, Vol. 2, Springer, Berlin, 1980.
4. Mark E. Braaten, "Solution of Viscous Fluid Flows on a Distributed Memory Concurrent Computer," Int. J. for Numerical Methods in Fluids, Vol. 10, 889-905, 1990.
5. Jolen Flores, "Convergence Acceleration for a Three-Dimensional Euler/Navier-Stokes Zonal Approach," AIAA Journal, pp. 1441-1442, Sept. 1986.
6. Dale A. Anderson, John C. Tannehill, and Richard H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Corp., 1984.
7. L. H. Back and R. F. Cuffel, "Detection of Oblique Shocks in a Conical Nozzle with a Circular-arc Throat," AIAA J., Vol. 4, pp. 2219-2221, 1966.
8. Carl B. Jenssen, "Implicit Multi Block Euler and Navier-Stokes Calculations," AIAA paper 94-0521, January, 1994.
9. C. C. Wong, M. Soetrisno, F. G. Blottner, and J. L. Payne, "PINCA: A Scalable Parallel Program for Compressible Gas Dynamics with Nonequilibrium Chemistry," SAND94-2436, Sandia National Laboratories, Albuquerque, NM, January 1995.
10. Scott Eberhardt and Scott Imlay, "Diagonal Implicit Scheme for Computing Flows with Finite Rate Chemistry," J. Thermophysics and Heat Transfer, Vol. 6, pp. 208-216, April-June, 1992.
11. B. Hassan, G. V. Candler, and D. R. Olynick, "Thermo-Chemical Nonequilibrium Effects on the Aerothermodynamics of Aerobraking Vehicles," J. of Spacecraft and Rockets, Vol. 30, pp. 647-655, Nov. - Dec., 1993.
12. C. C. Wong, M. Soetrisno, F. G. Blottner, and J. L. Payne, "Implementation of a Parallel Algorithm for Thermo-Chemical Nonequilibrium Flow Simulation," AIAA Paper 95-0152, Jan. 1995.
13. J. L. Gustafson, G. R. Mantra, R. E. Benner., "Development of Parallel Methods for a 1024-Processor Hypercube," SIAM Journal on Scientific and Statistical Computing, Vol. 9, No. 4, July 1988, pp. 609-638.

Acknowledgment

This work was supported by the United States Department of Energy under contract DE-AC04-94AL85000 through the Laboratory Directed Research and Development (LDRD) program.