A97-32442

AIAA-97-1893

An Efficient Multiblock Method for Aerodynamic Analysis and Design on Distributed Memory Systems

J. Reuther[‡]
RIACS - NASA Ames Research Center
Moffet Field, CA 94035

J. J. Alonso[+]
Department of Aeronautics & Astronautics
Stanford University
Stanford, CA 94305

J. C. Vassberg[††]
Aerodynamic Design
Douglas Aircraft Co.
Long Beach, CA 90846

A. Jameson[*]
Department of Aeronautics & Astronautics
Stanford University
Stanford, CA 94305

L. Martinelli[†]
Department of Mechanical & Aerospace Engineering
Princeton University
Princeton, NJ 08544

The work presented in this paper describes the application of a multiblock gridding strategy to the solution of aerodynamic design optimization problems involving complex configurations. The design process is parallelized using the MPI (Message Passing Interface) Standard such that it can be efficiently run on a variety of distributed memory systems ranging from traditional parallel computers to networks of workstations. Substantial improvements to the parallel performance of the baseline method are presented, with particular attention to their impact on the scalability of the program as a function of the mesh size. Drag minimization calculations at a fixed coefficient of lift are presented for a business jet configuration that includes the wing, body, pylon, aft-mounted nacelle, and vertical and horizontal tails. An aerodynamic design optimization is performed with both the Euler and Reynolds Averaged Navier-Stokes (RANS) equations governing the flow solution and the results are compared. These sample calculations establish the feasibility of efficient aerodynamic optimization of complete aircraft configurations using the RANS equations as the flow model. There still exists, however, the need for detailed studies of the importance of a true viscous adjoint method which holds the promise of tackling the minimization of not only the wave and induced components of drag, but also the viscous drag.

INTRODUCTION

During the course of the last few years, there has been a concentrated effort within our group to develop fast and efficient methods for the solution of viscous fluid flows over complex aircraft configurations. The path to achieve this goal has seen numerous improvements in convergence acceleration techniques (multigrid, implicit residual averaging), viscous discretization algorithms, higher order dissipation schemes for shock capturing and boundary layer resolution, unstructured and multiblock grid approaches, and parallel implementations based on domain decomposition ideas [30, 50, 43, 49, 2, 1]. The combination of all these factors has resulted in a variety of flow solvers that adhere to the highest standards of accuracy and efficiency.

‡ Research Scientist, Member AIAA
+ Assistant Professor, Member AIAA
†† Senior Principal Engineer, Senior Member AIAA
* T. V. Jones Professor of Engineering, AIAA Fellow
† Assistant Professor, Member AIAA

AIAA Paper 97-1893

Although direct flow analysis of existing configurations has in the past provided the aircraft designer with invaluable information to overcome a wide range of problems, there is the need for Computational Fluid Dynamics (CFD) methods which, in addition, provide information about geometry changes that are necessary to improve an existing design with respect to a pre-specified figure of merit. It is within this framework where the utilization of fast solution techniques is of utmost importance.

Most effective aerodynamic optimization methods are based on the calculation of the derivatives of the figure of merit with respect to the design variables in the problem. Unfortunately, the calculation of these derivatives using the straightforward method of finite differencing is usually prohibitively expensive. While it is possible to perform aerodynamic optimization on a limited class of problems using the finite difference approach [19, 18, 51, 44, 42], large scale problems that are of the greatest engineering interest do not belong to this class. An alternative technique first suggested for problems involving partial differential equations by Lions [40], and extended for the treatment of compressible flow by Jameson [23, 24] is the control theory approach. This methodology employs control theory applied to systems governed by partial differential equations to derive a co-state or adjoint system of equations. This adjoint equation has similar complexity to the flow solution, and allows the calculation of the complete gradient of the figure of merit with a cost which is essentially independent of the number of design variables in the problem.

The computational cost of performing viscous based design is considerably larger than for design using the Euler equations because: a) the number of mesh points must be increased by a factor of about five to resolve boundary layers and wakes, b) there is the additional cost of computing the viscous terms and a turbulence model, and c) Navier–Stokes calculations generally converge much more slowly than Euler solutions because of stiffness arising from the highly stretched boundary layer cells. Therefore, the computational feasibility of viscous design hinges on the development of a rapidly convergent Navier–Stokes flow solver which is able to handle complex configurations and is efficiently implemented on the current generation of distributed memory architectures.

With this in mind, the logical approach to the solution of the aerodynamic design problem is to link together fast iterative solvers and the adjoint solution methodology in order to produce a computational method which can address the needs of the aircraft designer: high solution accuracy, fast turnaround, geometric complexity, and automated shape design. Even with the use of an adjoint solver, large scale design problems using the Navier-Stokes equations that are considered in this work require massive computational resources. Future work will place even more extreme demands on the computational power needed. Therefore, it was decided to attempt to exploit the power of emerging distributed memory parallel computers with efficient standardized message-passing implementations. Thus, much emphasis in this paper has been placed, not only on demonstrating the viability of performing automatic designs on complex configurations, but also on minimizing the communication overhead incurred by mapping the method onto either parallel computers or clusters of workstations.

In this paper we present one of the possible variations of the adjoint based design technique for complex geometries, where the flow and adjoint solvers have been implemented using a multiblock strategy. Both the Euler and Reynolds Averaged Navier-Stokes equations are used to solve drag minimization problems. In both circumstances, the adjoint system solved to obtain the sensitivity of the figure of merit with respect to the design variables is based on the *inviscid* equations only. The authors feel that the effective use of a viscous adjoint in a realistic design environment will require much further development and validation work. Furthermore, the approach presented here is a natural evolution of our previous work [50, 43, 49] which had already been extended to treat inviscid flows over complex configurations. The work directed towards the improvement of the parallel performance of the method was motivated by an interest in demonstrating the performance of the method on computational platforms which do not possess the bandwidth and latency that is realizable on highly integrated parallel machines.

CONTROL THEORY FORMULATION FOR SHAPE DESIGN

The presentation of the control theory approach to optimal design is well documented elsewhere [23, 28], and only a brief summary is given here.

The progress of the design procedure is measured in terms of a cost function I, which could be, for example, the drag coefficient or the lift to drag ratio. For the flow about an airfoil or wing, the aerodynamic properties which define the cost function are functions of the flow-field variables (w) and the physical location of the boundary, which may be represented by the function $\mathcal{F}$, say. Then

$$I = I(w, \mathcal{F}),$$

and a change in $\mathcal{F}$ results in a change

$$\delta I = \frac{\partial I^T}{\partial w}\delta w + \frac{\partial I^T}{\partial \mathcal{F}}\delta \mathcal{F} \qquad (1)$$

in the cost function. Using control theory, the governing equations of the flow field are introduced as a constraint in such a way that the final expression for the gradient does not require multiple flow solutions. This corresponds to eliminating δw from (1).

Suppose that the governing equation R which expresses the dependence of w and $\mathcal{F}$ within the flowfield domain D can be written as

$$R(w, \mathcal{F}) = 0. \qquad (2)$$

In our current work, R may be expressed by either the Euler or Navier-Stokes equations. Then δw is determined from the equation

$$\delta R = \left[\frac{\partial R}{\partial w}\right]\delta w + \left[\frac{\partial R}{\partial \mathcal{F}}\right]\delta \mathcal{F} = 0. \qquad (3)$$

Next, introducing a Lagrange multiplier ψ, we have

$$\begin{aligned}\delta I &= \frac{\partial I^T}{\partial w}\delta w + \frac{\partial I^T}{\partial \mathcal{F}}\delta\mathcal{F} \\ &- \psi^T\left(\left[\frac{\partial R}{\partial w}\right]\delta w + \left[\frac{\partial R}{\partial \mathcal{F}}\right]\delta\mathcal{F}\right) \\ &= \left\{\frac{\partial I^T}{\partial w} - \psi^T\left[\frac{\partial R}{\partial w}\right]\right\}\delta w \\ &+ \left\{\frac{\partial I^T}{\partial \mathcal{F}} - \psi^T\left[\frac{\partial R}{\partial \mathcal{F}}\right]\right\}\delta\mathcal{F}.\end{aligned}$$

Choosing ψ to satisfy the adjoint equation

$$\left[\frac{\partial R}{\partial w}\right]^T \psi = \frac{\partial I}{\partial w} \tag{4}$$

the first term is eliminated, and we find that

$$\delta I = \mathcal{G}\delta\mathcal{F}, \tag{5}$$

where

$$\mathcal{G} = \frac{\partial I^T}{\partial \mathcal{F}} - \psi^T\left[\frac{\partial R}{\partial \mathcal{F}}\right].$$

The advantage is that (5) is independent of δw, with the result that the gradient of I with respect to an arbitrary number of design variables can be determined without the need for additional flow-field evaluations. In the case that (2) is a partial differential equation, the adjoint equation (4) is also a partial differential equation and determination of the appropriate boundary conditions requires careful mathematical treatment.

The computational cost of a single design cycle is roughly equivalent to the cost of two flow solutions since the the adjoint problem has similar complexity. When the number of design variables becomes large, the computational efficiency of the control theory approach over traditional finite differencing strategies, which require direct evaluation of the gradients by individually varying each design variable and recomputing the flow field, becomes compelling.

Once equation (5) is established, an improvement can be made with a shape change in the direction of the negative gradient

$$\delta\mathcal{F} = -\lambda\mathcal{G}$$

where λ is positive, and small enough that the first variation is an accurate estimate of δI. Then

$$\delta I = -\lambda\mathcal{G}^T\mathcal{G} < 0.$$

After making such a modification, the gradient can be recalculated and the process repeated to follow a path of descent until a minimum is reached. Variations on the optimization procedure which allow for the treatment of structural and aerodynamic constraints can be readily incorporated in this approach.

MULTIBLOCK FLOW SOLVER

Multiblock strategy

FLO107-MB is a three-dimensional, multiblock, Euler and Navier-Stokes flow solver suitable for the solution of external and internal flows around complex configurations. The discretization of the governing equations of the flow is accomplished using a cell-centered finite volume method. The flow domain is divided into a large number of small subdomains, and the integral form of the conservation laws

$$\frac{\partial}{\partial t}\int_{\mathcal{D}} \mathbf{w}dV + \int_{\mathcal{B}} \mathbf{F}\cdot d\mathbf{S} = 0$$

is applied to each subdomain. Here $\mathbf{F}$ is the flux function which can include the viscous fluxes in the case of the Navier–Stokes equations, $\mathbf{w}$ is the vector of flow variables, and $d\mathbf{S}$ is the directed surface element of the boundary $\mathcal{B}$ of the domain $\mathcal{D}$. The use of the integral form has the advantage that no assumption of the differentiability of the solution is implied, with the result that it remains a valid statement for a subdomain containing shock waves. In general the subdomains could be arbitrary, but in this work we use the hexahedral cells of a multiblock body-conforming curvilinear mesh. Discretizations of this type reduce to central differences on a regular Cartesian grid, and in order to eliminate possible odd-even decoupling modes allowed by the discretization, some form of artificial dissipation must be added. Moreover, when shock waves are present, it is necessary to upwind the discretization to provide a non-oscillatory capture of discontinuities. The current version of the multiblock flow solver accomplishes this task using either a switched scalar dissipation scheme or the more sophisticated Convective Upstream Split Pressure (CUSP) approach, coupled with an Essential Local Extremum Diminishing (ELED) formulation. Details on these techniques and an extensive validation of the scheme for both inviscid and viscous flow, can be found in [26, 27, 54].

In order to apply the finite volume technique to the solution of flows around complex configurations, we have chosen to implement a *multiblock* strategy. In a multiblock environment, a series of structured blocks of varying sizes is constructed such that these blocks fill the complete space and conform to the surface of the geometry of interest. This segmentation of the complete domain into smaller blocks avoids the topological problems present in constructing a grid around complex configurations and multiply connected regions. The general strategy in the solution procedure of the multiblock flow solver is to construct a halo of cells which surrounds each block and contains information from cells in the neighboring blocks. This halo of cells, when updated at appropriate times during the numerical procedure, allows the flow solution inside each block to proceed independently of the others.

This approach requires establishing the number and location of halo cells adjacent to block boundaries and constructing lists of halo cells and their internal counterparts in the global mesh. In our case, we have chosen to carry out these setup procedures as part of a pre-processing module. During the pre-processing step, a two-level halo is constructed around each block. The requirement of this double halo results from the necessity to calculate all the necessary fluxes for the internal cells of each block without reference to additional cell locations outside the block in question. In particular, the second differences used for the third order dissipation terms require the values of the flow variables in the two neighboring cells on all sides of the cell in question. As we will see later, this approach is at the heart of the parallel implementation of the method.

The system of equations solved as well as the solution strategy follows that presented in many earlier works [34, 22, 21]. The governing equations of the flow may be written as

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = 0 \quad \text{in } D, \tag{6}$$

where it is convenient to denote the Cartesian coordinates and velocity components by x_1, x_2, x_3 and u_1, u_2, u_3, and w and f_i are defined as

$$w = \left\{ \begin{array}{c} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{array} \right\}, \quad f_i = \left\{ \begin{array}{c} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i H \end{array} \right\} \tag{7}$$

with δ_{ij} being the Kronecker delta function. Note that this definition of the flux functions f_i corresponds to the Euler equations. They can be expanded to include the appropriate viscous terms in the Reynolds Averaged Navier–Stokes equations without modification to this discussion. Details of the construction of these viscous fluxes are presented in the following section. Also,

$$p = (\gamma - 1)\rho \left\{ E - \frac{1}{2}\left(u_i^2\right) \right\}, \tag{8}$$

and

$$\rho H = \rho E + p \tag{9}$$

where γ is the ratio of the specific heats. Consider a transformation to coordinates ξ_1, ξ_2, ξ_3 where

$$K_{ij} = \left[\frac{\partial x_i}{\partial \xi_j}\right], \quad J = \det(K), \quad K_{ij}^{-1} = \left[\frac{\partial \xi_i}{\partial x_j}\right].$$

Introduce scaled contravariant velocity components as

$$U_i = Q_{ij} u_j$$

where

$$Q = JK^{-1}.$$

The Euler equations can now be written as

$$\frac{\partial W}{\partial t} + \frac{\partial F_i}{\partial \xi_i} = 0 \quad \text{in } D, \tag{10}$$

with

$$W = J \left\{ \begin{array}{c} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{array} \right\}, \quad F_i = Q_{ij} f_j = \left\{ \begin{array}{c} \rho U_i \\ \rho U_i u_1 + Q_{i1} p \\ \rho U_i u_2 + Q_{i2} p \\ \rho U_i u_3 + Q_{i3} p \\ \rho U_i H \end{array} \right\}. \tag{11}$$

For the multiblock flow solver, the above notation applies to each block in turn. The flow is thus determined as the steady-state solution to equation (10) in all blocks, subject to the flow tangency or no-slip conditions on solid boundary faces:

$$U_\eta = 0 \quad \text{on all } B_S \tag{12}$$

for flow tangency, or

$$U_{\xi_i} = 0, \ i = 1, 2, 3 \quad \text{on all } B_S \tag{13}$$

for no-slip boundary conditions. In this notation, η is 1, 2, or 3 depending on the direction that is normal to face B_S where a solid surface is indicated. At the far field boundary faces, B_F, freestream conditions are specified for incoming waves, while outgoing waves are determined by the solution.

The time integration scheme follows that used in the single block solver [34]. The solution proceeds by performing the cell flux balance, updating the flow variables, and smoothing the residuals at each stage of the time-stepping scheme and at each level of the multigrid cycle. The main difference in the integration strategy is the need to loop over all blocks during each stage of the process. The use of the double-halo configuration permits standard single-block subroutines to be used, without modification, for the computation of the flow field within each individual block. This includes the single-block subroutines for convective and dissipative flux discretization, viscous discretization, multistage time stepping, and multigrid convergence acceleration.

The only difference between the integration strategies is in the implementation of the residual averaging technique. In the single-block solution strategy, tridiagonal systems of equations are set up and solved using flow information from the entire grid. Thus, each residual is replaced by a weighted average of itself and the residuals of its neighbors in the entire grid. In the multiblock strategy, the support for the residual smoothing is reduced to the extent of each block, in order to eliminate the need to solve scalar tridiagonal systems spanning the blocks, which would incur a penalty in communication costs. Depending on the topology of the overall mesh, the setup of tridiagonal systems that follow coordinate lines may lose the physical interpretation that it had in the single block implementation. This change has no effect on the final converged solution, and in all applications of the solver has not led to any reduction in the rate of convergence.

Viscous discretization

In order to include the viscous terms of the Navier-Stokes equations into the spatial discretization scheme it is necessary to approximate the velocity derivatives $\frac{\partial u_i}{\partial x_j}$ which constitute the stress tensor σ_{ij}. These derivatives may be evaluated by applying the Gauss formula to a control volume V with boundary S:

$$\int_V \frac{\partial u_i}{\partial x_j} dV = \int_S u_i n_j dS \ ,$$

where n_j is the outward normal. For a hexahedral cell this gives

$$\overline{\frac{\partial u_i}{\partial x_j}} = \frac{1}{\mathrm{V}} \sum_{\text{faces}} \overline{u}_i \, n_j \, S \ , \qquad (14)$$

where $\overline{u}_i$ is an estimate of the average of u_i over the face, n_j is the jth component of the normal, and S is the face area.

In a cell centered scheme, the integration is carried out on a dual mesh obtained by connecting the centers of the computational cells in the original mesh. This process yields an approximation of the stress tensor at the vertices of the original computational mesh. Once the stress tensor is computed at the cell vertices, it is averaged at the face centers before computing the viscous flux balance. This discretization is very efficient because it does not require the evaluation of the gradients separately for each cell face. However, as a consequence of the averaging process, the discretization may admit odd/even decoupling modes. These modes should be damped by the third-order artificial dissipation already added to damp the odd/even modes arising from the central difference approximation of the convective terms. Alternatively, it is possible to add a correction stencil to the velocity gradients calculated at the vertices to approximately convert it to a more compact stencil characteristic of a face-centered approach [31] without increasing the computational cost. The first approached described is used here.

The implementation of this discretization procedure for the viscous terms in the multiblock method requires only a single halo for the both the flow values and the grid locations.

MULTIBLOCK DESIGN STRATEGY

With the discussion of the multiblock flow solver completed, we will now describe the adjoint based design methodology. The development and implementation of adjoint approaches for aerodynamic shape optimization has reached a stage of maturity in which problems of practical interest are starting to be considered. In one of our recent publications, both transonic and supersonic shape optimizations were performed for complex aircraft configurations subject to a variety of geometric constraints [49]. Although the inviscid Euler equations were used as the core CFD algorithm for these design calculations, an accompanying paper at the same conference presented the derivation of a viscous adjoint algorithm and demonstrated a preliminary wing-body design capability using a viscous flow solver coupled with an inviscid adjoint solver [32]. The methodology presented here will follow this latter approach to allow for the capability of complete aircraft shape design in the presence of viscous effects.

The course of action can be described as follows: first, the structured multiblock Navier-Stokes flow solver described in the previous section replaces the inviscid multiblock method used in reference [49]. Therefore, although the expression for the cost function does not include viscosity related items, it reflects the effects of the presence of the boundary layer. The ability to perform inviscid design calculations is retained with a simple input flag. The gradient of the cost function with respect to geometric design variables is then calculated via the solution of the inviscid adjoint system of [49].

The lack of a viscous adjoint solver corresponding to the viscous flow solver has various important implications. First and foremost is the fact that the use of an Euler adjoint is mathematically inconsistent with a cost function evaluated via a set of viscous governing equations. Therefore, it is impossible to obtain gradients from this approach that match those obtained using finite differencing. However, for problems of engineering interest, the objective is not necessarily to find the true optimum at all cost, but to get within a reasonable vicinity of the minimum for a cost that is acceptable.

It is interesting to note that other design approaches also suffer from a similar inconsistency for reasons of engineering interest. Quasi-inverse design methods such as those used by Campbell [10, 11] assume a relationship between the pressure distribution and the local surface curvature. This relationship effectively provides an inconsistent gradient in order to obtain improved designs. In reference [11] the idea has been pursued in applications using the Navier–Stokes equations. These approaches, which may be applicable for a small sub-class of problems, are likely to fail in situations where the heuristic assumptions used to obtain gradient information cease to be valid.

With these ideas in mind, it is important to consider both the advantages and the limitations of the present design technique. If the aerodynamic figure of merit to be minimized has a direct dependence on viscosity such as through the friction drag, the approach is rendered invalid since the inviscid adjoint system lacks direct sensitivity to viscosity. However, for problems in which viscosity plays an indirect role the proposed design technique is bound to produce useful results. Some important aerodynamic shape optimization problems fall into this latter category. Take for example the problem of pressure drag minimization for commercial transport aircraft. Without breakthroughs in either laminar flow control or turbulent skin friction reduction technologies, most of the aerodynamic performance improvements attainable for a given configuration can be achieved through pressure drag minimization (both induced drag and wave drag). In addition, since the pressure gradient normal to a viscous boundary layer for aircraft at cruise conditions is negligible, the pursuit of inviscid methods for aerodynamic shape optimization has yielded moderate success [15, 50, 49].

However, inviscid design methods must be used cautiously even for inverse pressure distribution or pressure drag minimization problems, since the viscous effects will indirectly alter these quantities. The most noticeable effect is due to the boundary layer displacement thickness. The magnitude and importance of the effective changes in wing shape caused by the presence of the boundary layer depend on the flow field in question, and generally become more pronounced under transonic conditions. The position and strength of shock waves as well as the level of pressure recovery at the trailing edge can be strongly impacted by the existence of a boundary layer. In transonic flow, it is thus highly desirable to take viscous effects into account when designing the aerodynamic shape of a wing to minimize pressure drag.

When the effect of the boundary layer on the outer flow couples very strongly, as is the case at transonic buffet or at maximum lift coefficient conditions, the ability to perform meaningful design without a viscous adjoint can be questioned.

In summary, a design methodology has been developed that uses the Navier–Stokes equations for the flow solution and an inviscid adjoint formulation to obtain gradient information. This method is suitable for a large class of problems of practical aerodynamic interest. For problems in which the viscous effects dominate the behavior of the flow, the viscous formulation of the adjoint equations more than likely will be necessary. It is our intention to pursue this issue further in the coming months.

Adjoint Solver

The mathematical development of the inviscid adjoint equations used in this research has been extensively discussed in our earlier work [23, 24, 25, 29, 45, 33, 46, 47, 48, 50]. An introductory treatment of the derivation of a viscous adjoint has been given in reference [32]. In this section we present a short review of the development of the inviscid adjoint equations for the illustrative problem of pressure drag minimization subject to a variety of constraints.

$$\begin{aligned} I = & \quad C_D \\ = & \quad C_A \cos\alpha + C_N \sin\alpha \\ = & \quad \frac{1}{\mathbf{S_{ref}}} \iint_{B_S} C_P \left(S_x \cos\alpha + S_y \sin\alpha\right) d\xi_1 d\xi_2, \end{aligned}$$

where S_x and S_y define projected surface areas, $\mathbf{S_{ref}}$ is the reference area, and $d\xi_1$ and $d\xi_2$ are the two coordinate indices that are in the plane of the face in question. Note that the integral in the final expression above is carried out over all solid boundary faces. The design problem is now treated as a control problem where the control function is the geometry shape, which is chosen to minimize I, subject to the constraints defined by the flow equations. A variation in the shape will cause a variation δp in the pressure and consequently a variation in the cost function

$$\delta I = \tilde{\delta} C_D + \frac{\partial C_D}{\partial \alpha}\delta\alpha$$

where $\tilde{\delta} C_D$ is the variation due to changes in the design parameters with α fixed. To treat the problem of practical design, drag must be minimized at a fixed lift coefficient. Thus an additional constraint is given by

$$\delta C_L = 0,$$

which yields

$$\tilde{\delta} C_L + \frac{\partial C_L}{\partial \alpha}\delta\alpha = 0.$$

Combining these two expressions to eliminate $\delta\alpha$ gives

$$\delta I = \tilde{\delta} C_D - \frac{\left(\frac{\partial C_D}{\partial \alpha}\right)}{\left(\frac{\partial C_L}{\partial \alpha}\right)} \tilde{\delta} C_L. \qquad (15)$$

Since p depends on w through the equation of state, the variation δp can be determined from the variation δw. If a fixed computational domain is used, the variations in the shape result in variations in the mapping derivatives. Define the Jacobian matrices

$$A_i = \frac{\partial f_i}{\partial w}, \quad C_i = Q_{ij} A_j. \qquad (16)$$

Then the equation for δw in the steady state becomes

$$\frac{\partial}{\partial \xi_i}(\delta F_i) = 0, \tag{17}$$

where in the domain

$$\delta F_i = C_i \delta w + \delta(Q_{ij}) f_j,$$

and on the solid surface,

$$\delta F_\eta = \left\{ \begin{array}{c} 0 \\ Q_{\eta 1}\delta p \\ Q_{\eta 2}\delta p \\ Q_{\eta 3}\delta p \\ 0 \end{array} \right\} + p \left\{ \begin{array}{c} 0 \\ \delta(Q_{\eta 1}) \\ \delta(Q_{\eta 2}) \\ \delta(Q_{\eta 3}) \\ 0 \end{array} \right\} \text{ on any } B_S. \tag{18}$$

Now, multiplying equation (17) by a vector co-state variable ψ, assuming the result is differentiable, and integrating by parts over the entire domain,

$$\int_D \left(\frac{\partial \psi^T}{\partial \xi_i} \delta F_i \right) d\xi_j - \int_B \left(\bar{n}_i \psi^T \delta F_i \right) d\xi_j = 0, \tag{19}$$

where $\bar{n}_i$ are components of a unit vector normal to the boundary. Equation (19) can now be subtracted from equation (15) without changing the value of δI. Then ψ may be chosen to cancel the explicit terms in δw and δp. For this purpose ψ is set to the steady-state solution of the adjoint equation

$$\frac{\partial \psi}{\partial t} - C_i^T \frac{\partial \psi}{\partial \xi_i} = 0 \quad \text{in } D, \tag{20}$$

with the surface boundary condition

$$(\psi_2 Q_{\eta 1} + \psi_3 Q_{\eta 2} + \psi_4 Q_{\eta 3}) = \mathcal{Q} \quad \text{on all } B_S, \tag{21}$$

where

$$\mathcal{Q} = \frac{1}{\frac{1}{2}\gamma M_\infty^2 \mathbf{S_{ref}}} \{(S_x \cos\alpha + S_y \sin\alpha) + \Omega(S_y \cos\alpha - S_x \sin\alpha)\}.$$

At internal block boundaries, the face integrals cancel from the contributions of the adjacent blocks. At the far field the choice of the adjoint boundary conditions depends on whether the flow is subsonic or supersonic. For subsonic flow, so long as the outer domain is very far from the configuration of interest, we may set

$$\psi_{1-5} = 0 \quad \text{on all } B_F.$$

It is noted that the waves in the adjoint problem propagate in the opposite direction to those in the flow problem because of the transpose in equation (20).

Finally we obtain the expression

$$\delta I = \frac{1}{\mathbf{S_{ref}}} \iint_{B_S} C_p \{(\delta S_x \cos\alpha + \delta S_y \sin\alpha) + \Omega(\delta S_y \cos\alpha - \delta S_x \sin\alpha)\} d\xi_1 d\xi_2 + \int_D \psi^T \frac{\partial}{\partial \xi_i}(\delta Q_{ij} f_j) d\xi_k. \tag{22}$$

In order to evaluate the changes in the cost from the above expression, the function ψ must be defined through the solution of (20). A major difference between the development of adjoint solvers presented by our group and those cited in references [4, 5, 6, 8, 7, 13, 14, 9, 38, 36, 20, 41, 37, 35] is that we have relied on a continuous formulation. In this formulation the adjoint system of equations is derived starting from the continuous governing equations to produce a set of continuous co-state equations including boundary conditions. This set of co-state equations is then discretized for computational analysis as a final step. A discrete formulation interchanges the order of these operations by starting from the discrete governing equations and employing linear algebra to obtain a discrete adjoint system. It is useful to note that the final results from these two approaches can be explained as alternate discretizations of the continuous adjoint formulation.

This subtle difference in the order of the adjoint and discretization operations has several important implications. Some of these differences have been explored in detail in the first author's Ph.D. dissertation [52] as well as in the recent work by Anderson and Venkatakrishnan [3]. For purposes of the present work, it is important to focus on one particular difference between the continuous and discrete adjoints. If a discrete approach is followed, gradients obtained via either the resulting adjoint or directly through finite differences should converge to the same result. Hence, the very idea of using an inviscid adjoint for a viscous state equation would not exist. The combination of methods used in this paper is derived from the natural flexibility of employing a continuous adjoint formulation.

Details of the particular discretization used here are covered in reference [52]. The discrete adjoint system is solved in precisely the same manner as the flow equations. More details of the approach as well as the development for other cost functions have been presented in references [25, 29, 33, 46, 47, 50, 43, 49].

Design Variables and Underlying Geometry Database

Even with the rapid developments of the last few years regarding the derivation and implementation of adjoint solvers, many unresolved issues require further research efforts. Not the least of these remaining difficulties is the precise description of the machinery used to modify the shape of interest. This choice directly affects other aspects of the design algorithm. Observing that equation (22) requires not only the flow and the adjoint solutions, but also variations in the mesh metrics, we see the importance of choosing the design variable formulation. In order to obtain all the discrete gradient components from equation (22) it will be necessary either to develop an analytic expression for the variation in mesh metrics or to calculate them directly. The availability of this choice will be determined by the choice of the design variable formulation.

Available choices for the design variables span a wide spectrum ranging from employing the locations of the actual mesh points, to relying on the analytic control points used in a CAD definition of the geometry.

In the case of using the actual mesh points, no underlying geometry database exists. Constraints, if present, must be imposed directly on the locations of these mesh points. This approach will surely prove problematic in general. Consider, for example the difficulties involved in the imposition of a wing fuel volume constraint. In addition, the treatment of surface intersections (such as the wing-body) raises difficulties for this approach since the path for the motion of the mesh points lying directly on these intersections is ill-defined.

However, an advantage of using the mesh points as design variables is that, when combined with an analytic mesh mapping transformation, the calculation of the gradient can be performed without explicitly computing the variations in the mesh metrics. Unfortunately, obtaining such a general mapping transformation increases in difficulty with added geometric complexity.

The alternative of using an underlying geometry database, which may be modified either by the direct application of design variables or by changes in the coefficients of its possibly analytic definition, also has its advantages. First, since the raw unintersected geometries are available, constraints and design changes affecting intersections are easily treated. This can be done without regard to the actual mesh that is used for the flow and adjoint calculations. However, these strengths are counterbalanced by the fact that additional computational work is required to calculate the mesh metric variations.

In the current research, we have used an underlying geometry database where a set of simple geometric entities, such as wings and bodies, are input to the design algorithm in addition to the multiblock mesh used for the calculations. Design variables which are defined as a set of analytic shape functions are applied directly to these geometric entities. Linear and nonlinear geometric constraints are then evaluated on these primary entities. At any particular point in the design process, changes to the mesh surfaces are obtained by first intersecting all of the geometric entities to construct a set of parametric surfaces representing the complete configuration. The location of each surface mesh point on this parametric representation of the geometry is determined for the initial configuration in a pre-processing step. Thus, the results of this pre-processed mapping from parametric geometry to the computational surface mesh points is also a part of the necessary input. The perturbed surface mesh point locations are determined by evaluating the parametric geometry surfaces at these predetermined locations. Once the surface mesh points have been updated, the volume mesh may be perturbed (see following section on mesh motion) and either the gradient or the solution can be calculated. The important feature of this approach is that a set of simple geometric entities lies at the core of the entire design process. This technique retains the typical way in which aerodynamic vehicles are defined, and provides strict control over how surface intersections are treated. Furthermore, since the chosen design variables act directly on the geometric entities, at the end of the design process these entities may be output for future analysis.

In the current implementation, input geometric entities are restricted to those defined by sets of points. However, in the future, CAD entities such as NURBS surfaces will also serve this role, thereby allowing both the input and the output from the aerodynamic surface optimization method to interface directly with a CAD database.

Mesh Perturbation Algorithm

After we have applied a set of design variables to the underlying geometry and mapped these changes to changes in the computational surface mesh points, two related tasks remain. For gradient calculations, variations in the mesh metrics must be calculated. In addition, when a design step is to be taken it must be possible to deform the entire mesh to accommodate design changes. Both tasks are accomplished in this work by the approach presented in references [49] and are only outlined here.

Since it would be difficult in the current application to obtain an explicit relationship between arbitrary surface changes and variations in the multiblock mesh metrics, these latter quantities are calculated by finite differences. This approach avoids the use of multiple flow solutions to determine the gradient, but it unfortunately still requires the mesh to be regenerated repeatedly. The number of mesh generations required is proportional to the number of design variables. The inherent difficulty in the approach is two-fold. First, for complicated three-dimensional configurations, elliptic or hyperbolic partial differential equations must normally be solved iteratively in order to obtain acceptably smooth meshes. These iterative mesh generation procedures are usually computationally expensive. In the worst case they approach the cost of the flow solution process. Thus the use of finite difference methods for obtaining metric variations in combination with an iterative mesh generator leads to computational costs which strongly hinge on the number of design variables, despite the use of an adjoint solver to eliminate the flow variable variations. Second, multiblock mesh generation is by no means a trivial task. In fact no method currently exists that allows this to be accomplished as a completely automatic process for complex three-dimensional configurations.

Here, these difficulties are overcome through the use of a mesh perturbation technique. In this approach, a high quality mesh appropriate for the flow solver is first generated by any available procedure prior to the start of the design. In examples to be shown later, these meshes were created using the Gridgen software developed by Pointwise, Inc.[53]. This initial mesh becomes the basis for all subsequent meshes which are obtained by analytic perturbations.

In order to perturb the multiblock mesh, two capabilities are required. First, the block corners, edges and faces must be moved in a manner that follows the desired geometric changes and simultaneously retains mesh continuity throughout the domain. The second requirement is to move all the points interior to each block such that the spacing distributions and smoothness of the original mesh are retained. This latter requirement is accomplished by the WARP3D algorithm [43]. Since our current flow solver and design algorithm assume a point-to-point match between blocks, each block may be independently perturbed by WARP3D, provided that perturbed surfaces are treated continuously across block boundaries. The methodology used to achieve the first requirement of maintaining continuity in the blocking structure is given as follows:

1. All faces that are directly affected by the design variables (active faces) are explicitly perturbed.

2. All edges that touch an active face, either in the same block or in an adjacent block, are implicitly perturbed by a simple arc-length-based algorithm.

3. All inactive faces that either include an implicitly perturbed edge or abut to an active face are implicitly perturbed by a quasi-3D form of WARP3D.

4. WARP3D is used on each block that has one or more explicitly or implicitly perturbed faces to determine the adjusted interior points.

Note that much of the mesh, especially away from the surfaces, will not require mesh perturbations and thus may remain fixed through the entire design process. Close to the surfaces, many blocks will either contain an active face or touch a block which contains an active face, either by an edge or by a corner. As the design variations affect the active faces, the above scheme ensures that the entire mesh will remain attached along block boundaries. Added complexity is needed to accomplish step (2) since the connectivity of the various edges and corners must be indicated somehow. Currently, pointers to and from a set of master edges and master corners are determined as a pre-processing step. During the design calculation, perturbations to any edges or corners are fed to these master edges and master corners which in turn communicate these changes to all connected edges and corners.

Since this mesh perturbation algorithm is explicit it is possible to work out the analytic variations in the metric terms required for equation (22). This approach was followed in [46]. However since the mesh perturbation algorithm that is used in the current paper was significantly more complex, and it was discovered that the computational cost of repeatedly using the block perturbation algorithm was within reason, finite differences were used to calculate δQ_{ij} instead of deriving the exact analytical relationships.

Optimization Algorithm and Problem Constraints

With all of the machinery to obtain gradients for an arbitrary set of design variables in place, it remains as a final detail to outline the numerical optimization algorithm and the imposition of constraints. The NPSOL optimization algorithm employed here was chosen because of its extensive past use on aerodynamic optimization problems, and its treatment of both linear and nonlinear inequality constraints. NPSOL [16] is a sequential quadratic programming (SQP) method in which the search direction is calculated by solving the quadratic subproblem where the Hessian is defined by a quasi-Newton approximation of an augmented Lagrangian merit function. The

Lagrange multipliers in this merit function serve to scale the effect of any nonlinear constraints that the design may contain. Linear constraints are treated by solving the quadratic subproblem such that the search direction remains in feasible space. A complete treatment of the method and other optimization strategies is given in [17].

The entire design procedure is outlined below:

1. Decompose the multiblock mesh into an appropriate number of processors, and create lists of pointers for the communication of the processor halo cells.
2. Solve the flow field governing equations (6-11) for each design point.
3. Solve the adjoint equations (20) subject to the boundary condition (21) for each design point.
4. For each of the n design variables repeat the following:
    - Perturb the design variable by a finite step to modify the geometric entities.
    - Reintersect the geometric entities and form parametric geometry surfaces.
    - Explicitly perturb all face mesh points affected by the geometry changes by evaluating their locations on the parametric geometries.
    - Implicitly perturb all faces that share an edge with an explicitly perturbed face.
    - Obtain the perturbed internal mesh point locations via WARP3D for those blocks with perturbed faces.
    - Calculate all the delta metric terms, $\delta Q_{i,j}$, within those blocks that were perturbed using finite differencing.
    - Integrate equation (22) to obtain δI for those blocks that contain nonzero $\delta Q_{i,j}$, and for each design variable, to determine the gradient component.
5. Calculate the search direction and perform a line search via NPSOL.
6. Return to (2) if a minimum has not been reached.

PARALLELIZATION STRATEGY

Communication Management

Efficient parallel computation on a set of distributed processors is achieved by a combination of minimizing the overhead of communication between these processors and balancing the partitioned workload among them. The first obvious choice in order to improve parallel performance is then to minimize the amount of communication required between processors. This includes minimizing the number of messages sent and received (latency) as well as the total amount of data to be transferred (bandwidth). Further improvement may be achieved by using other techniques such as latency hiding and scheduled communication, if either the problem or the specific architecture of the distributed platform allow it.

Latency hiding is in itself a form of local parallelism, where the communication and computations of an individual node proceed concurrently with each other. This benefit is accomplished by initiating the bi-directional asynchronous communication as soon as the data to be passed has been updated and in such a manner that calculations that can be done (without updated data from remote nodes) are performed while the communication continues in the background.

In the case of networks which utilize a collision-detection based protocol (e.g., ethernet), scheduled information transfer may help reduce the communication overhead. A consequence of the collision-detection mechanism is that the effective bandwidth of a saturated network is degraded since a portion of it is wasted when two or more processors are trying to initiate communication simultaneously. Hence, by scheduling or synchronizing the messages between processors, one can minimize this deterioration in performance. On the other hand, the current generation of parallel computers typically include a networking environment which is capable of sustaining simultaneous communication among all the processors in the machine. This enhanced communication ability comes with a high price tag. An intermediate solution where hardware switching is embedded in a workstation distributed computing environment will be shown to be a compromise between these two options.

In this paper, we revisit the task of minimizing the amount of total communication required between processors. However, we avoid the temptation to communicate less often than is consistent with the baseline serial calculation so that the convergence of the original scheme is exactly maintained. The possibility of improving the parallel performance of the method by further restricting the amount of communication at different points in the multigrid sequence will be investigated at a later time.

The following subsections describe the baseline method against which all improvements are measured, and the modifications (associated with communication overheads) developed under the present work.

Baseline Communication

The multiblock solver is parallelized by using a domain decomposition model, a SPMD (Single Program Multiple Data) strategy, and the MPI Library for message passing.

The baseline parallel scheme is exactly consistent with the serial multiblock solution: the results produced by both programs are identical, including the convergence history of the method. Updates to the solution vector in all processors occur at every stage of the Runge-Kutta time-stepping scheme and in every level of a multigrid W-cycle. In addition, the baseline computations are performed with 64-bit arithmetic. Therefore, flow residuals in the calculation can be converged approximately 13 orders-of-magnitude before roundoff effects stall the convergence process.

The multiblock strategy adopted in this work allows the independent update of the internal cells of every block in the mesh by using a halo or ghost cell approach. The information in this halo of cells surrounding each block is transferred from the corresponding physical cells in the interior of the neighboring blocks. The baseline scheme utilizes a three-pass communication model which allows for the computation of solutions on arbitrarily oriented multiblock meshes.

Under this model, updated halo information is transferred across the six faces of each block during each phase of the three-pass communication. The first pass transfers face information, the second pass provides edge data, and the final pass is required to update the solution across the block corners. With this three-pass approach, each block is guaranteed to have the proper information in its complete halo (including edges and corners) regardless of the topology of the mesh. This is a particularly challenging situation when more than four blocks meet at a given edge. The double halo is used to compute the third-order artificial dissipation terms while preserving a fully conservative scheme. Although the current discretization of the viscous fluxes requires only a single level halo, future variations which require the presence of a double halo can be accommodated with this procedure.

In addition to the above, the blocks of the baseline solution are distributed to the individual processors in such a manner that the total number of unknowns per processor is as evenly loaded as possible. While finding the optimum distribution is recognized to be an NP-Complete problem, a simple algorithm is employed which routinely yields a load balancing in the neighborhood of the optimum. The essence of this algorithm is to take the largest of the remaining blocks (yet to be distributed) and assign it to the processor with the smallest current load. Repeating this procedure until all blocks have been distributed, an effective load balance algorithm is obtained. Table 1 illustrates the effectiveness of this algorithm for our benchmarking test case.

Processor	Ncells	Cell-Ratio
1	185,088	1.007
2	185,088	1.007
3	182,400	0.993
4	182,400	0.993

Table 1: Baseline Load-Balance for the Benchmark Test Case on 4-Processors.

The test case of Table 1 is used to benchmark the effectiveness of the enhancements described below. It is a 72-block grid about a wing-fuselage-nacelle geometry. The total number of cells in the system is 734,976. Of these, roughly 300,000 are halo cells. This case was chosen specifically because it accentuates the penalty of communication. Yet, on a high-speed, low-latency network such as those on the IBM SP2 or the SGI Origin2000, the corresponding flow solutions scale reasonably with the number of processors.

One-Pass Communication Model

The use of the original three-pass communication model was necessary for handling a completely general block structure. Drawbacks of this approach are that redundant communications are performed and that the second and third passes must wait until the previous passes have completed before they are started.

The source of redundant data passing can be seen by following the flow of information from one block to a neighboring block coincident with an edge or a corner of the originating block. For example, across an edge, information from one block to another (located above and to the right) can flow in one of two ways. Firstly, the data could flow from the originating block to its right-hand neighbor, then this information could be transferred from this neighbor to the block directly above it. Alternatively, the data could first move upward, then to the right. Because of the complexity involved in determining which path the data should flow along and which it should not, the baseline three-pass model transfers the information in both directions. Similarly, for communications across corners, this redundancy is three-fold.

Hence, an obvious source of improvement is to remove redundant data transfers from the communications model. This is accomplished by adopting

Processor	1	2	3	4	*MPI*
1	22,848	30,080	27,040	12,960	70,080
2	30,080	23,296	5,376	20,224	55,680
3	27,040	5,376	21,248	31,648	64,064
4	12,960	20,224	31,648	12,992	64,832
MPI	70,080	55,680	64,064	64,832	254,656

Table 2: Three-Pass Communication Matrix using the Baseline Load-Balance.

Processor	1	2	3	4	*MPI*
1	17,504	26,480	21,128	13,744	61,352
2	26,496	17,696	8,264	16,264	51,024
3	21,232	8,256	16,792	27,864	57,352
4	13,776	16,232	27,768	10,136	57,776
MPI	61,504	50,968	57,160	57,872	227,504

Table 3: One-Pass Communication Matrix using the Baseline Load-Balance.

a single-pass scheme which reproduces exactly the end state of the original three-pass model. In order to ensure that the the one-pass communication model produces results identical to the three-pass approach, the original three-pass model is used to initialize the communication lists of the one-pass method. This is accomplished in the following manner: after the blocks of the grid system have been assigned to an appropriate processor through the load balancing procedure, the solution vector is colored with information that describes its starting location prior to any communication. This encoding includes information such as block and processor numbers and local cell indices. With this state set, the solution vector is processed with the original three-pass communication model. Upon completion of this data transfer, every halo cell in the distributed system has been reset with information which points back to its origin, i.e. block number, processor number and "distant" cell index. At this stage, new communication lists are constructed and returned to the source processor which stores them for future use by the one-pass model.

Tables 2 and 3 illustrate the reduction in communication achieved for the one-pass model. These tables provide the communication matrix (for the finest mesh in the multigrid sequence) of message sizes for each communication approach. The diagonal terms of these matrices correspond to messages that a processor needs to send to itself. For this kind of message, the present method uses a local memory copy instead of an actual MPI (Message-Passing Interface) message, which is used for inter-processor communication. For the benchmark test case, the one-pass model reduces the total message length by about 11% on the fine mesh. However, because there is no forced synchronization between passes as in the three-pass model, the overhead reduction approaches 25%.

Delta Updates

In the baseline code, communication always transferred the actual values of the solution vector. In order to preserve 64-bit accuracy, all of these values were transferred as 64-bit floating-point numbers. In the present one-pass model, an additional choice of communication model has been implemented. We refer to this communication model as the *delta update* procedure.

The purpose of including a delta form in the present work is motivated by the fact that these delta increments can be transferred as 32-bit numbers while maintaining 64-bit accuracy in the converged solution. Naturally, maintaining this level of precision during the course of convergence requires an occasional reset of the halo values with a 64-bit communication, although the large majority of the communication is now performed using only single precision 32-bit numbers. For this occasional reset, we have maintained the capability to transfer actual full precision values of the solution vector.

For all practical purposes, the communication overhead of the new delta form is half that of the baseline (full precision) transfers.

Communication-Weighted Load Balancing

As mentioned above, the original load-balancing algorithm was guided solely by the number of cells being distributed to the complete set of processors. This form of load balancing has proved to be quite acceptable for platforms with state-of-the-art communication capabilities such as the IBM SP2. However, for a cluster of workstations linked together with a lower performance network, this technique can be further refined.

A new load-balancing algorithm has been developed which includes the penalties associated with out-of-processor communication. In this setting, the load is defined as the time it takes each processor to complete all of its tasks–numerical processing as well as sending and receiving the necessary messages. The predicted times of each of these tasks are derived using experimentally obtained MFLOPS (Millions of Floating Point Operations per Second) ratings, and the MPI latency and bandwidth values associated with the particular distributed platform.

The new load-balancing algorithm is very similar to that of the original method, but the "size" of each block is now initialized assuming that the information of all halo cells will be transferred to another

PLATFORM	MFLOPS	LATENCY (μ-sec)	BANDWIDTH (MBytes/s)
SP2 Switch US	50	43	35.0
SP2 Switch IP	50	285	13.0
HP/J280-100BaseT	30	290	7.0
HP/J280-ethernet	30	600	0.8

Table 4: Observed Capacities of Various Platforms.

processor. The algorithm then proceeds by taking the largest of the remaining blocks (yet to be distributed) and temporarily assigning it to every processor. When assigned to each processor, a temporary update of the load of that processor is made by adding the size of the current block to that processor's previous load. This assignment is rewarded by a decrease in the equivalent size if neighboring blocks are already assigned to that processor and thus no communication is necessary. After all temporary assignments have been done, the processor whose load is the smallest after the assignment is selected and the block is permanently assigned to that processor. The previous steps are repeated until all blocks have been distributed.

Table 4 provides representative values for the IBM SP2 using the switch in User Space mode (high performance communication mode), for the switch and IP (Internet Protocol), and for an HP workstation cluster of J280s using switched-100BaseT and standard ethernet. These values have been experimentally observed and may vary from site to site. They correspond to the measured values of latency and bandwidth using various implementations of the MPI standard on the different platforms mentioned above. They are not the manufacturer's published data for the communication hardware. It is interesting to note that the values of latency and bandwidth obtained for the switched-100BaseT network are quite close to those for the IBM SP2 system communicating in IP mode. Therefore, the SP2 can be used to simulate large networks of workstations linked together by a switched-100BaseT network.

Using the network characteristics for the HP cluster on standard ethernet, a new distribution of blocks is obtained. This distribution is provided in Table 5. Comparing it with Table 1, it is noticed that the number of cells per processor is not nearly as well "balanced" as before. Yet, the solution's cycle time of the new distribution on the HP cluster using an ethernet network is only 57.46 seconds as compared with the original cycle time of 108.38 seconds.

The secret of this performance improvement can be seen by comparing the communication matrices of the original load-balanced distribution with that of the improved one. This information is provided by Tables 3 and 6, respectively. Notice that the new

Processor	Ncells	Cell-Ratio
1	169,536	0.923
2	203,136	1.106
3	188,544	1.026
4	173,760	0.946

Table 5: New Load-Balance for the Benchmark Test Case on 4-Processors.

Processor	1	2	3	4	*MPI*
1	28,848	11,888	16,736	4,304	32,928
2	11,968	42,168	2,368	16,328	30,664
3	16,744	2,552	43,840	11,696	30,992
4	4,456	16,576	11,648	35,384	32,680
MPI	33,168	31,016	30,752	32,328	127,264

Table 6: One-Pass Communication Matrix using the New Load-Balance.

load-balancing algorithm has done an effective job of reducing the amount of data to be transferred via MPI. Under the baseline distribution, MPI messages are used to set a total of 227,504 halo cells in the fine mesh. Using the new load-balancing algorithm, MPI calls are only responsible for resetting now 127,264 halo cells. This is accomplished by increasing the amount of data transfer each processor does with itself (i.e., in a global sense, communication is drawn toward the diagonal of these matrices).

Single-Layer Halo Communication

In the baseline method, we stated that a double-layer halo surrounds each block and it is utilized to facilitate calculation of the third-order artificial dissipation fluxes. However, upon close inspection of the 5-stage Runge-Kutta scheme and multigrid processes, we note that the dissipative fluxes are not recomputed as often as the solution updates occur. In particular, these dissipative terms are typically reset only during the odd stages of Runge-Kutta on the finest mesh and never computed in any of the coarser levels of the multigrid scheme.

Immediately, we can omit transferring the outer-layer halo data during the even (of five) stages of Runge-Kutta in the fine mesh. This reduces the fine-mesh communication by 20%.

For a 4-level multigrid W-cycle in the baseline code, more than 45% of the total data transferred during the cycle resides in the coarser-level meshes. [††] By updating only the data of the inner halo during the coarse-level communication, an additional improvement is realized.

[††]A W-cycle with four levels of multigrid traverses the second-level grid exactly twice when communication is involved; four times in the third and fourth levels. The number of halo cells of a coarse-mesh is at least one-fourth that of

The above two improvements combine to reduce the total amount of data transferred per multigrid cycle. Relative to the baseline communication, this reduction in overhead is between 33.3% and 54.7%, depending on the granularity of the mesh involved.

Communication Improvement Summary

For the 72-block mesh in question, the relative improvements in communication overhead with respect to the original scheme can be summarized as follows:

- 20% reduction in overhead with one-pass (benchmark).
- 50% reduction in overhead with delta form (in general).
- 50% reduction in overhead with new load balancer (benchmark, ethernet).
- 33%-55% reduction in overhead with single-halo transfers (in general).
- Communication reduced by a minimum of 75% when combined.

RESULTS

Design Results

The design test cases to be presented here will focus on the wing redesign of a typical transonic business jet. The designs will be carried out independently using the Euler and Navier-Stokes equations. The discussion will conclude with comparisons between the final Euler and Navier-Stokes designs. For the Euler design case, reference [50] gives a treatment of the reliability of the flow solver as well as the ability of the adjoint method to provide accurate gradients very efficiently. With regard to the validity of the Navier-Stokes case, a comparison will be made for the initial configuration using both the inviscid and the viscous equations. The adjoint gradients for the Navier-Stokes test case will not be compared with finite difference calculations for two reasons. First, since the adjoint used to obtain the gradients is not of the viscous type, it is understood and accepted that it will not produce gradients that are consistent with the finite difference approach. Secondly, the computational cost of obtaining finite difference gradients for the Navier-Stokes design on a large three-dimensional test case is prohibitive. In order to obtain accurate finite difference gradients, the flow solution must be converged at least two or three orders more than is necessary for adjoint gradients [52, 39]. Navier-Stokes solvers with their notoriously slow convergence would take an unacceptable number of iterations to achieve such a level of convergence.

the next finer mesh. Hence, the baseline communication in the coarse grids is at least 81% as intense as it is in the finest mesh. Further study of a grid with only one interior cell in the fourth-level mesh shows that the baseline communication in the coarser grids can approach 183% that of the finest mesh.

Flow Solver Comparison

In the design demonstration of the multiblock optimization algorithm to follow, a typical transonic business jet configuration is considered. The same geometry was also studied in [50, 15, 49]. Here the complete configuration including wing, body, nacelle, pylon, vertical tail, and horizontal tail will be used. Prior to the start of the designs, flow analyses were completed using the Euler and Navier-Stokes equations.

Two alternative meshes were constructed that shared the same block topologies and differed only in the normal wall spacings and cell counts. The meshes were both created with a general C-O topology and flow-through nacelles. Both meshes featured 240 blocks, with the Euler mesh having 4.1 million computational cells and the Navier-Stokes mesh having 5.8 million computational cells. The relative ratio between the two is smaller than expected since the Euler mesh was constructed from the Navier-Stokes mesh simply by coarsening in the viscous direction. Furthermore, while complete configurations are being modeled, only the wing is treated as a viscous solid surface. The other components are handled with inviscid boundary conditions. A single flow calculation for the Euler solution using 200 multigrid cycles converges 5 orders of magnitude in 0.8 hours of wall clock time on 32 processors of an IBM SP2 machine. By comparison, a Navier-Stokes analysis uses 300 multigrid cycles to converge 4.7 orders of magnitude and consumes 2.0 hours of wall clock time on 32 processors of an IBM SP2 machine. An Euler and a Navier-Stokes solution are compared for the baseline configuration at the same over-design flight conditions and compared in Figure (1). The C_p distributions depicted in the figure show the usual trend of having the shock strength and location being moved forward for the viscous analysis when compared to the inviscid analysis. The Navier-Stokes calculation was carried out using an all-turbulent boundary layer with a Baldwin-Lomax turbulence model. The wall normal spacing of the first cell was such that at the cruise condition a $y^+ = 1$ would be attained at the half span trailing edge assuming a flat plate turbulent boundary layer. At the cruise condition ($M = 0.80$ and an altitude of 40,000 ft) the Reynolds number is 1.45 million/ft.

The wing sweep for the design is a low 20 degrees. Thus, with the thick airfoil sections featured in the configuration, it represents a challenge to contain wave drag at the moderate Mach numbers of its design point (M = 0.75 - 0.82). Although they are not presented here, correlations of the wing pressure distributions have been obtained with experimental data. The comparisons with tunnel data are excellent except for a 5% difference in the location of the upper surface shock for the inviscid analysis. The Navier-Stokes solutions virtually overlay the wind tunnel data.

Inviscid Transonic Flow Constrained Aircraft Design

The Euler mesh described above was used during an inviscid redesign of the wing in the presence of the complete configuration. The baseline configuration was designed for flight at $M = 0.80$ with a $C_L = 0.30$. In this inviscid design case, a single point constrained design is attempted in which the Mach number and C_L are pushed to 0.82 and 0.35 respectively. The objective is to minimize configuration pressure drag at a fixed lift coefficient by modifying the wing shape. Eighteen Hicks-Henne design variables were chosen for six wing defining sections for a total of 108 design variables. Spar thickness constraints were also enforced at each defining station at $x/c = 0.2$ and $x/c = 0.8$. Maximum thickness was forced to be preserved at $x/c = 0.4$ for all six defining sections. Each section was also constrained to have the thickness preserved at $x/c = 0.95$ to ensure an adequate included angle at the trailing edge. A total of 55 linear geometric constraints were imposed on the configuration. Figure (2) shows overlays of the C_p distributions for the initial and final design after 6 NPSOL design iterations at four stations along the wing. It is seen that the final result has reached a near-shock-free condition over much of the outboard wing panel. The drop in complete configuration pressure drag for this case was 24.6%. Noting that most of this drag reduction came from a decrease in wing wave drag implies that further improvements may be possible through the reshaping of other components. The program took 13.5 hours to complete 6 design cycles on 32 processors of an IBM SP2.

Viscous Transonic Flow Constrained Aircraft Design

In a second design example for this complete business jet configuration, a viscous redesign of the wing is attempted. The design point is again chosen to be Mach = 0.82 with a $C_L = 0.35$. Again the design algorithm, this time with a no-slip boundary condition on the wing and the viscous terms turned on, is run in drag minimization mode. Figure (3) shows an iso-C_p colored representation of the initial design and the final design after 5 NPSOL design iterations. It is clearly seen that the rather low C_p region terminated by a strong shock spanning the entire wing upper surface has been largely eliminated in the final design. Figure (4) shows the initial and final C_p distributions achieved using the same 108 design variables and 55 geometric constraints employed for the inviscid test case. Note that the strong shocks present on both the upper and lower surfaces in the initial configuration have been eliminated. Furthermore, it is apparent that the character of the changes to the pressure distribution follow those that occurred for the Euler based design to some extent. The main difference is that the Navier-Stokes design tends to have a more benign behavior in the pressure distributions. The overall pressure drag for the complete configuration was reduced by 21.5%. Before proceeding to the next section, it should be noted that these business jet design examples are only representative of the potential for automated design, and are not intended to provide designs for actual construction. First, in each case only 5 or 6 NPSOL steps were taken where considerably more could have improved the designs slightly. More importantly, since these are only single point designs, either may suffer unacceptable off-design behavior. In our most recent previous paper [49] we treated the case of inviscid design at multiple design points while here we address the case of viscous design at a single design point. Eventually, both the multipoint and viscous design capabilities must be treated concurrently. The calculation took 28 hours to complete 5 design cycles on 32 processors of a IBM SP2.

Crosscheck

To see the difference between the two design cases explored here the Euler based design was reanalyzed using the Navier-Stokes equations. The mesh for this cross check was created by entering the design variable coefficients produced from the Euler design process into a set of inputs for a Navier–Stokes analysis. The result of this reanalysis is shown in Figure (5). It is seen that the wing designed using the new Navier-Stokes approach employed here has slightly weaker shocks than the one designed via the use of the inviscid approach. This conclusion is also supported by the fact that the drag improvement for the Euler design analyzed with the Navier-Stokes equations has a 20.5% improvement as contrasted with the 21.5% for the Navier-Stokes based design. However, it is seen from these two designs that the Euler strategy did not perform at all poorly and indeed was able to achieve much of the improvement that was possible from a Navier-Stokes based design method. This conclusion must be taken with caution since the sensitivity of the pressure distributions to the presence of the boundary layer can vary widely depending on the configuration. Furthermore, had we used a true viscous adjoint it may have been possible to lower the pressure drag for the configuration even further. Clearly, much further testing of the design approaches developed here is needed. These calculations must be taken as the preliminary steps towards Navier-Stokes based aircraft design.

Parallel Performance of the Method

The following section presents a series of parallel scalability curves for the baseline code with the various added improvements to reduce the communication overhead. The scalability study was conducted for two different meshes whose ratio of computation vs. communication was chosen to be at both ends of the spectrum. The first mesh is the benchmark mesh referred to above. This mesh consists of 72 blocks of varying sizes and has a total of $734,976$ cells of which over $300,000$ reside in the halos. As one can see, the results on this mesh will provide an extreme test of the scalability of the method since a large part of the time will be spent in the communication process. The second mesh is referred to as the *fine* mesh and consists of 48 blocks of different sizes with a total of 2.5 million cells, from which about $600,000$ reside in the halos. This mesh has a much higher ratio of computation to communication, and therefore, the parallel scalability of the code is expected to improve when compared with the benchmark mesh. Nevertheless, the contrast provided between the results in the two meshes is intended to be illustrative of the range of performance that can be expected for meshes of varying sizes.

Scalability studies were conducted on a range of platforms whose performance characteristics are presented in Table 4. These platforms include the very high performance communication network of the IBM SP2 (in User Space mode) and networks of workstations connected via standard 10BaseT ethernet as well as a higher performance (although still low cost) switched-100BaseT fast ethernet network. For the benchmark mesh, the range $1-16$ was selected for the number of processors, whereas for the fine mesh, $4-32$ was selected instead. This latter choice results from the inability to fit this large size calculation in a number of processors smaller than 4. In order to present speedup data for this mesh, the best achievable timing for a one processor calculation was derived from the measurements of computational time (not communication) observed for the 4 processor case. It will be seen from the data that for this mesh the program must scale at worst linearly between 1 and 4 processors and therefore our timing assumption is conservative.

Figures 6-8 present the results obtained for the benchmark mesh using the IBM SP2 in User Space and Internet Protocol modes and the cluster of workstations using switched-100BaseT for both the baseline and the improved load balancing algorithms. Since the SP2 in User Space Mode has a very high performance network, the impact of the use of different load balancing techniques is small. For example, the parallel speedup for the 1-pass, single precision scheme using 16 processors remained virtually unchanged from 11.88 to 11.89. When used in IP mode (a lower performance network that very well simulates the results on the network of workstations linked via switched-100BaseT) the results of improved load balancing schemes start to show up. For the same communication algorithm and the same mesh, the parallel speedup improved from 8.46 to 8.70 using 16 processors. Using 8 processors on the switched-100BaseT network for the same algorithm, the parallel speedup improves from 3.212 to 3.645. These results are even more dramatic for the case of the lowest performance network: unswitched 10BaseT (ethernet). In this case, the use of the improved load balancing scheme decreased the time to complete one multigrid iteration from 108.38 seconds to 57.46 seconds using 4 processors in the network.

A more interesting point addressed by the results in these scalability plots is the increase in performance derived from successive improvements to the communication algorithm. Figures 6a, 7a, and 8a show the progression from the 3-pass and double precision scheme to the 1-pass and double precision, 1-pass and single precision, and 1-pass and single precision with single level halo schemes using the IBM SP2 in both US and IP modes and the network of switched-100BaseT workstations. These results use the original load balancing technique. Similar conclusions can be obtained from Figures 6b, 7b, and 8b which use the improved load balancing algorithm. For the US results, the parallel speedup using 16 processors improved from 10.41 to 13.00. For the IP results, the improvement is larger (as expected from the use of a lower performance communication network): parallel scalability for the same number of processors improved from 7.11 to 10.69. It must be noted that in both cases, there is an upper limit to the parallel scalability achievable by the scheme derived from the impossibility of evenly load balancing a calculation in which the sizes of the different blocks in the mesh vary. For this case, this upper limit on parallel scalability using 16 processors is found to be 14.9, which is very close to the actual achieved value using the IBM SP2 in US mode, considering the highly communicative nature of the benchmark mesh. At the same time, the results for the switched-100BaseT network of workstations using 8 processors show an improvement in parallel scalability from 2.98 to 5.41, pointing out how much more important these successive communication improvements are for networks with lower performance. From these data, it is clearly seen that a high performance message passing implementation is essential to allow the effective use of networks of workstations.

A more clear representation of the improvement in communication performance can be seen in Figure 9a where the parallel scalability results using the IBM SP2 US and IP modes for the benchmark mesh have been overlaid. The lower curve shows the speedup obtained using the original scheme (3-pass, double precision) with the SP2 in US mode, whereas the upper curve shows the speedup of the improved scheme (1-pass, single precision, single level halo) using the SP2 in the lower performance IP communication mode. Since the IP mode is representative of a switched-fast ethernet network of workstations, one can deduce from this graph that the improvements in communication performance introduced in this work make the use of networks of workstations for the design process entirely feasible.

All these scalability studies were repeated for the fine mesh with 48 blocks and 2.5 million cells. The difficulties with parallel scalability exhibited by the previous figures are substantially ameliorated. In the interest of space, only a summary is presented in Figure 9b where the results using up to 32 processors are presented for the IBM SP2 in US mode and the different variations in communication algorithms. It can be seen that the communication improvements shift the scalability curve slightly upwards. This shift is small since for numbers of processors up to 16 the ratio of communication time to processing time is very small. Note that some of the calculations exhibit *superlinear speedup* for some numbers of processors; this effect has been observed previously [12] and is attributed to a better utilization of the processor cache resulting from a decrease in size of the datasets that the processor operates on. The decrease in performance for the 32 processor results is a consequence of the poor load balancing that can be obtained with a mesh that only has 48 blocks.

Although not reported in any of the figures, scalability studies were performed for the baseline unswitched ethernet network of workstations (10BaseT). This communication fabric has the limitation that the total bandwidth of the system remains constant, independent of the number of processors in the calculation. This lack of bandwidth scalability results from the very nature of ethernet communication; the network can be used by only one processor at a time. Table 7 shows the results for the parallel speedups obtained using the baseline algorithm (original load balancing, 3-pass, double precision) and the improved communication (new load balancing, 1-pass, single precision, single level halo). The disappointing performance of this network clearly points out that typical ethernet networks are not suitable for parallel computations such as these ones which place a high demand on the communication layer. Despite this negative result, it is apparent that the new communication algorithms dramatically improved the performance of the method.

No. Processors	Baseline	Optimized
1	1.000	1.000
2	0.768	1.552
4	0.688	2.301
8		1.493

Table 7: Parallel Speedups for the Baseline and Optimized Communication Schemes Using unswitched Ethernet (10BaseT)

CONCLUSIONS

A general aerodynamic shape optimization method has been developed and demonstrated for the case of Euler and Navier–Stokes automatic redesign of a complete aircraft configurations in transonic flow. The design method is implemented on distributed memory systems (including parallel computers and distributed networks of workstations) using the MPI Standard and it achieves excellent scalability in all platforms except for the simple unswitched-Ethernet case (10BaseT). For a full configuration viscous mesh containing 5.8 million cells and 240 blocks, a complete design including a total of 5 design iterations can be completed in 28 hours using 32 processors of an IBM SP2 system. The present method uses the Navier–Stokes equations for the solution of the flow and an inviscid adjoint solver for the calculation of the gradient information. Further research and development work is required to assess the applicability and usefulness of a viscous adjoint technique.

With the present method, the automatic aerodynamic design of complex aircraft configurations using a high fidelity viscous flow model based on the RANS equations becomes feasible on the current generation of parallel computers. Moreover, the applicability of the method is demonstrated for use in networks of workstations with a moderate investment in networking resources (switched-100BaseT).

ACKNOWLEDGEMENTS

Our research has benefited greatly from the support of ARPA under Grant Number N00014-92-J-1796, AFOSR under Grant Number AFOSR-F49620-95-1-0259, and USRA through RIACS at the NASA Ames Research Center. The support provided by the IBM-CRA grant is gratefully acknowledged by the authors. The Douglas Aircraft Company is also acknowledged in part for supporting the implementation of the design method on a distributed networks of workstations. Special thanks are extended to Mark Rimlinger for the generation of the initial meshes used in the calculations presented here.

REFERENCES

[1] J. J. Alonso, L. Martinelli, and A. Jameson. Multigrid unsteady Navier-Stokes calculations with aeroelastic applications. *AIAA paper 95-0048*, AIAA 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1995.

[2] J. J. Alonso, T. J. Mitty, L. Martinelli, and A. Jameson. A two-dimensional multigrid Navier-Stokes solver for multiprocessor architectures. In Satofuka, Periaux, and Ecer, editors, *Parallel Computational Fluid Dynamics: New Algorithms and Applications*, pages 435–442, Kyoto, Japan, May 1994. Elsevier Science B. V. Parallel CFD '94 Conference.

[3] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *AIAA paper 97-0643*, 35th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1997.

[4] O. Baysal and M. E. Eleshaky. Aerodynamic sensitivity analysis methods for the compressible Euler equations. *Journal of Fluids Engineering*, 113(4):681–688, 1991.

[5] O. Baysal and M. E. Eleshaky. Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA Journal*, 30(3):718–725, 1992.

[6] O. Baysal and M. E. Eleshaky. Airfoil shape optimization using sensitivity analysis on viscous flow equations. *Journal of Fluids Engineering*, 115:75–84, 1993.

[7] G. W. Burgreen and O. Baysal. Aerodynamic shape optimization using preconditioned conjugate gradient methods. *AIAA paper 93-3322*, July 1993.

[8] G. W. Burgreen and O. Baysal. Three-dimensional aerodynamic shape optimization of wings using sensitivity analysis. *AIAA paper 94-0094*, 32nd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1994.

[9] G. W. Burgreen, O. Baysal, and M. E. Eleshaky. Improving the efficiency of aerodynamic shape optimization procedures. *AIAA paper 92-4697*, September 1992.

[10] R. Campbell. An approach to constrained aerodynamic design with application to airfoils. *NASA Technical Paper 3260*, Langley Research Center, November 1992.

[11] R. Campbell. Efficient constrained design using Navier-Stokes codes. *AIAA paper 95-1808*, 13th Applied Aerodynamics Conference, San Diego, CA, June 1995.

[12] S. Cliff, S. Thomas, T. Baker, W. Cheng, and A. Jameson. Supersonic transport design on the IBM parallel system SP2. Santa Clara Convention Center, Sunnyvale, CA, March 1995. Proceedings of the Computational Aerosciences Workshop.

[13] M. E. Eleshaky and O. Baysal. Preconditioned domain decomposition scheme for three-dimensional aerodynamic sensitivity analysis. In *Proceedings of of the 12th AIAA computational fluid dynamics conference*, pages 1055–1056, July 1993.

[14] M. E. Eleshaky and O. Baysal. Shape optimization of a 3-D nacelle near a flat plate wing using multiblock sensitivity analysis. *AIAA paper 94-0160*, 32nd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1994.

[15] J. Gallman, J. Reuther, N. Pfeiffer, W. Forrest, and D. Bernstorf. Business jet wing design using aerodynamic shape optimization. *AIAA paper 96-0554*, 34th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1996.

[16] P.E. Gill, W. Murray, M.A. Saunders, and M.A. Wright. User's guide for NPSOL (version 4.0). a FORTRAN package nonlinear programming. *Technical Report SOL86-2*, Stanford University, Department of Operations Research, 1986.

[17] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[18] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15:407–412, 1978.

[19] R. M. Hicks, E. M. Murman, and G. N. Vanderplaats. An assessment of airfoil design by numerical optimization. *NASA TM X-3092*, Ames Research Center, Moffett Field, California, July 1974.

[20] A. C. Taylor III, G. W. Hou, and V. M. Korivi. Sensitivity analysis, approximate analysis, and design optimization for internal and external viscous flows. *AIAA paper 91-3083*, September 1991.

[21] A. Jameson. Solution of the Euler equations for two dimensional transonic flow by a multigrid method. *Applied Mathematics and Computations*, 13:327–356, 1983.

[22] A. Jameson. Multigrid algorithms for compressible flow calculations. In W. Hackbusch and U. Trottenberg, editors, *Lecture Notes in Mathematics, Vol. 1228*, pages 166–201. Proceedings of the 2nd European Conference on Multigrid Methods, Cologne, 1985, Springer-Verlag, 1986.

[23] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[24] A. Jameson. Automatic design of transonic airfoils to reduce the shock induced pressure drag. In *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics, Tel Aviv*, pages 5–17, February 1990.

[25] A. Jameson. Optimum aerodynamic design via boundary control. In *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics.* von Karman Institute for Fluid Dynamics, 1994.

[26] A. Jameson. Analysis and design of numerical schemes for gas dynamics 1, artificial diffusion, upwind biasing, limiters and their effect on multigrid convergence. *Int. J. of Comp. Fluid Dyn.*, 4:171–218, 1995.

[27] A. Jameson. Analysis and design of numerical schemes for gas dynamics 2, artificial diffusion and discrete shock structure. *Int. J. of Comp. Fluid Dyn.*, 5:1–38, 1995.

[28] A. Jameson. Optimum aerodynamic design using CFD and control theory. *AIAA paper 95-1729*, AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA, June 1995.

[29] A. Jameson. *Optimum Aerodynamic Design Using Control Theory, Computational Fluid Dynamics Review 1995.* Wiley, 1995.

[30] A. Jameson and J.J. Alonso. Automatic aerodynamic optimization on distributed memory architectures. *AIAA paper 96-0409*, 34th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1996.

[31] A. Jameson and D. A. Caughey. A finite volume method for transonic potential flow calculations. *AIAA Paper 77-635*, 1977.

[32] A. Jameson, L. Martinelli, and N. Pierce. Optimum aerodynamic design using the navier-stokes equations. *AIAA paper 97-0101*, 35th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1997.

[33] A. Jameson and J. Reuther. Control theory based airfoil design using the Euler equations. *AIAA paper 94-4272*, 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, September 1994.

[34] A. Jameson, W. Schmidt, and E. Turkel. Numerical solutions of the Euler equations by finite volume methods with Runge-Kutta time stepping schemes. *AIAA paper 81-1259*, January 1981.

[35] V. M. Korivi, A. C. Taylor III, G. W. Hou, P. A. Newman, and H. E. Jones. Sensitivity derivatives for three-dimensional supersonic Euler code using incremental iterative strategy. In *Proceedings of of the 12th AIAA computational fluid dynamics conference*, pages 1053–1054, July 1993.

[36] V. M. Korivi, A. C. Taylor III, and P. A. Newman. Aerodynamic optimization studies using a 3-D supersonic Euler code with efficient calculation of sensitivity derivatives. *AIAA paper 94-4270*, 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 1994.

[37] V. M. Korivi, A. C. Taylor III, P. A. Newman, G. W. Hou, and H. E. Jones. An incremental strategy for calculating consistent discrete CFD sensitivity derivatives. *NASA TM 104207*, Langley Research Center, Hampton, VA, February 1992.

[38] J.M. Lacasse and O. Baysal. Design optimization of single- and two-element airfoils on multiblock grids. *AIAA paper 94-4273*, 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 1994.

[39] R. M. Lewis. A nonlinear programming perspective on sensitivity calculations for systems governed by state equations. *NASA Contractor Report 201659 97-0103*, ICASE Report, Langley Research Center, February 1997.

[40] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer-Verlag, New York, 1971. Translated by S.K. Mitter.

[41] P. A. Newman, G. W. Hou, H. E. Jones, A. C. Taylor III, and V. M. Korivi. Observations on computational methodologies for use in large-scale gradient-based, multidisciplinary design incorporating advanced CFD codes. *NASA TM 104206*, Langley Research Center, Hampton, VA, February 1992.

[42] J. Reuther. *Practical Aerodynamic Optimization of Airfoils and Wings.* Masters Thesis, University of California Davis, Davis, California, 1991.

[43] J. Reuther, J.J. Alonso, M.J. Rimlinger, and A. Jameson. Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on parallel computers. *AIAA paper 96-4045*, 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September 1996.

[44] J. Reuther, S. Cliff, R. Hicks, and C.P. van Dam. Practical design optimization of wing/body configurations using the Euler equations. *AIAA paper 92-2633*, 1992.

[45] J. Reuther and A. Jameson. Control theory based airfoil design for potential flow and a finite volume discretization. *AIAA paper 94-0499*, 32nd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1994.

[46] J. Reuther and A. Jameson. Aerodynamic shape optimization of wing and wing-body configurations using control theory. *AIAA paper 95-0123*, 33rd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1995.

[47] J. Reuther and A. Jameson. A comparison of design variables for control theory based airfoil optimization. Technical report, 6th International Symposium on Computational Fluid Dynamics, Lake Tahoe, Nevada, September 1995.

[48] J. Reuther and A. Jameson. Supersonic wing and wing-body shape optimization using an adjoint formulation. Technical report, The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, California, November 1995.

[49] J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders. Contrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers. *AIAA paper 97-0103*, 35th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1997.

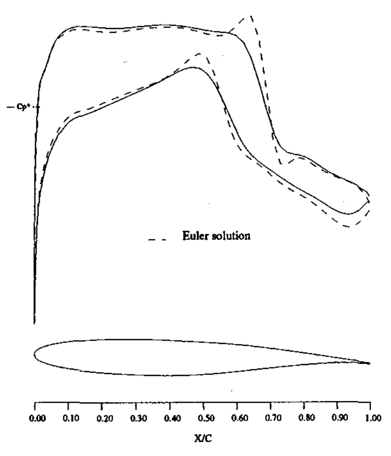
[50] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. *AIAA paper 96-0094*, 34th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1996.

[51] J. Reuther, C.P. van Dam, and R. Hicks. Subsonic and transonic low-Reynolds-number airfoils with reduced pitching moments. *Journal of Aircraft*, 29:297–298, 1992.

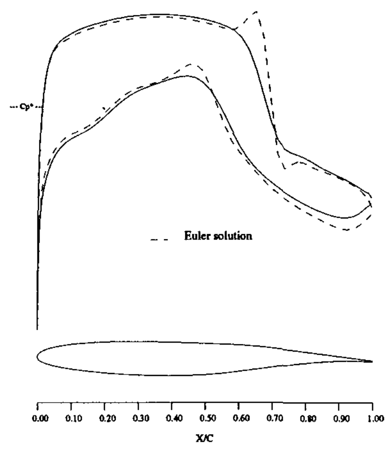
[52] J. J. Reuther. Aerodynamic shape optimization using control theory. *Ph. D. Dissertation*, University of California, Davis, Davis, CA, June 1996.

[53] J.P. Steinbrenner, J.R. Chawner, and C.L. Fouts. The GRIDGEN 3D multiple block grid generation system. Technical report, Flight Dynamics Laboratory, Wright Research and Development Center, Wright-Patterson Air Force Base, Ohio, July 1990.

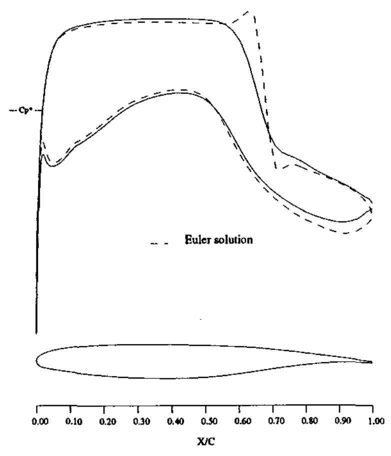
[54] S. Tatsumi, L. Martinelli, and A. Jameson. A new high resolution scheme for compressible viscous flows with shocks. *AIAA paper* To Appear, AIAA 33nd Aerospace Sciences Meeting, Reno, Nevada, January 1995.



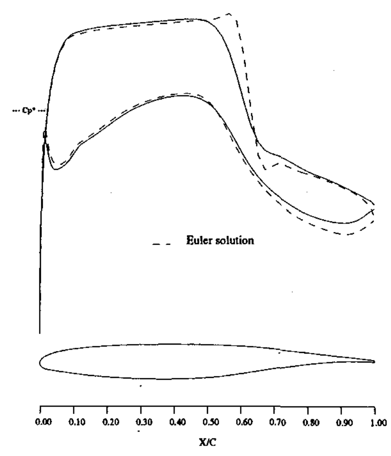
1a: span station $z = 0.283$



1b: span station $z = 0.472$

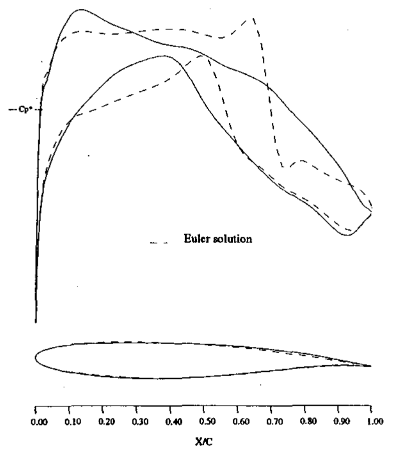


1c: span station $z = 0.660$

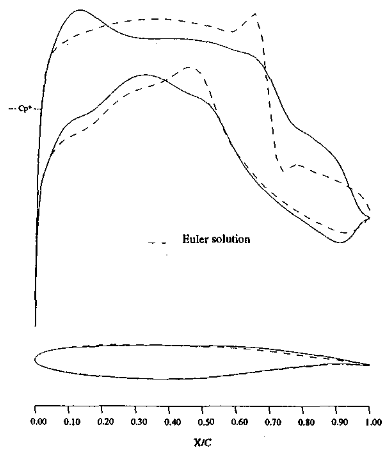


1d: span station $z = 0.849$

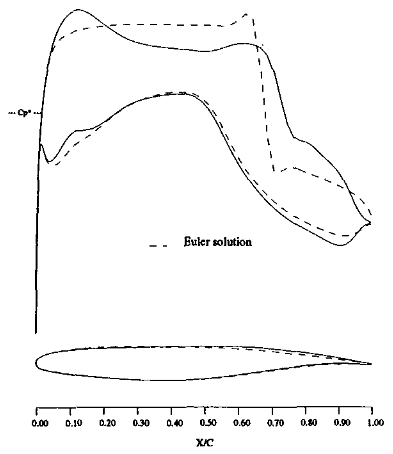
Figure 1: Business Jet Configuration Configuration. Comparison of Baseline Solutions.
$M = 0.82$, $C_L = 0.35$
- - -, Euler Solution Pressure Distribution.
—, Navier-Stokes Solution Pressure Distribution.



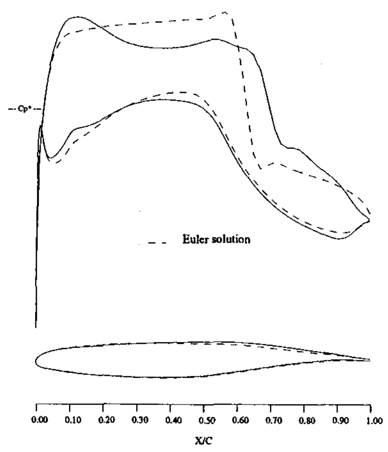
2a: span station $z = 0.283$



2b: span station $z = 0.472$



2c: span station $z = 0.660$



2d: span station $z = 0.849$

Figure 2: Business Jet Configuration. Euler Based Drag Minimization at Fixed Lift.
$M = 0.82$, $C_L = 0.35$
108 Hicks-Henne variables. Spar Constraints Active.
- - -, Initial Pressures
—, Pressures After 6 Design Cycles.

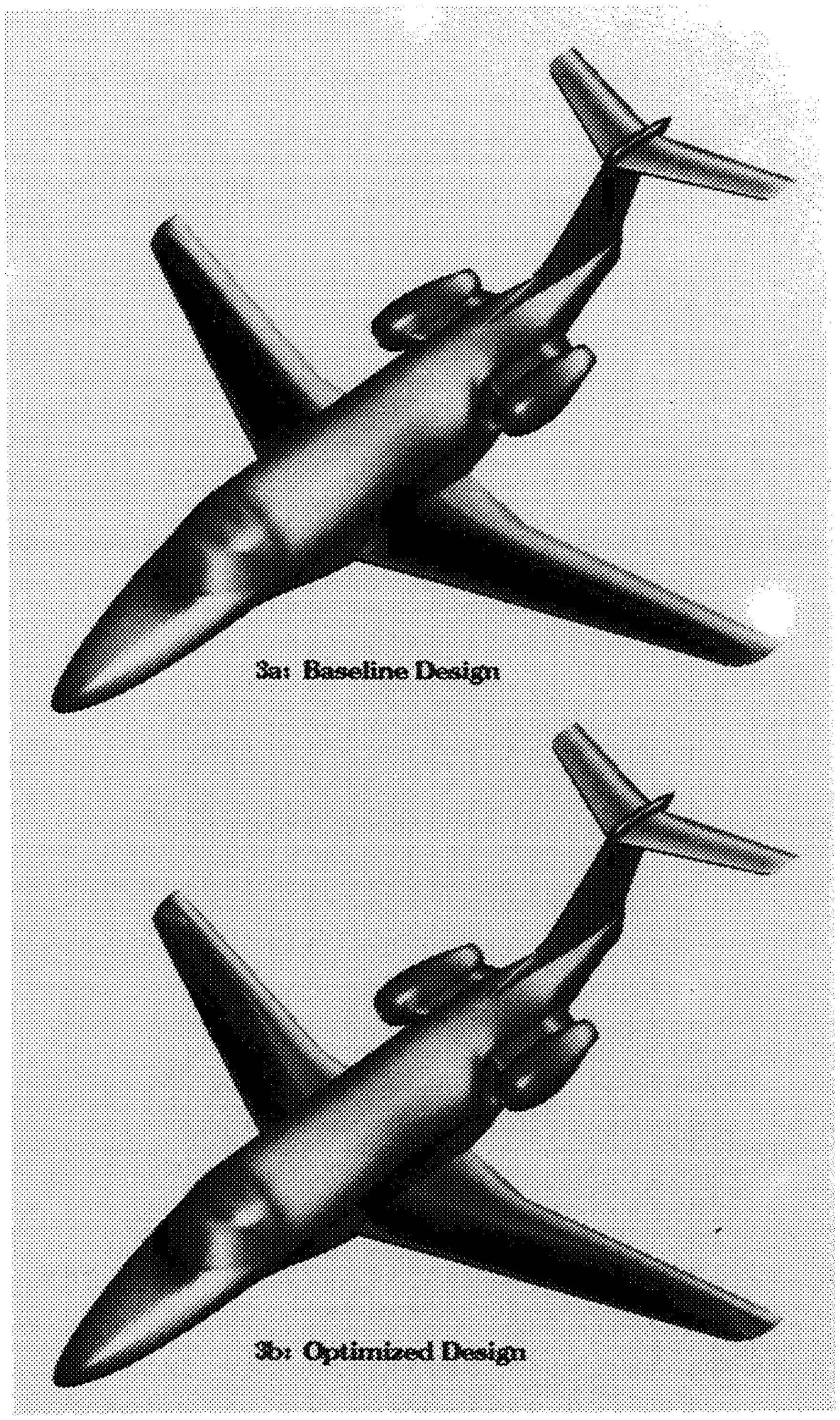
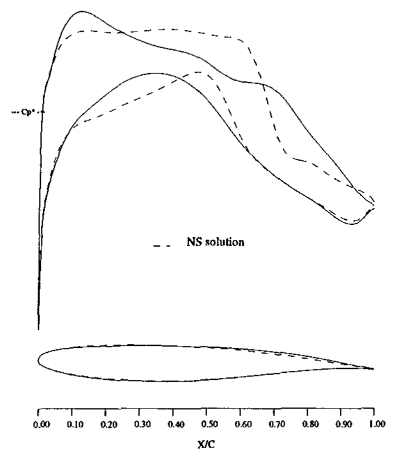
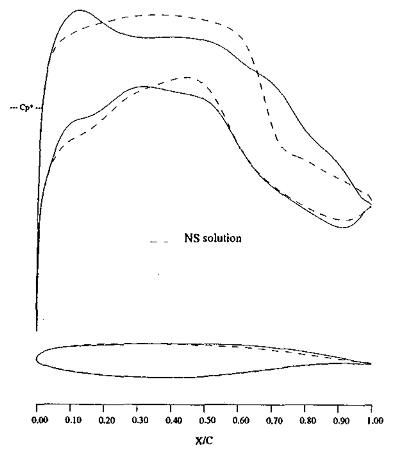


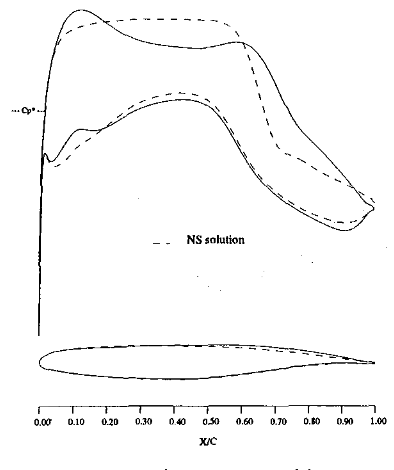
Figure 3: Transonic Business Jet Configuration
Iso-Cp Contours, Navier-Stokes.
Baseline and Optimized Designs.
$M = 0.82, \quad C_L = 0.35$



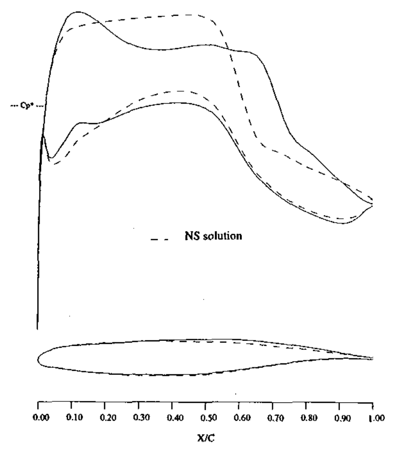
4a: span station $z = 0.283$



4b: span station $z = 0.472$

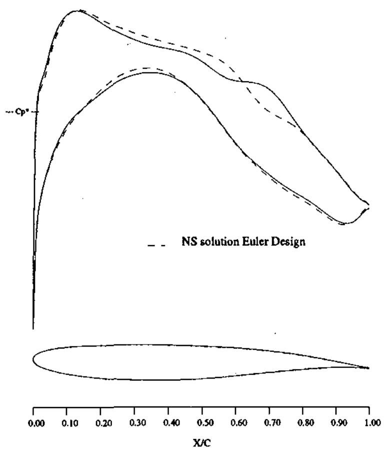


4c: span station $z = 0.660$

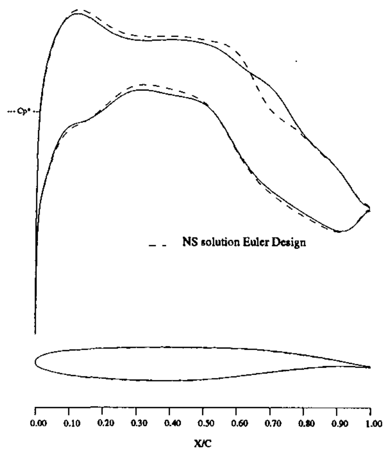


4d: span station $z = 0.849$

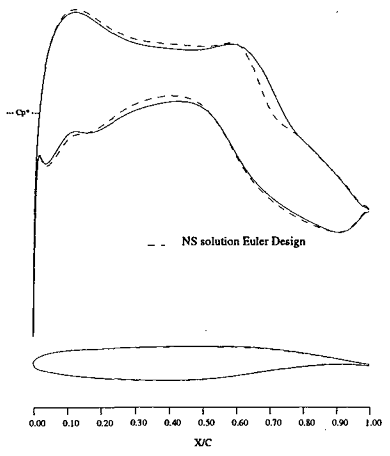
Figure 4: Business Jet Configuration. Navier-Stokes Based Drag Minimization at Fixed Lift.
$M = 0.82$, $C_L = 0.35$
108 Hicks-Henne variables. Spar Constraints Active.
- - -, Initial Pressures
—, Pressures After 5 Design Cycles.



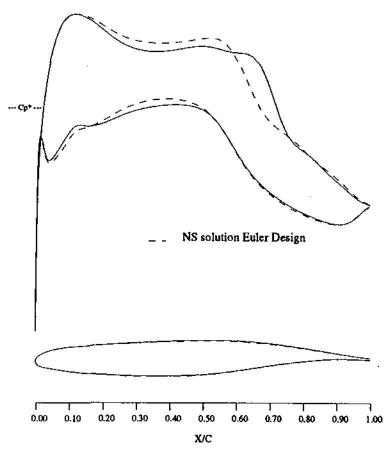
5a: span station $z = 0.283$



5b: span station $z = 0.472$

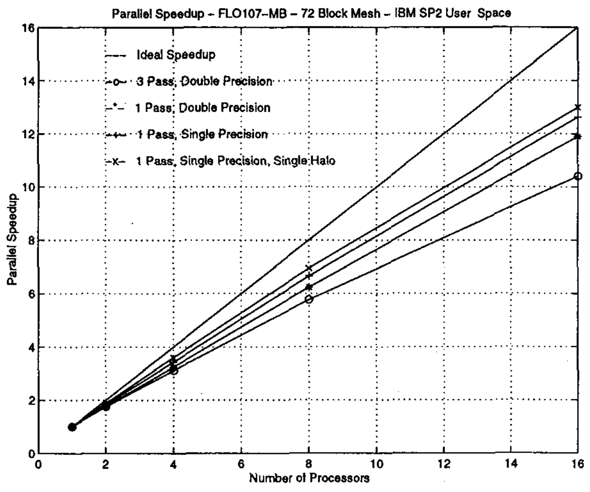


5c: span station $z = 0.660$

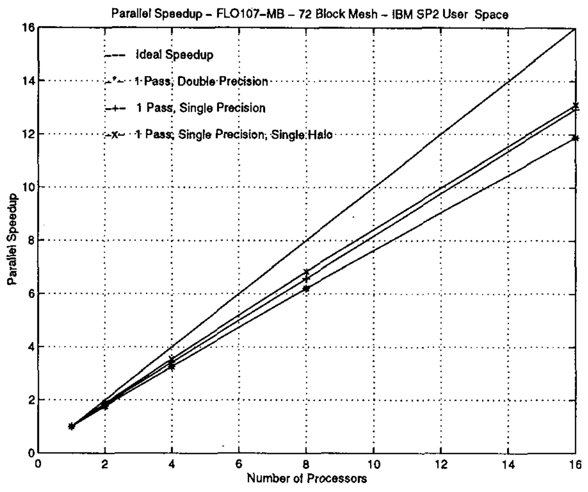


5d: span station $z = 0.849$

Figure 5: Business Jet Configuration. Comparison of Euler and Navier-Stokes Designs.
$M = 0.82$, $C_L = 0.35$
108 Hicks-Henne variables. Spar Constraints Active.
- - -, Navier-Stokes Pressures for Euler Based Design.
—, Navier-Stokes Pressures for Navier-Stokes Based Design.

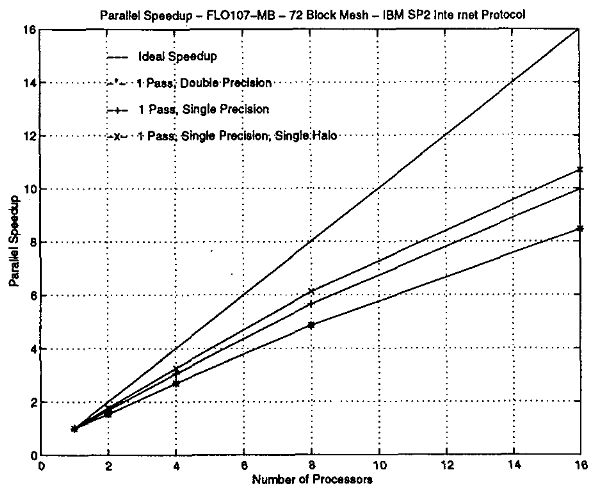


6a: Original Load Balancing Scheme.

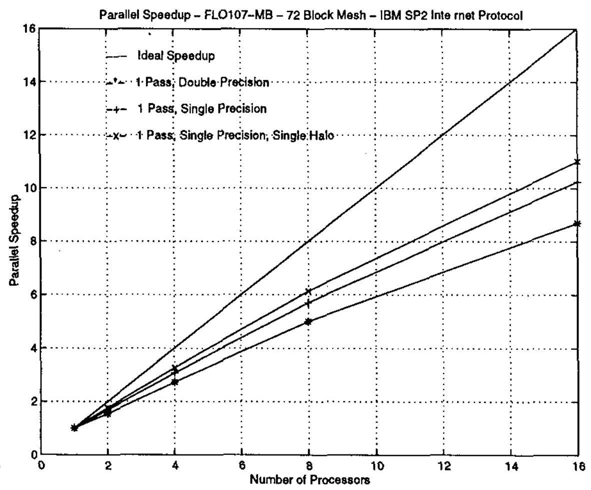


6b: Improved Load Balancing Scheme.

Figure 6: Parallel Speedup for Benchmark Mesh Using the IBM SP2 in User Space Mode

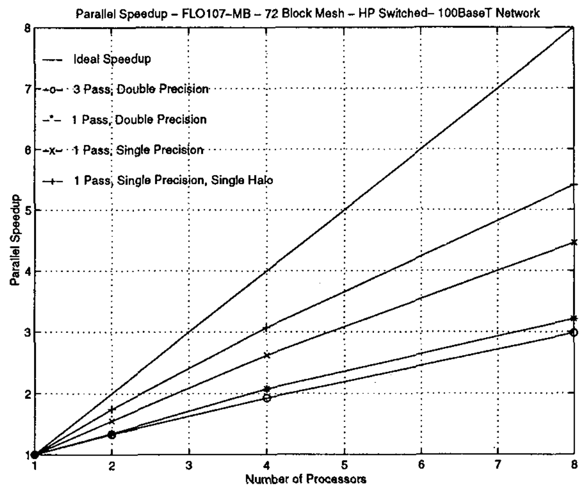


7a: Original Load Balancing Scheme.

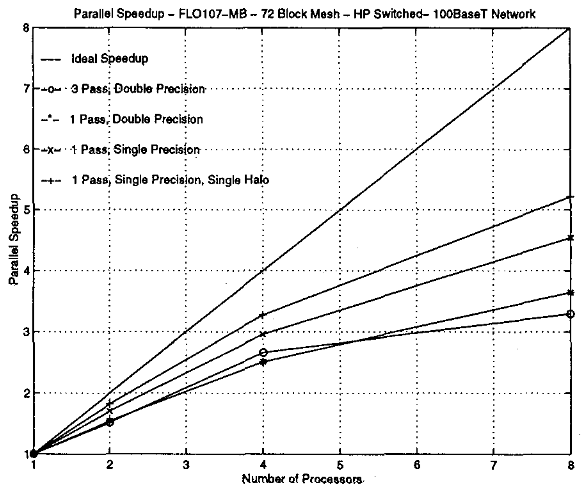


7b: Improved Load Balancing Scheme.

Figure 7: Parallel Speedup for Benchmark Mesh Using the IBM SP2 in Internet Protocol Mode

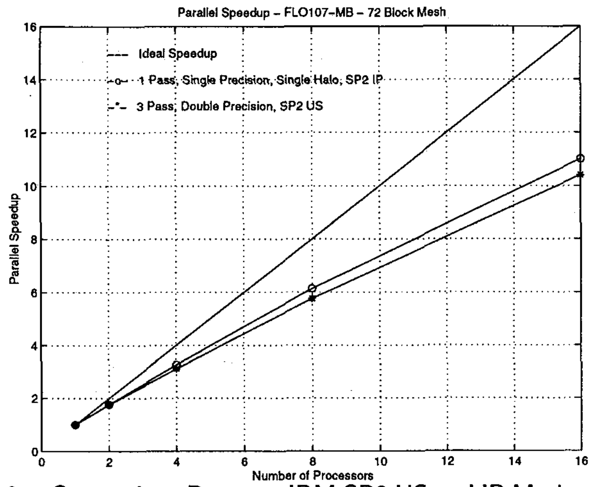


8a: Original Load Balancing Scheme.

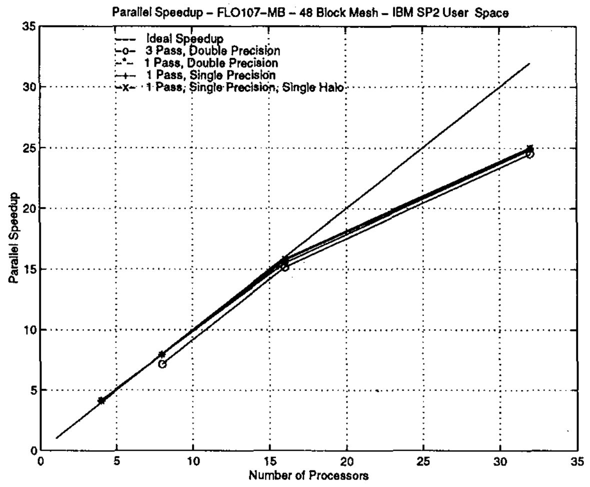


8b: Improved Load Balancing Scheme.

Figure 8: Parallel Speedup for Benchmark Mesh Using the HP Cluster with Switched-100BaseT Fast Ethernet



9a: Comparison Between IBM SP2 US and IP Modes for the Benchmark Mesh.



9b: Parallel Speedups for Fine Mesh.

Figure 9: Parallel Speedup Comparison Results for Benchmark and Fine Meshes