AIAA Meeting Papers on Disc, September 1996
A9639989, AF-AFOSR-91-0391, N00014-92-J-1976, AIAA Paper 96-4045

Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on distributed memory parallel computers

J. Reuther
NASA, Ames Research Center, Moffett Field, CA

M. J. Rimlinger
NASA, Ames Research Center, Moffett Field, CA

J. J. Alonso
Princeton Univ., NJ

A. Jameson
Princeton Univ., NJ

AIAA, NASA, and ISSMO, Symposium on Multidisciplinary Analysis and Optimization, 6th, Bellevue, WA, Sept. 4-6, 1996

This work describes the application of a control theory-based aerodynamic shape optimization method to the problem of supersonic aircraft design. The design process is greatly accelerated through the use of both control theory and a parallel implementation on distributed memory computers. Control theory is employed to derive the adjoint differential equations whose solution allows for the evaluation of design gradient information at a fraction of the computational cost required by previous design methods. The resulting problem is then implemented on parallel distributed memory architectures using a domain decomposition approach, an optimized communication schedule, and the MPI (Message Passing Interface) Standard for portability and efficiency. The final result achieves very rapid aerodynamic design based on higher order CFD methods. In this paper, our concern will be to demonstrate that the combined power of these new technologies can be used routinely in an industrial design environment by applying it to the case study of the design of typical supersonic transport configurations. (Author)

96-4045
A96-39989

Aerodynamic Shape Optimization of Supersonic Aircraft Configurations via an Adjoint Formulation on Distributed Memory Parallel Computers

J. Reuther*
Research Institute for Advanced Computer Science
NASA Ames Research Center, MS 227-6
Moffett Field, California 94035, U.S.A.

J. J. Alonso†
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, New Jersey 08544, U.S.A.

M. J. Rimlinger‡
Simco
NASA Ames Research Center, MS 227-6
Moffett Field, California 94035, U.S.A.

A. Jameson§
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, New Jersey 08544, U.S.A.

ABSTRACT

This work describes the application of a control theory-based aerodynamic shape optimization method to the problem of supersonic aircraft design. The design process is greatly accelerated through the use of both control theory and a parallel implementation on distributed memory computers. Control theory is employed to derive the adjoint differential equations whose solution allows for the evaluation of design gradient information at a fraction of the computational cost required by previous design methods [13, 12, 44, 38]. The resulting problem is then implemented on parallel distributed memory architectures using a domain decomposition approach, an optimized communication schedule, and the MPI (Message Passing Interface) Standard for portability and efficiency. The final result achieves very rapid aerodynamic design based on higher order computational fluid dynamics methods (CFD).

In our earlier studies, the serial implementation of this design method [19, 20, 21, 23, 39, 25, 40, 41, 42, 43, 9] was shown to be effective for the optimization of airfoils, wings, wing-bodies, and complex aircraft configurations using both the potential equation and the Euler equations[39, 25]. In our most recent paper, the Euler method was extended to treat complete aircraft configurations via a new multiblock implementation. Furthermore, during the same conference, we also presented preliminary results demonstrating that this basic methodology could be ported to distributed memory parallel computing architectures [24]. In this paper, our concern will be to demonstrate that the combined power of these new technologies can be used routinely in an industrial design environment by applying it to the case study of the design of typical supersonic transport configurations. A particular difficulty of this test case is posed by the propulsion/airframe integration.

INTRODUCTION

To realize the potential of CFD to produce superior designs, there is a need not only for accurate aerodynamic prediction algorithms, but also for design methods capable of creating new optimum configurations. Yet, while flow analysis has matured to the extent that Navier-Stokes calculations are routinely carried out over very complex configurations, CFD-based design techniques are just beginning to treat moderately complex three-dimensional configurations.

Existing CFD analysis methods have previously been used to treat the design problem by coupling them with numerical optimization methods [13, 12, 44, 38]. The essence of these methods, which incur heavy computational expense, is very simple: a numerical optimization procedure is used to extremize a chosen aerodynamic figure of merit which is evaluated by the given CFD code. The configuration is systematically modified through user specified design variables. Most of these optimization procedures require the evaluation of the gradient of the cost function with respect to the specified

*Student Member AIAA
†Student Member AIAA
‡Student Member AIAA
§James S. McDonnell Distinguished University Professor of Aerospace Engineering, AIAA Fellow

design variables. The simplest of the methods to obtain these necessary gradients is the finite difference method. In this technique, the gradient components are obtained by independently perturbing each design variable with a finite step, calculating the corresponding value of the objective function using CFD analysis, and forming the ratio of the differences. The gradient is used by the numerical optimization algorithm to calculate a search direction using steepest descent, conjugate gradient, or quasi-Newton techniques. After finding the minimum or maximum of the objective function along the search direction, the entire process is repeated until the gradient approaches zero and further improvement is not possible.

The finite difference-based optimization strategy is computationally expensive because the flow must be repeatedly calculated for perturbations in every design variable. Nevertheless, it is attractive when compared with other traditional design strategies such as inverse methods, since it permits any choice of the aerodynamic figure of merit. The use of numerical optimization for transonic aerodynamic shape design was pioneered by Hicks, Murman and Vanderplaats [13]. They applied the method to two-dimensional profile design governed by the potential flow equation. The method was quickly extended to wing design by Hicks and Henne [12]. Later, in the work of Reuther, Cliff, Hicks and Van Dam, this method was successfully used for the design of supersonic wing-body transport configurations [38]. However all of these cases, which were confined to finite difference gradients on serial computer architectures, were limited in their geometric complexity simply due to computational expense. For example, the designs presented in [38] were limited to wing-body configurations. Yet it is well known that optimum performance (especially for supersonic configurations) will require highly tuned nacelle/airframe integrations. It was not possible to include nacelle/airframe considerations into the design problem outlined in [38] since the required number of mesh points, which more than doubles with the inclusion of nacelles, could not be afforded.

In the last few years, alternative methods for obtaining design sensitivities have been developed which greatly reduce the computational cost of optimization. References [1, 2, 3, 5, 4, 7, 8, 6, 32, 29, 16, 35, 30, 28, 34, 47, 31, 36, 37, 49, 15, 33, 14] present a partial list of recent works in this developing area of research. An exhaustive report on the various approaches to the problem and their advantages and disadvantages is given by the first author in [45]. The most promising of these approaches is the adjoint formulation whereby the sensitivity of some objective function with respect to an arbitrary number of design variables is obtained with the equivalent of only one additional flow calculation. Here, the solution of the adjoint system (using the same mature techniques as perfected for the flow equations) enables each gradient element to be calculated very cheaply, meaning the number of design variables is essentially eliminated as a constraining factor. Moreover, the adjoint solution (and to a lesser extent the accompanying flow solution) need not be highly converged to be useful, in significant contrast to the highly-converged flow solutions which are crucial to accurate finite difference gradients [45].

In spite of the large decrease in computational cost provided by the adjoint formulation of the design problem, the aerodynamic optimization of a complete configuration still remains a formidable computational task. The advent of reliable and efficient parallel computers using distributed memory is a key enabling technology to decrease the turnaround time of these design calculations to the point where configurations can be optimized almost in real time.

The work presented in this paper combines these two ingredients (adjoint formulations and parallel implementations) to produce a robust, accurate, and efficient method that can be used for the design of supersonic aircraft including the effects of airframe/nacelle integration.

FORMULATION OF THE ADJOINT EQUATIONS

The aerodynamic properties which define the cost function I are functions of the flow field variables, w, and the physical location of the boundary, which may be represented by the function $\mathcal{F}$. That is,

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

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

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

in the cost function. The governing equation R and its first variation express the interdependence of w and $\mathcal{F}$ within the flow field domain D:

$$R(w, \mathcal{F}) = 0, \quad \delta R = \left[\frac{\partial R}{\partial w}\right]\delta w + \left[\frac{\partial R}{\partial \mathcal{F}}\right]\delta \mathcal{F} = 0. \tag{2}$$

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

$$\left[\frac{\partial R}{\partial w}\right]^T \psi = \frac{\partial I}{\partial w} \tag{3}$$

the first term is eliminated, and we find that the desired gradient given by

$$\mathcal{G}^T = \frac{\partial I^T}{\partial \mathcal{F}} - \psi^T\left[\frac{\partial R}{\partial \mathcal{F}}\right] \tag{4}$$

is only a function of $\mathcal{F}$. Since (4) is independent of δw, the gradient of I with respect to an arbitrary number of design variables can be determined without the need for additional flow field evaluations. The main cost is in solving the adjoint equation (3). In general, the adjoint problem is about as complex as a flow solution. Therefore, when the number of design variables is larger than 2, it becomes compelling to take advantage of the cost differential between one adjoint solution and the large number of flow field evaluations required to determine the gradient by finite differences. Once equation (4) is obtained, $\mathcal{G}$ can be provided to a variety of numerical optimization algorithms to obtain an improved design.

MULTIBLOCK FLOW SOLUTION

In order to extend the methods presented in our earlier three-dimensional work to the treatment of complete aircraft configurations, the single-block flow solver used in [21, 40, 42] must be replaced. As with the single-block solver, the more general flow

solver must meet fundamental requirements of accuracy, efficiency, and robust convergence to be employed in an automated design environment. High accuracy is required since the predicted improvements in the design realized by the method can only be as good as the accuracy of the flow analysis. Efficiency of the flow solver is also critical since the optimization of the design will generally require the computation of many flow solutions or other solutions of comparable complexity. Finally, robust convergence is also of significant importance since the main benefit of aerodynamic optimization is in obtaining the last few percentage points in improved efficiency. The solutions must be converged well enough that the noise in the figure of merit, say drag at a fixed lift, is well below the level of realizable improvement. The desirable ability to compare adjoint-based gradients with finite differencing as a check also requires highly-converged flow solutions.

In our three-dimensional single-block applications, the FLO87 code written by the third author readily met all of the above criteria. FLO87 achieves fast convergence with the aid of multigridding and residual smoothing. It is normally easy to obtain solutions that converge to machine accuracy. The challenge in the present work was to meet these strict requirements within the framework of a multiblock flow solver. The use of a multiblock approach is a first step towards the treatment of more complex configurations. However, the multiblock strategy presented here is not the only viable approach. Other alternatives such as unstructured mesh solvers are also currently under investigation.

The general strategy in developing the multiblock flow solver is to construct and update a halo of cells around each block such that the flow solution inside each block is transparent to the block boundaries. This task requires establishing the size and location of halo cells adjacent to block boundaries and loading the halo cell values with appropriate flow field data at the appropriate times. To accomplish this task, a two-level halo is constructed around each block. The requirement of this double halo results from the necessity of preserving a complete stencil of calculated fluxes entering and leaving each cell in the entire domain without regard to block boundaries. This ensures that the conservative flow solution algorithm is fully maintained. Since both the convective and the dissipative fluxes are calculated at the cell faces (boundaries of the control volumes), all six neighboring cells are necessary, thus requiring the existence of a single level halo for each block in the multiblock calculation. The dissipative fluxes are composed of a blend of first and third order differences corresponding to terms that mimic second and fourth derivatives of the flow quantities [26]. For the third order differences at the boundary faces of each cell for all blocks, the presence of the twelve neighboring cells (two adjacent to each face) is required. For each cell within a block, Figure 1 shows the neighboring cells that are required for the calculation of convective and dissipative fluxes. For each block, some of these cells will lie directly next to an interblock boundary, in which case, the values of the flow variables residing in a different block will be necessary to calculate the convective and dissipative fluxes. Halo cells on the external boundary of the entire computational domain are constructed and updated by extrapolation and reflection, depending on the kind of boundary condition applied. Once the halo configuration is set up for each block, standard methods for spatial discretization and time integration (including artificial dissipation, implicit residual averaging, and multigridding) are employed to compute the flow solution within each individual block.

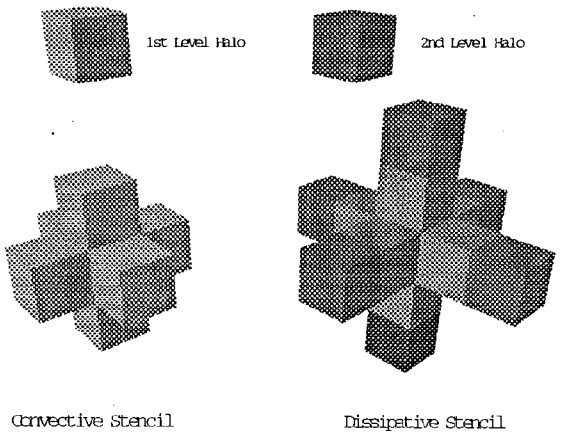


Figure 1: Convective and Dissipative Discretization Stencils.

The strategy for a complete flow solution proceeds as follows: First, the blocks that comprise the flow field mesh are read from an external file. Then, the double halo configuration is established, for each individual block, by inserting into halo cell locations values for grid metrics, etc., taken from the interior cells of adjacent blocks. For the coarse grids required in the multigrid procedure, the process is repeated with coarse grid halo cells defined by the internal cells of adjacent coarse grid blocks. For block faces that lie on solid, symmetry, or far field boundaries, standard single-block techniques are used to define the halo cells. As an example, consider the simple 4-block grid depicted in Figure 3. The halo cells for block I will be obtained from the internal cells of blocks II, III, and IV, and from solid or far field boundary techniques for the faces not adjacent to other blocks. Coarse grids are computed in the usual fashion, by aggregating groups of eight cells and then repeating the above halo cell process. Once the halo configuration is complete for the fine and all coarse grids, the flow solution commences.

The system of equations solved here as well as the solution strategy follows that presented in many earlier works [26, 18, 17]. The three-dimensional Euler equations may be written as

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = 0 \quad \text{in } D, \tag{5}$$

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

$$w = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{Bmatrix}, \quad f_i = \begin{Bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i H \end{Bmatrix} \tag{6}$$

where δ_{ij} is the Kronecker delta. Also,

$$p = (\gamma - 1)\rho \left\{ E - \frac{1}{2}\left(u_i^2\right) \right\}, \tag{7}$$

and

$$\rho H = \rho E + p, \tag{8}$$

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

$$\frac{\partial W}{\partial t} + \frac{\partial F_i}{\partial \xi_i} = 0 \quad \text{in } D, \tag{9}$$

with

$$W = J \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{Bmatrix}, \quad F_i = Q_{ij} f_j = \begin{Bmatrix} \rho U_i \\ \rho U_i u_1 + Q_{i1} p \\ \rho U_i u_2 + Q_{i2} p \\ \rho U_i u_3 + Q_{i3} p \\ \rho U_i H \end{Bmatrix}. \tag{10}$$

For the multiblock case, the above notation applies to each block in turn. The flow is thus determined as the steady-state solution to equation (9) in all blocks subject to the flow tangency conditions on all solid boundary faces of all blocks:

$$U_\eta = 0 \quad \text{on all } B_S \tag{11}$$

where η is 1, 2, or 3 depending on the direction that is normal to the face B_S where a solid surface is assumed to exist. 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 strategy [26]. 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 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 integration 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, multistage time stepping, and multigrid convergence acceleration.

The only difference between the integration strategies is in the implementation of the implicit residual averaging technique. In the single-block solution strategy, a tridiagonal system of equations is set up and solved using flow information from the entire grid. Thus, each residual is replaced by an average of itself and the residuals of the entire grid. In the multiblock strategy, the support for the implicit residual smoothing is reduced to the extent of each block, in order to eliminate the need to solve large tridiagonal systems spanning the blocks, which would incur a penalty in communication costs and may not even be defined. This change has no effect on the final converged solution, and in the present application has not led to any significant reduction in the rate of convergence.

THE ADJOINT FORMULATION FOR THE EULER EQUATIONS

The application of control theory to aerodynamic design problems is illustrated by treating the case of three-dimensional design, using the Euler equations discussed above as the mathematical model for compressible flow. In our previous work, the illustrative problem most often used specified the cost function as a measure of the difference between the current and some desired pressure distribution. In the case of transonic flows over conventional commercial transport wings this aerodynamic figure of merit proves to be very effective since the tailoring of these pressure distributions to achieve close to optimum performance is well understood by most aerodynamicists. However, for the case of supersonic design of three-dimensional configurations, the specification of pressure distributions that will determine near optimum performance is a considerably more challenging problem. Thus the development here will focus on the more salient problem for supersonic design: drag minimization at a fixed lift.

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

where S_x and S_y are projected surface areas, $\mathbf{S_{ref}}$ is the reference area, and $d\xi_1$ and $d\xi_2$ are the two coordinate indices that define 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 to be chosen to minimize I, subject to the constraints defined by the flow equations (5–10). A variation in the shape will cause a variation δp in the pressure and consequently a variation in the cost function

$$\begin{aligned} \delta I = \quad & \tilde{\delta} C_A \cos\alpha + \tilde{\delta} C_N \sin\alpha \\ + \quad & \{-C_A \sin\alpha + C_N \cos\alpha\} \delta\alpha \\ + \quad & \left\{\frac{\partial C_A}{\partial \alpha} \cos\alpha + \frac{\partial C_N}{\partial \alpha} \sin\alpha\right\} \delta\alpha \end{aligned}$$

where $\tilde{\delta} C_A$ and $\tilde{\delta} C_N$ are variations due to changes in the design parameters with α fixed. To treat the interesting 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 gives

$$\begin{aligned} & \tilde{\delta} C_N \cos\alpha - \tilde{\delta} C_A \sin\alpha \\ + \quad & \{-C_N \sin\alpha - C_A \cos\alpha\} \delta\alpha \\ + \quad & \left\{\frac{\partial C_N}{\partial \alpha} \cos\alpha - \frac{\partial C_A}{\partial \alpha} \sin\alpha\right\} \delta\alpha = 0 \end{aligned}$$

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

$$\begin{aligned}\delta I \quad &= \tilde{\delta}C_A\cos\alpha + \tilde{\delta}C_N\sin\alpha \\ &+\Omega\left\{\tilde{\delta}C_N\cos\alpha - \tilde{\delta}C_A\sin\alpha\right\},\end{aligned} \tag{12}$$

where Ω is given by

$$\Omega = \frac{\left(-C_L + \frac{\partial C_A}{\partial\alpha}\cos\alpha + \frac{\partial C_N}{\partial\alpha}\sin\alpha\right)}{\left(C_D + \frac{\partial C_N}{\partial\alpha}\cos\alpha - \frac{\partial C_A}{\partial\alpha}\sin\alpha\right)}.$$

Since p depends on w through the equation of state (7–8), 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

$$A_i = \frac{\partial f_i}{\partial w}, \quad C_i = Q_{ij}\Lambda_j. \tag{13}$$

Then the equation for δw in the steady state becomes

$$\frac{\partial}{\partial\xi_i}\left(\delta F_i\right) = 0,$$

where

$$\delta F_i = C_i\delta w + \hat{\delta}\left(Q_{ij}\right) f_j.$$

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

$$\int_D\left(\frac{\partial\psi^T}{\partial\xi_i}\delta F_i\right)d\xi_j = \int_B\left(\bar{n}_i\psi^T\delta F_i\right)d\xi_j, \tag{14}$$

where $\bar{n}_i$ are components of a unit vector normal to the boundary. The variation in the cost function can also be expressed in terms of δp after (12) and (14) are summed to give,

$$\begin{aligned}\delta I = \quad & \frac{1}{\frac{1}{2}\gamma M_\infty^2\mathbf{S_{ref}}}\iint_{B_S}\delta p\left\{\left(S_x\cos\alpha + S_y\sin\alpha\right)\right. \\ &\left.+\Omega\left(S_y\cos\alpha - S_x\sin\alpha\right)\right\}d\xi_1 d\xi_2 \\ + \quad & \frac{1}{\mathbf{S_{ref}}}\iint_{B_S}C_p\left\{\left(\delta S_x\cos\alpha + \delta S_y\sin\alpha\right)\right. \\ &\left.+\Omega\left(\delta S_y\cos\alpha - \delta S_x\sin\alpha\right)\right\}d\xi_1 d\xi_2 \\ &-\int_D\left(\frac{\partial\psi^T}{\partial\xi_i}\delta F_i\right)d\xi_j + \int_B\left(\bar{n}_i\psi^T\delta F_i\right)d\xi_j.\end{aligned} \tag{15}$$

On the solid surfaces B_S, $\bar{n}_1 = \bar{n}_2 = 0$. It follows from equation (11) that

$$\delta F_\eta \quad = \quad \begin{Bmatrix}0\\ Q_{\eta 1}\delta p\\ Q_{\eta 2}\delta p\\ Q_{\eta 3}\delta p\\ 0\end{Bmatrix} + p\begin{Bmatrix}0\\ \delta\left(Q_{\eta 1}\right)\\ \delta\left(Q_{\eta 2}\right)\\ \delta\left(Q_{\eta 3}\right)\\ 0\end{Bmatrix} \quad \text{on any } B_S. \tag{16}$$

Suppose now that ψ is the steady-state solution of the adjoint equation

$$\frac{\partial\psi}{\partial t} - C_i^T\frac{\partial\psi}{\partial\xi_i} = 0 \quad \text{in } D. \tag{17}$$

At internal block boundaries, the face integrals cancel from 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

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

If, however, the flow is subsonic and the boundary is fairly close, then far field faces may be set by $\psi_{1-5} = 0$ for incoming waves, while outgoing waves are determined by the solution. It is noted that the waves in the adjoint problem propagate in the opposite direction to those in the flow problem due to the transpose in equation (17). For supersonic flows, the choice of boundary conditions at the outer domain can be developed from physical intuition as well as mathematical analysis. For a given geometry, say a wing, a change in the surface at any particular point, $\mathcal{P}$, will incur changes in the flow field and hence the performance in the region defined by the Mach cone originating at $\mathcal{P}$. Similarly, it is possible to determine the region over which surface changes affect the flow condition at a given point. This region would also form a cone that would point exactly in the opposite direction of the Mach cone, depending on local conditions. It is the solution of this reverse problem that the adjoint represents. The contribution to, say, drag at a given point is influenced by changes to the surface at all points within the reverse cone. The correct supersonic far field boundary conditions for the adjoint equation that are consistent with this reversed character are:

ψ_{1-5} = 0 at the exit;

ψ_{1-5} extrapolated from the interior at the inflow boundary.

Then if the coordinate transformation is such that $\delta\left(JK^{-1}\right)$ is negligible in the far field, the last integral in (15) reduces to

$$-\iint_{B_S}\psi^T\delta F_\eta\, d\xi_1 d\xi_2. \tag{18}$$

Thus by letting ψ satisfy the boundary conditions,

$$\left(\psi_2 Q_{\eta 1} + \psi_3 Q_{\eta 2} + \psi_4 Q_{\eta 3}\right) = \mathcal{Q} \quad \text{on all } B_S, \tag{19}$$

where

$$\begin{aligned}\mathcal{Q} \quad = \quad & \frac{1}{\frac{1}{2}\gamma M_\infty^2\mathbf{S_{ref}}}\left\{\left(S_x\cos\alpha + S_y\sin\alpha\right)\right. \\ &\left.+\Omega\left(S_y\cos\alpha - S_x\sin\alpha\right)\right\},\end{aligned}$$

we find after integrating by parts again that

$$\begin{aligned}\delta I = \quad & \frac{1}{\mathbf{S_{ref}}}\iint_{B_S}C_p\left\{\left(\delta S_x\cos\alpha + \delta S_y\sin\alpha\right)\right. \\ &\left.+\Omega\left(\delta S_y\cos\alpha - \delta S_x\sin\alpha\right)\right\}d\xi_1 d\xi_2 \\ &+\int_D\psi^T\frac{\partial}{\partial\xi_i}\left(\delta Q_{ij}f_j\right)d\xi_k\end{aligned} \tag{20}$$

which is independent of w.

MULTIBLOCK MESH VARIATIONS AND DESIGN VARIABLES

In order to construct δI in equation (20), the variation in the metric terms must be obtained in each block. One way to accomplish this is to use finite differences to calculate the necessary information. This approach avoids the use of multiple flow solutions to determine the gradient, but it unfortunately still requires the mesh generator to be used 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 often be solved iteratively in order to obtain acceptably smooth meshes. These iterative mesh generation procedures are often 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.

In our earlier works [40, 39, 25, 19, 20, 21], two methods have been explored which avoid these difficulties. In the first method, a completely analytic mapping procedure was used for the mesh generation. This technique is not only fully automatic and results in smooth consistent meshes, but it also allows for complete elimination of finite difference information for the mesh metric terms. Since the mapping function fully determines the entire mesh based on the surface shape, this analytic relationship may be directly differentiated in order to obtain the required information without considering a finite step. An analytic mapping method requires the geometry topology to be built directly into the formulation, and only works for simple configurations. Nevertheless, within these limitations it has proven to be highly effective [19, 20, 21].

The second method that we have explored is the use of an analytic 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.[46]. This initial mesh becomes the basis for all subsequent meshes which are developed by analytical perturbations. In the method that was previously developed for wing-body configurations it had been assumed that only one surface, say the wing, was perturbed during a design case. This permitted the use of a very simple algebraic mesh perturbation algorithm. New meshes are created by moving all the mesh points on an index line projecting from the surface by an amount which is attenuated as the arc length from the surface increases. If the outer boundary of the grid domain is held constant the modification to the grid has the form

$$x_i^{new} = x_i^{old} + \mathcal{S}^{old}\left(x_{s_i}^{new} - x_{s_i}^{old}\right) \qquad (21)$$

where x_i represents the volume grid points, x_{s_i} represents the surface grid points and $\mathcal{S}$ represents the arc length along the radial mesh line measured from the outer domain, normalized so that $\mathcal{S} = 1$ at the inner surface. Unfortunately this simple logic breaks down in the case where multiple faces sharing common edges are allowed to move. Thus in order to use analytic mesh perturbations for the treatment of the more general problem where multiple faces of a given block may be simultaneously deformed, equation (21) had to be modified in a way that resembles transfinite interpolation (TFI) [48]. Unlike TFI, where there is no prior knowledge of the interior mesh, the perturbation algorithm developed here (WARP3D) does make use of the relative interior point distributions in the initial mesh.

The WARP3D algorithm has been modified from that presented in reference [43] and is now a three stage procedure. The first stage shifts the internal mesh points to produce an interim block that is determined entirely by the new locations of the 8 corner points defining the block. Corresponding to the motion of each corner point, each interior point is shifted by a displacement proportional to one minus the normalized distance along the index lines away from that corner point. The second stage corrects the perturbations resulting from the first stage by determining the distance each of the 12 edges of the stage 1 block needs to be moved to attain the desired edge locations. These perturbations are then also incorporated into the volume mesh points through a weighting scheme that is proportional to the relationship of an individual edge point motion and the volume point in question. Finally with both corner and edge point motion accounted for, the third stage checks the perturbation of each point in all six faces relative to the position of the corresponding point in the stage 2 block. If the perturbation of the domain involves a simple translation of all boundary points, the relative changes from stages 2 and 3 will be zero and all the perturbation will be accomplished by stage 1. If, however, face points are perturbed relative to the reference block, then these changes are propagated to the interior points through relative arc length-based perturbations along projecting index lines. In general all 3 stages are required. The idea of WARP3D is to use an initial mesh with good quality attributes as a starting point, and then systematically perturb this mesh in a manner such that the original grid quality is maintained, without the need for expensive elliptic smoothing.

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 entire method of creating a new mesh is given by the following algorithm.

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 (21).

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, a list of master edges and master corners is determined as a preprocessing step. During the design calculation, these lists are updated and transferred to all connected edges and corners as the mesh is moved.

Since this mesh perturbation algorithm is analytic it is possible to work out the analytical variations in the metric terms required for equation (20). This approach was followed in reference [40]. However since the mesh perturbation algorithm that was 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 minimal, finite differences were used to calculate δQ_{ij} instead of deriving the exact analytical relationships. Even in cases with hundreds of design variables, the computational cost of repeatedly re-evaluating δQ_{ij} for all necessary blocks is still insignificant compared with the cost of a single flow solution. The conclusion is that the analytical mesh perturbation algorithm, WARP3D, unlike an elliptical mesh generation method, is efficient to the extent that the cost of remeshing can be neglected.

It remains to choose a set of design variables which smoothly modifies the original shape, say b_i. The gradient can then be defined with respect to these design variables as

$$\mathcal{G}(b_i) = \frac{\delta I}{\delta b_i}, \tag{22}$$

where δI is calculated by (20) and each term b_i is independently perturbed by a finite step. Therefore, to construct $\mathcal{G}$, a basis space of independent perturbation functions b_i, $i = 1, 2, \ldots, n$ (n = number of design variables) must be chosen to allow for the needed freedom of the design space. In this work, design variables were chosen as a set of Hicks-Henne functions simply for their ease of implementation and their proven reliability. The form of the Hicks-Henne functions which were initially proposed in Reference [12] is given by:

$$b(x) = \left[\sin\left(\pi x^{\frac{\log(.5)}{\log(t_1)}}\right)\right]^{t_2}, \quad 0 \le x \le 1 \tag{23}$$

Here t_1 locates the maximum of the bump in the range $0 \le x \le 1$ at $x = t_1$, since the maximum occurs when $x^\alpha = \frac{1}{2}$, where $\alpha = \log\frac{1}{2}/\log t_1$, or $\alpha \log t_1 = \log\frac{1}{2}$. The parameter t_2 controls the width of the bump. To generalize the application of the Hicks-Henne functions, which have traditionally been used for the modification of airfoil sections where the x in equation (23) refers to the chordwise position, they are applied directly to a parameterized $(\tilde{u}, \tilde{v})$ surface which may be composed of one or more faces in different blocks. The parameterization may be accomplished in many ways. For this study, the wing is designed by projecting all surface points to be perturbed onto a plane and normalizing by the planform outline. Thus the Hicks-Henne shape functions may be applied as functions in either the $\tilde{u}$, $\tilde{v}$, or both directions. The design code is further structured so that these variables may be applied to any subset of the parametric surface. Alternatives are provided such that these variables may be linearly lofted in the second direction as opposed to Hicks-Henne functions in both directions. All of these options may be prescribed at the input level, leading to a highly versatile design code in which one or more faces in the multiblock domain may be perturbed by the design variables. To enforce geometric constraints, each design variable may be activated on more than one face. For example, if the thickness of a wing is to be preserved and the upper and lower halves of the wing are in separate blocks, then the design variables need to be applied at the proper locations with the proper weights and on the appropriate faces in both blocks such that thickness does not change while both surfaces are allowed to be modified.

DOMAIN DECOMPOSITION AND PARALLEL IMPLEMENTATION

The main strategies that are used to accomplish the parallelization of the design code are: a domain decomposition model, a SPMD (Single Program Multiple Data) strategy, and the MPI (Message Passing Interface) Library for message passing. The choice of message passing library was determined by the requirement that the resulting code be portable to different parallel computing platforms as well as to homogeneous and heterogeneous networks of workstations.

As one can see from the previous sections, obtaining the desired parallelization by domain decomposition entails the treatment of four separate parts: the solution of the flow equations, the solution of the adjoint equations, the calculation of the mesh perturbations, and the calculation of the gradient integral formulas. No attempt is made to parallelize the quasi-Newton optimization algorithm. It is thus assumed in this construct that the determination of the step sizes and search directions provided by the optimization algorithm is computationally insignificant when compared with the other elements necessary during the design.

Since the flow and adjoint equations are to be solved using exactly the same efficient numerical techniques, the same parallelization techniques used for the flow equations apply to the solution of the adjoint equations. Therefore, all details of the parallel implementation corresponding to these first two parts of the program will be explained with reference only to the flow equations. Furthermore, since the mesh perturbation algorithm WARP3D also works on a block-by-block basis, the communication necessary to maintain mesh consistency can also be addressed by the same domain decomposition strategies that are used for the state and costate fields.

The subdomains of the flow solution resident on each processor are divided along the block boundaries such that one or more complete blocks are allocated to each processor. This has the natural consequence that the communication between subdomains is performed through the same halo cells that were described earlier for the multiblock computations, only now, the interior cells corresponding to given halo cells might reside in a different processor. An alternative to this would be to partition the complete problem along the three coordinate directions for each of the blocks in the mesh. Since the sizes of the blocks can be quite small, this further partitioning would severely limit the number of multigrid levels that could be used in the flow and adjoint solutions. The underlying assumption is the fact that there always will be more blocks in the mesh than processors available. If this is the case, every processor in the domain would be responsible for the computations inside one or more blocks. In the case in which there are more processors than blocks available, the blocks can be adequately split during a pre-processing step in order to have at least as many blocks as processors. This approach has the advantage that the number of multigrid levels that can be used in the parallel implementation of the code is always the same as in the serial version, and is only limited by the size of the smallest block in

the mesh.

One advantage of the assignment of complete blocks to a given processor is that the number of processors in the calculation can be an arbitrary integer. The drawback of this approach is the loss of the exact load balancing that can be achieved by coordinate direction partitioning. All blocks in the calculation can have different sizes, and consequently, it is very likely that different processors will be assigned a different total number of cells in the calculation. This, in turn, will imply that some of the processors will be waiting until the processor with the largest number of cells has completed its work with the result being that the parallel performance will suffer. The approach that we have followed to solve the load balancing problem is to assign to each processor, in a pre-processing step, a certain number of blocks such that the total number of cells in each processor is as close as possible to the exact share for perfect load balancing. In practice, this approach yields quite good load balancing [24]. One must note that load balancing based on the total number of cells in each processor is only an approximation to the optimal solution of the problem. Other variables such as the number of blocks, the block locations, the size of each block, and the size of the buffers to be communicated play an important role in proper load balancing, and are the subject of current study.

Now, within each processor, there will be several blocks that need to communicate with their neighboring blocks. The data for these neighboring blocks can reside in a different processor, and therefore, communication is necessary. In order to minimize communication cost, it was decided to pack all data that needed to be communicated from one processor to another in one single message, regardless of the number of blocks that resided in each of the processors. Within each processor, the data for the flow variables, adjoint variables, and grid locations are stored on a large one-dimensional array. In order to accomplish this type of communication, during the pre-processing step, each processor compiles a pointer list with all the entries in these large arrays that need to be sent to all other processors. Similarly, another pointer list for the locations of the data to be received is also setup. At the time of communication exchanges each processor communicates all the information for the blocks that it contains to those processors that need to receive it. The communication is implemented using asynchronous (non-blocking) send and receive MPI constructs.

The final formulas for the gradient of the cost function include a number of volume and surface integrals (see [22]) that can also be calculated in parallel. The nature of the integrations is such that, from a domain decomposition point of view, the problem is nearly embarrassingly parallel, since the partial values of the integrals in each domain can be calculated without the need for any communication at all. Once the individual processors have completed their calculation, a single global MPI reduce operation can combine the results into the total gradients, which can then be passed to the numerical optimization algorithm to develop the shape change at each design iteration.

COMPLETE MULTIBLOCK DESIGN ALGORITHM (SYN87-MB

With all the necessary components defined for the multiblock adjoint-based design, it is now possible to outline the complete procedure:

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 (5-10).
3. Solve the adjoint equations (17) subject to the boundary condition (19).
4. For each of the n design variables repeat the following:
   - Perturb the design variable by a finite step according to (23, etc.).
   - Explicitly perturb all faces affected by the design variable.
   - Implicitly perturb all faces that share an edge with an explicitly perturbed design variable.
   - Obtain the new 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 by finite differencing.
   - Integrate equation (20) to obtain δI for those blocks that contain nonzero $\delta Q_{i,j}$.
   - Determine the gradient component by equation (22).
5. Calculate the search direction and perform a line search.
6. Return to (2) if minimum has not been reached.

The basic method here builds on that used in reference [40] with the proper extensions to treat multiblock domains. In order to implement the method, equation (17) and boundary condition (19) must be discretized on the multiblock domain. In the current implementation, a cell centered, central difference stencil that mimics the flux balancing used for the flow solution is used. Since this choice of discretization differs from the one obtained if the discrete flow equation Jacobian matrix were actually transposed to form the adjoint system, the gradients obtained by the present method will not be exactly equal to the gradients calculated by finite differencing the discrete flow solutions. However, as the mesh is refined these differences should vanish. Continuing, the adjoint system so discretized is solved on the multiblock domain in a fashion identical to that used for the flow solution. Therefore, the adjoint solver, like the flow solver, uses an explicit multistage Runge-Kutta-like algorithm accelerated by residual smoothing and multigridding. Intra-block communication is again handled through a double halo which allows for the full transfer of information across boundaries except for the stencil of support for the implicit residual smoothing.

Step (3) in the above procedure is the portion of the method that is still treated by finite differences. Fortunately, all of these steps incur only a trivial computational cost compared with even a single flow analysis time step. It is therefore possible, without significant penalty, to leave this in finite difference form even for cases where many hundreds of design variables are used.

The present implementation uses the quasi-Newton algorithm, QNMDIF, developed by Gill, Murray and Pitfield [10] and enhanced by Kennelly [27], to calculate the search direction. It is an unconstrained optimization algorithm that uses Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates to the Cholesky factored Hessian matrix. A complete treatment of the quasi-Newton and other optimization strategies is given by Gill, Murray and Wright [11].

PARALLEL EFFICIENCY

For problems with a low task granularity (ratio of the number of bytes received by a processor to the number of floating point operations it performs), large parallel efficiencies can be obtained. Unfortunately, convergence acceleration techniques developed in the 1980s base their success on global communication in the computational domain. Thus, current multigrid and implicit residual smoothing techniques could possibly hinder parallel performance in traditional mesh sizes. However, for larger meshes used in viscous flows and complex geometries, the granularity becomes low enough, and the parallel performance is quite high.

A careful implementation of the communication schedules can yield a design method with excellent parallel performance. The parallel efficiency of the design method is presented in Figure 2 for the case of the automated redesign of a typical business jet configuration. In the modeling of the aircraft, a wing-body-nacelle topology was used containing 72 blocks of varying sizes with a total of 750,000 cells. It must be noted that the parallel speed-ups below are computed using true execution wall-times. The execution time of the one processor job is taken as that of the most efficient serial implementation of the method. In addition, the complete execution of the design problem (for parallel performance calculations) was taken to be that of 2 complete design iterations, which include several flow and adjoint solutions. Finally, I/O time was also included in the performance calculations. As one can see, for a problem of moderate size such as this, excellent speed-ups are obtained using up to 16 processors. The speed-up for the 32 processor calculation shows a breaking trend due to the impossibility of achieving good load balancing by partitioning a 72 block mesh into 32 processors. The optimum solution for this problem shows a spread in the load balancing of almost 10%. This, together with the fact that the serial parts of the algorithm play a large role for problems with short execution times (large numbers of processors), accounts for the decrease in parallel efficiency.

Just as important as the speed-up curves of the method is the true wall-clock execution time. Two complete design iterations were completed in under 35 minutes using 32 processors. The rapid response of the design method clearly allows the possibility of investigating a large number of configurations, and increases the probability of arriving at a truly optimum design. Timings for the 1,500,000-cell HSCT mesh used in the Results sections can be readily extrapolated from the ratio of the numbers of cells in the two meshes.

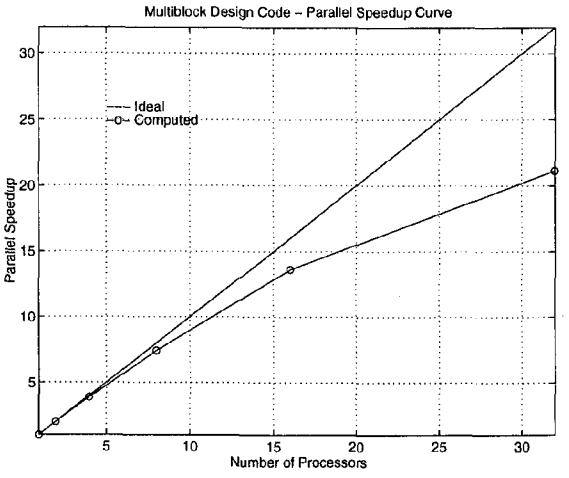


Figure 2: Parallel Speed-ups for the Adjoint Design Method

NUMERICAL TESTS AND RESULTS

To demonstrate the utility of the design method, a supersonic transport configuration will be used as a testbed for the optimization algorithm. The baseline supersonic transport configuration depicted in Figures 4—6 was sized to accommodate 300 passengers with a gross take-off weight of 750,000 lbs. The supersonic cruise point is Mach 2.2 with a C_L of 0.105. As can be seen in Figure 6, the planform has a break in the leading edge sweep. The inboard leading edge sweep is 68.5 degrees while the outboard is 49.5 degrees. Since the Mach angle at M = 2.2 is 63 degrees it is clear that some leading edge bluntness may be used inboard without a significant wave drag penalty. Airfoils which use blunt leading edges were created that range from 4% thick at the root to 2.5% thick at the leading edge break point. The symmetric initial airfoils were chosen with the purpose of accommodating thick spars at roughly 30% and 80% chord over the span up to the leading edge break. Outboard of the leading edge break where the wing sweep is ahead of the Mach cone, a sharp leading edge was used to avoid undue wave drag. The airfoils were chosen to be symmetric, biconvex shapes modified to have a region of constant thickness over the mid-chord. Figure 5 shows that the four-engine configuration features axisymmetric nacelles tucked close to the wing lower surface. This layout favors reduced wave drag by minimizing the exposed diverter area. However, it may be problematic because of the channel flows occurring in the juncture region of the diverter, wing, and nacelle at the wing trailing edge. The leading edge heights of the diverters are determined by the boundary layer local displacement thickness such that entrainment of boundary layer flow into the engines is avoided. Since the distance from the wing leading edge to the diverter leading edge is different for the two nacelle, this causes a corresponding diverter height difference.

In this design problem, we have chosen to attempt the straightforward task of recambering the wing so as to minimize the total configuration wave drag while maintaining fixed C_L and Mach number. This represents a much simpler problem than allowing both thickness and camber to vary while maintaining spar constraints and adequate fuel volume. However, since the complex nacelle geometry and resulting meshes on the wing lower surface must be moved in unison with the wing motion, even a camber-only design represents a significant challenge for demonstration purposes. Further optimization benefits could also be attained not only by allowing thickness variations in the wing optimization but also by performing both body optimization and nacelle orientation and position optimization. Such studies will be made in future work.

Returning to the wing camber design at hand, the mesh system shown in Figure 4 was created with 180 blocks and 1,533,440 computational cells including halos. While the point-to-point topology forces a larger number of blocks to be used, this is actually beneficial from the standpoint of parallel load balancing. Ninety design

variables of the Hicks-Henne type are lofted in both the spanwise and chordwise direction. These are spread over the entire wing such that complete freedom is allowed with the exception of the wing root intersection and the wing trailing edge. The former was held fixed since the capability of reintersecting different geometry elements has yet to be implemented. The latter was held fixed since it was determined that freedom in the trailing edge would likely produce waves in the spanwise direction along the trailing edge which would violate possible flap constraints. Figure 6 shows the solution on the upper and lower surface for the baseline configuration at the cruise point. The nacelles and diverters were forced to follow the variations that were imposed on the wing. This allowed the nacelles to be included in the optimization without approximations. The rest of the complex mesh system in and around the nacelles shown in Figure 4 was perturbed automatically as the design proceeded.

Due to a shortage of both resources and time, only a two-design-cycle run of the optimization algorithm was achieved using 16 processors on an IBM SP2. This must be thought of as a very preliminary result. It nevertheless demonstrates the power of a parallel implementation of the adjoint-based design method. The total wall clock time was 1.75 hours for the entire run. Figure 7 shows the final solution at the end of the second design iteration for both the upper and lower surfaces. It is clear that large differences have been made to the upper surface pressure field. The comparisons of the cross-sectional cuts of both the airfoils and the Cps shown in Figure 8 are more indicative of the design differences. The strong oblique shock that ran near the leading edge has been largely softened. Meanwhile, the lower surface regions around the nacelles have been tuned so as to cusp the region where the nacelle shocks impinge on the wing lower surface. This creates a forward facing region with high Cps, thus reducing wave drag. The final overall pressure drag was reduced by 3.5% or from $C_D = 0.00882$ to $C_D = 0.00851$. It is confidently believed that much more could be gained through the investment of more computer resources.

CONCLUSIONS

In the period since this approach to optimal shape design was first proposed by the fourth author [19], the method has been verified by numerical implementation for both potential flow and flows modeled by the Euler equations [20, 39, 25, 23]. It has been demonstrated that it can be successfully used with a finite volume formulation to perform calculations with arbitrary numerically generated grids [39, 25]. Further, results have been presented for three-dimensional calculations using both the analytic mapping and general finite volume implementations [40]. In the last year the technique has been adopted by some industry participants to perform the aerodynamic design of future configurations [9]. With the parallel implementation of the multiblock flow solver presented here, the technology has advanced to the degree that aerodynamic shape design of complete aircraft configurations with very rapid turnaround is possible.

In this paper we have shown how the complicated design of supersonic aircraft configurations including airframe/nacelle interaction effects can be accomplished in a routine fashion with the current method. For demonstration purposes a typical supersonic configuration with closely coupled nacelles and diverters was used as a test case. Nacelles of the axisymmetric type were employed and situated such that they almost abutted the wing lower surface. The diverters filling the space between the nacelles and the wing lower surface were constructed to eliminate the possibility of boundary layer entrainment into the nacelle inlets. These geometry entities as well as the complete wing and body were fully modeled using the new multiblock analysis and design method. A preliminary camber-only design was carried out to optimize the cruise point performance for the complete aircraft configuration subject to wing thickness constraints and fixed lift. All analysis and design cases were performed on parallel architecture machines in less than one day, demonstrating that complete configuration designs may be achieved with rapid turn-around even with the most conservative estimates of available computational resources. In future efforts, the techniques will be extended to address both unstructured meshes and flows governed by the Navier-Stokes equations.

ACKNOWLEDGMENTS

This research has benefited greatly from the generous support of the AFOSR under grant number AFOSR-91-0391, ARPA under grant number N00014-92-J-1976, USRA through RIACS, the High Speed Research branch of NASA Ames Research Center, and IBM. Considerable thanks also goes to David Saunders of Sterling Software for his help in assembling the text.

References

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

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

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

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] P. Gill, W. Murray, and R. Pitfield. The implementation of two revised quasi-Newton algorithms for unconstrained optimization. *NAC 11*, National Physical Laboratory, Division of Numerical Analysis and Computing, 1972.

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

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

[13] 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.

[14] J. Huan and V. Modi. Design of minimum drag bodies in incompressible laminar flow. Technical report, The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, California, November 1995.

[15] W.P. Huffman, R.G. Melvin, D.P. Young, F.T. Johnson, J.E. Bussoletti, M.B. Bieterman, and C.L. Hilmes. Practical design and optimization in computational fluid dynamics. *AIAA paper 93-3111*, AIAA 24th Fluid Dynamics Conference, Orlando, Florida, July 1993.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] R. Kennelly. Improved method for transonic airfoil design-by-optimization. *AIAA paper 83-1864*, AIAA Applied Aerodynamics Conference, Danvers, Massachusetts, July 1983.

[28] 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.

[29] 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.

[30] 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.

[31] G. Kuruvila, S. Ta'asan, and M. D. Salas. Airfoil optimization by the one-shot method. In *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics*. von Karman Institute for Fluid Dynamics, 1994.

[32] 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.

[33] J. Lewis and R. Agarwal. Airfoil design via control theory using the full-potential and Euler equations. Technical report, The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, California, November 1995.

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

[35] 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.

[36] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer-Verlag, New York, 1984.

[37] O. Pironneau. Optimal shape design for aerodynamics. In *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics*. von Karman Institute for Fluid Dynamics, 1994.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] S. Ta'asan, G. Kuruvila, and M. D. Salas. Aerodynamic design and optimization in one shot. *AIAA paper 92-0025*, 30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1992.

[48] J.F. Thompson, Z.U.A Warsi, and C.W. Mastin. *Numerical Grid Generation, Foundations and Applications*. Elsevier Science Publishing Company, New York, NY, 1985.

[49] D.P. Young, W.P. Huffman, R.G. Melvin, M.B. Bieterman, C.L. Hilmes, and F.T. Johnson. Inexactness and global convergence in design optimization. *AIAA paper 94-4286*, 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 1994.

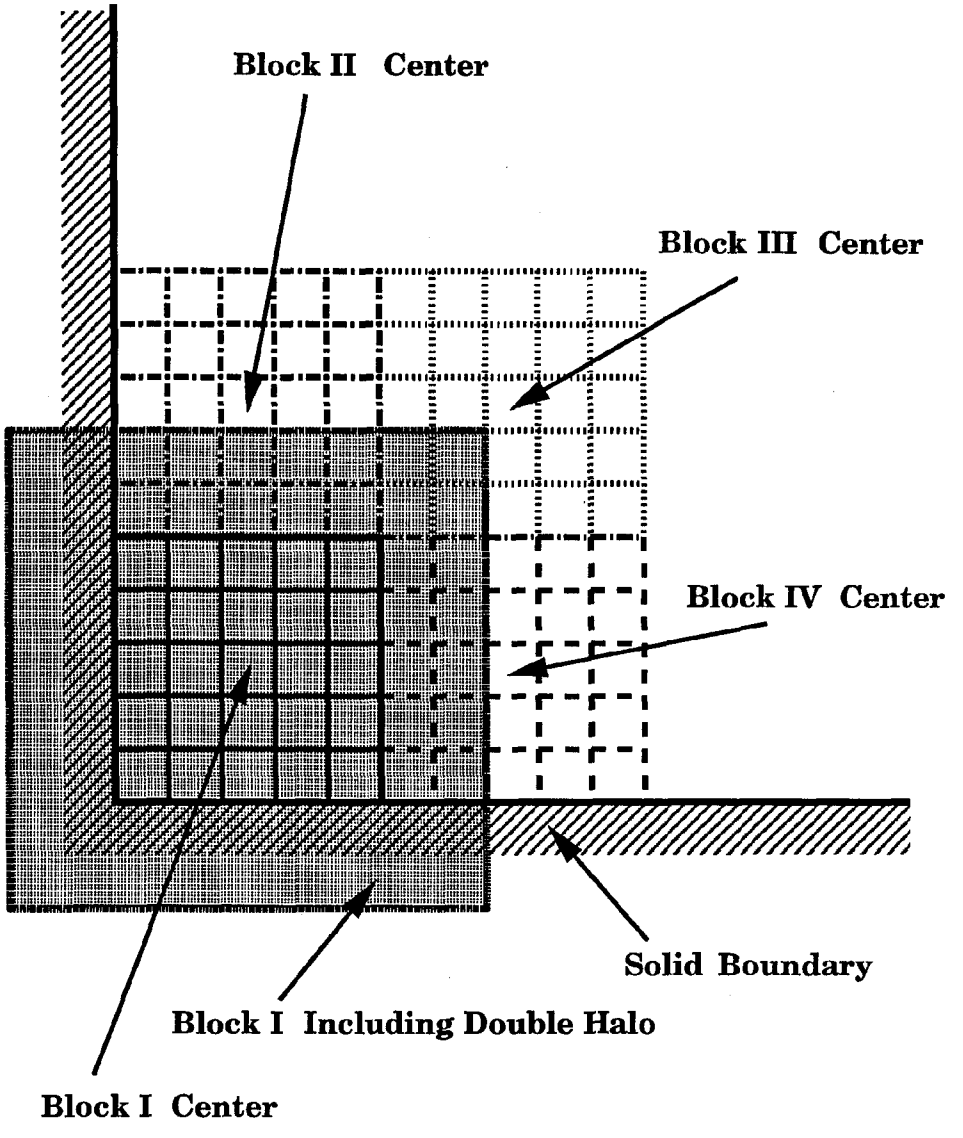


Figure 3: 4 Block interface using a double halo of cells around each block. Each block's double halo of cells contains information from internal cells in adjacent blocks.

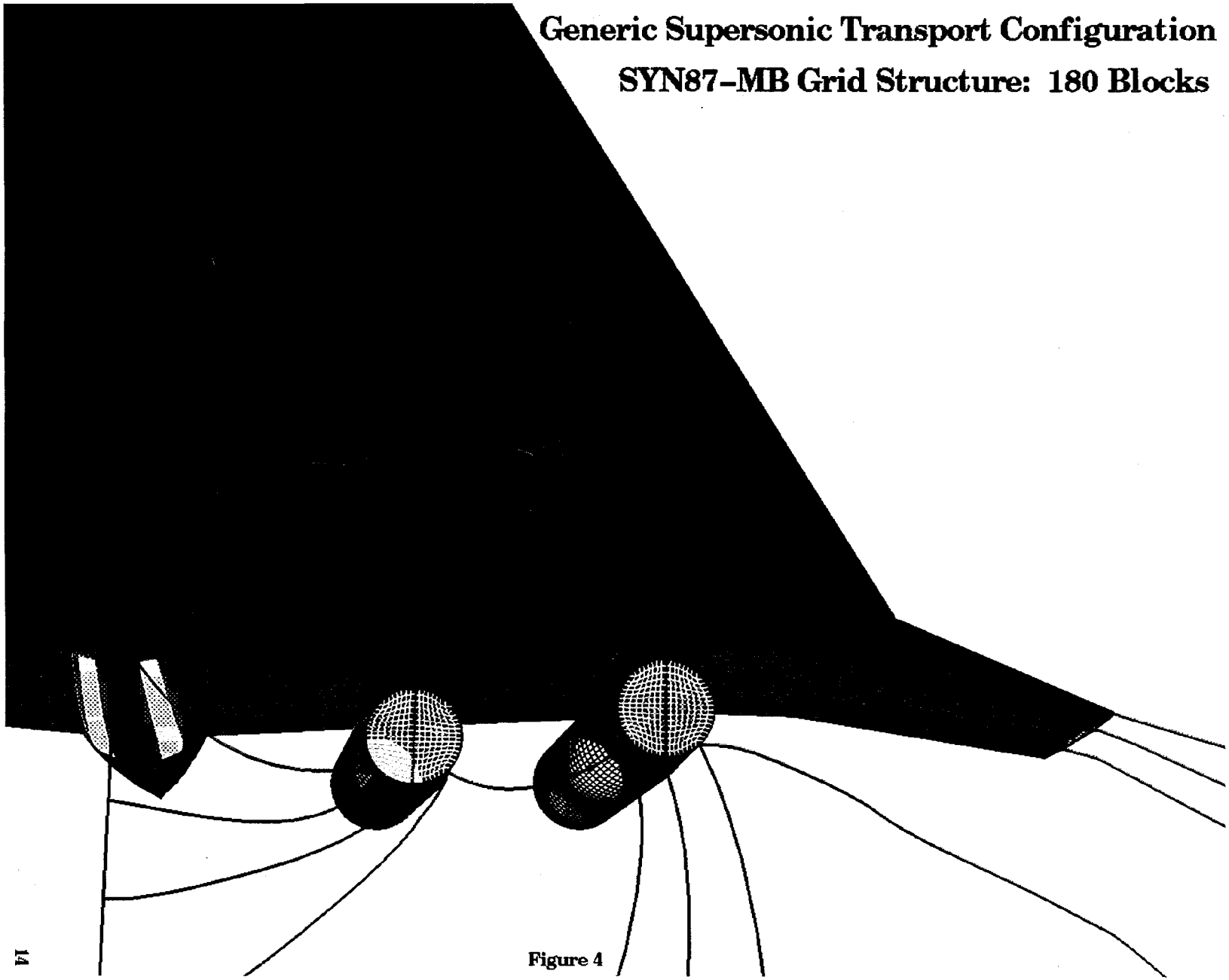


Figure 4

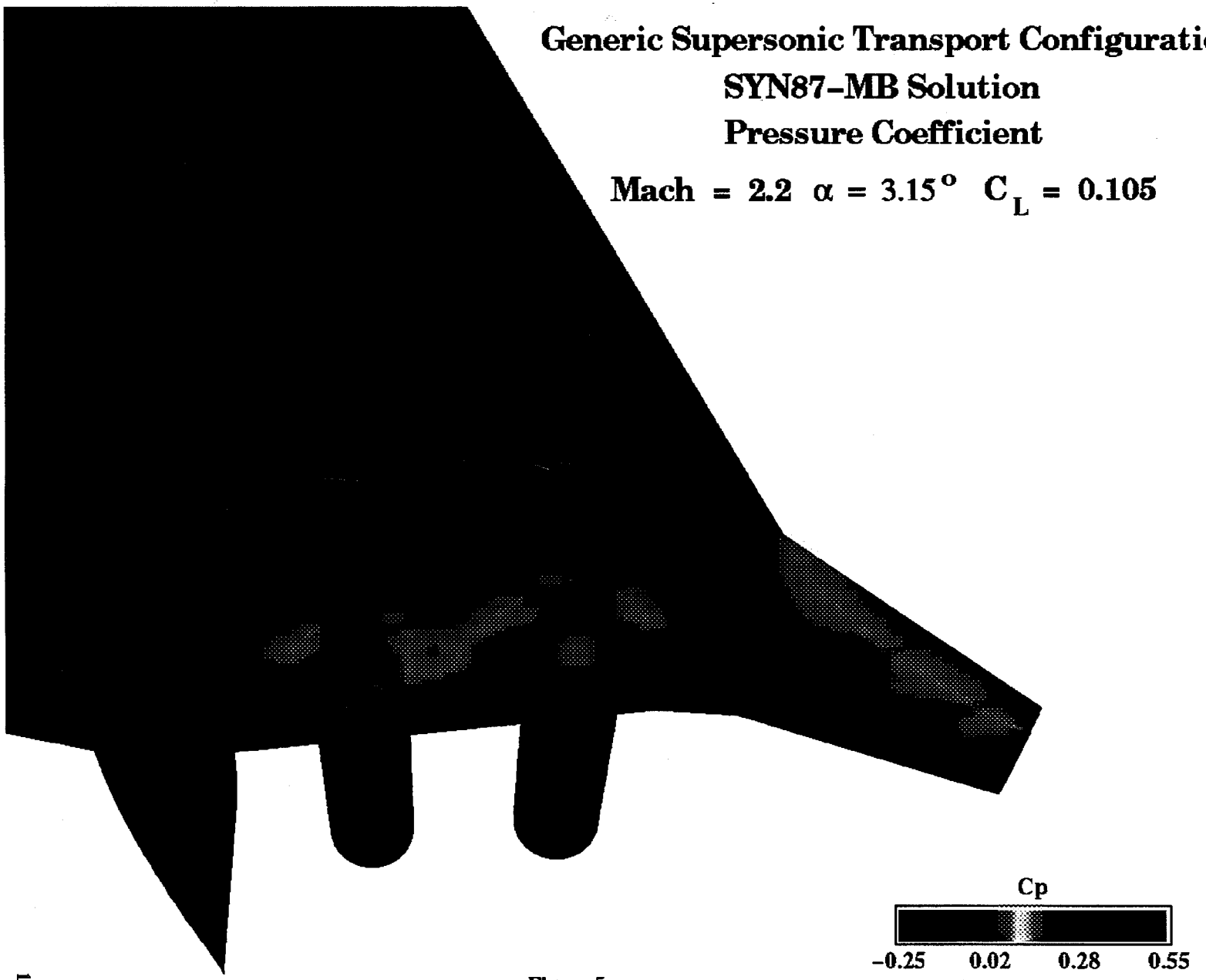


Figure 5

Generic Supersonic Transport Configuration
SYN87-MB Solution
Pressure Coefficient
Mach = 2.2 $\alpha = 3.15^{o}$ $C_L = 0.105$

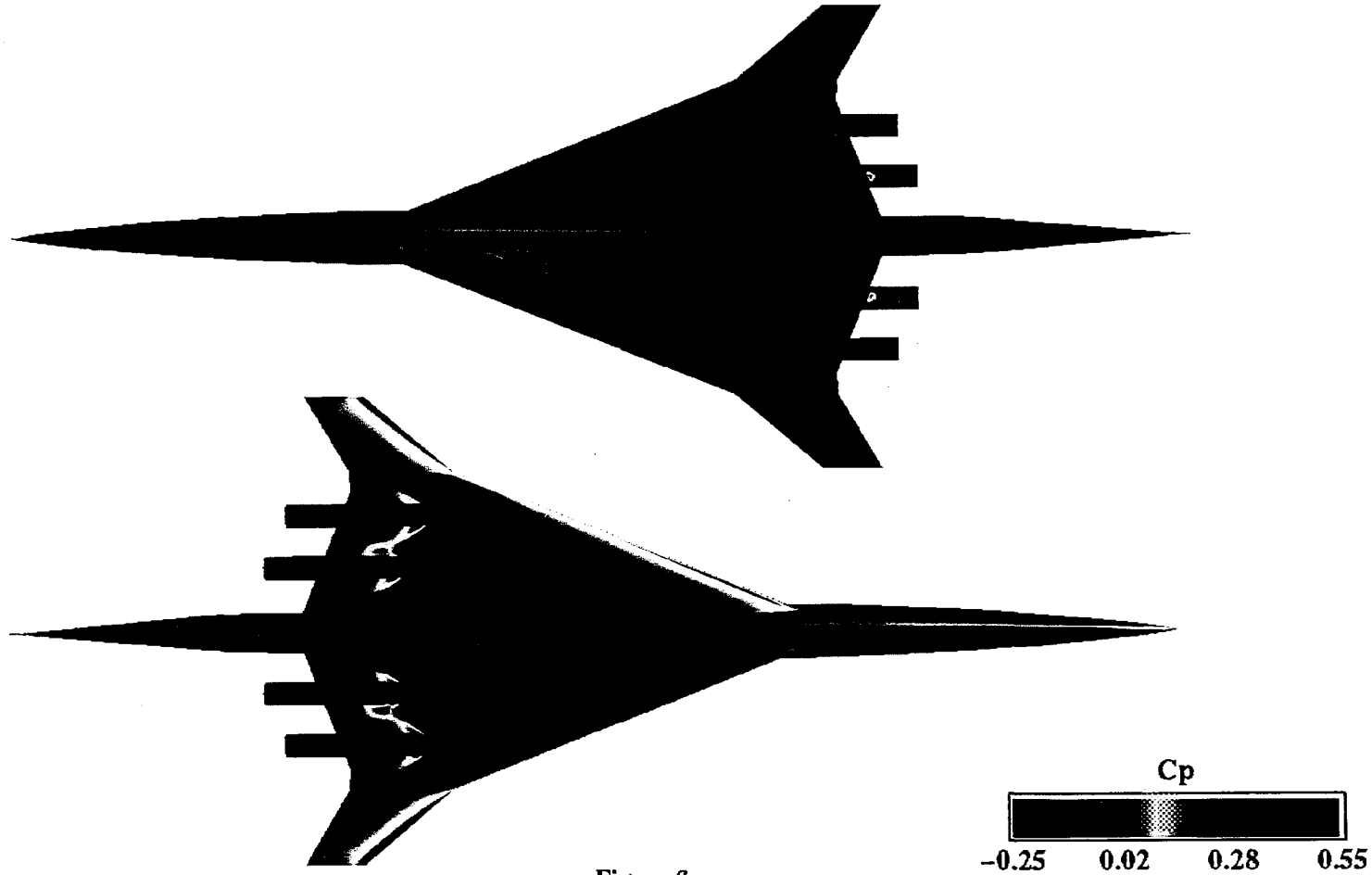


Figure 6

Optimized Supersonic Transport Configuration
SYN87-MB Solution
Pressure Coefficient
Mach = 2.2 $\alpha = 3.15^{o}$ $C_L = 0.105$

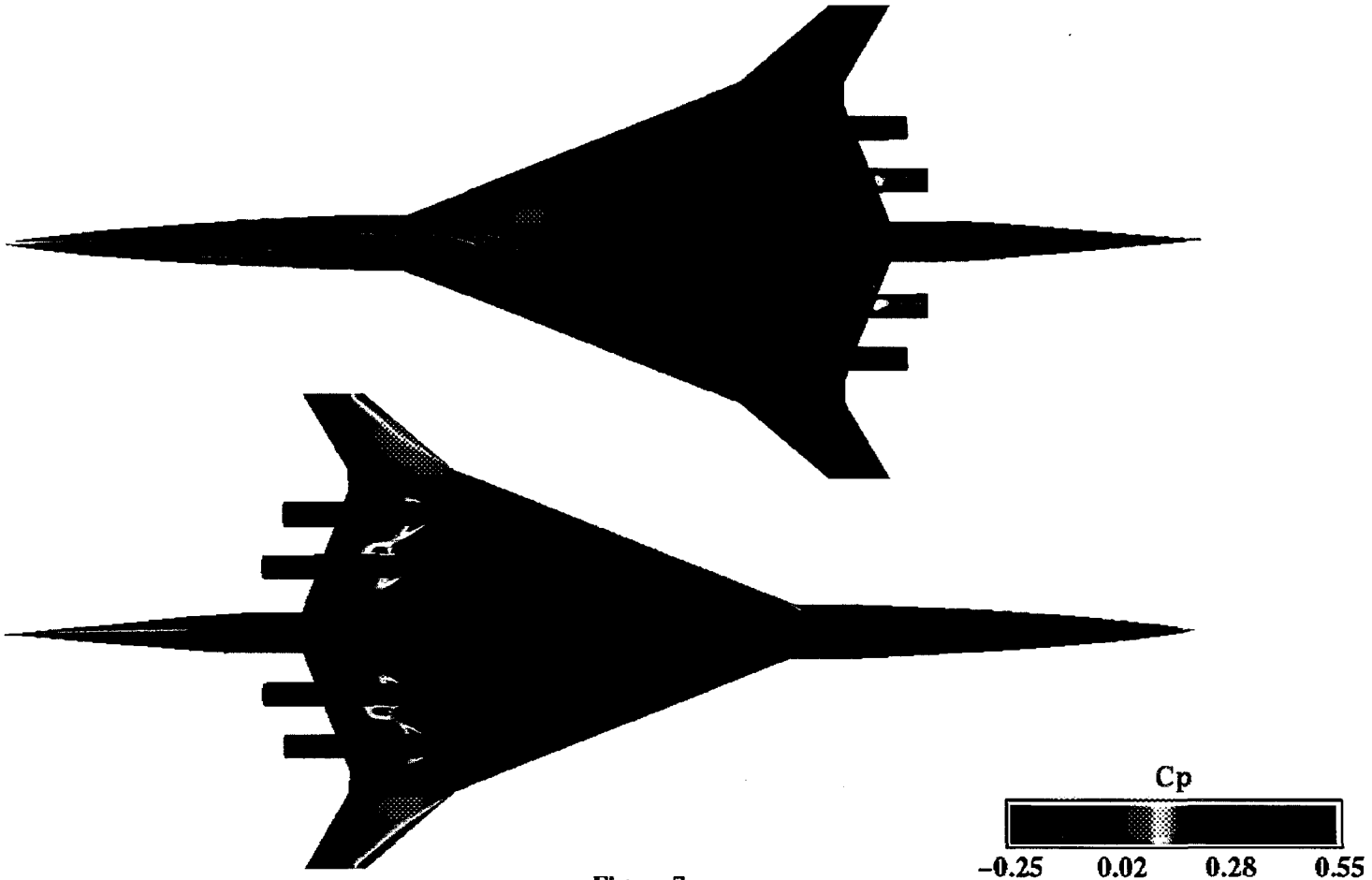
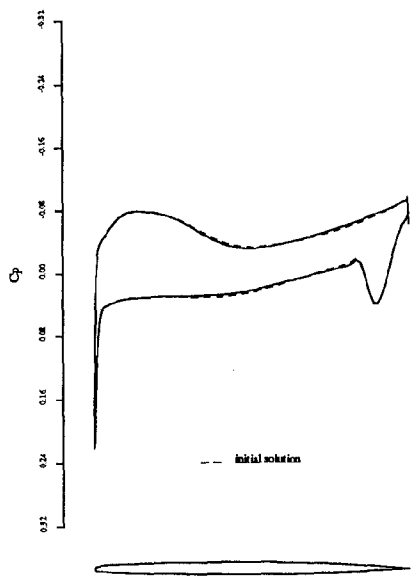
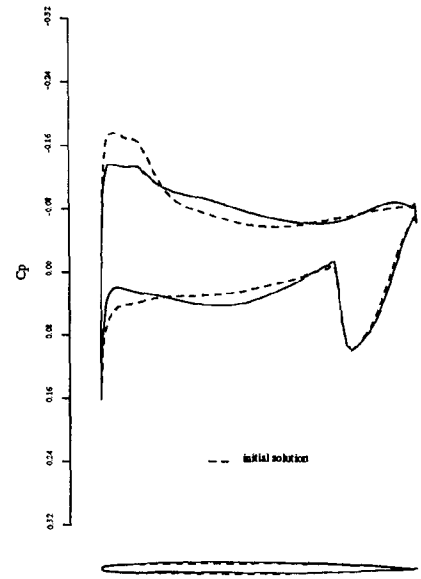


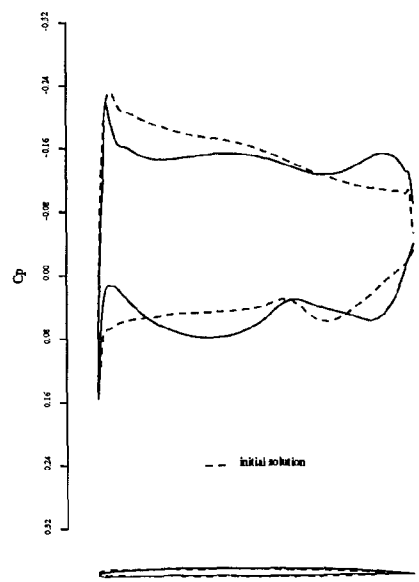
Figure 7



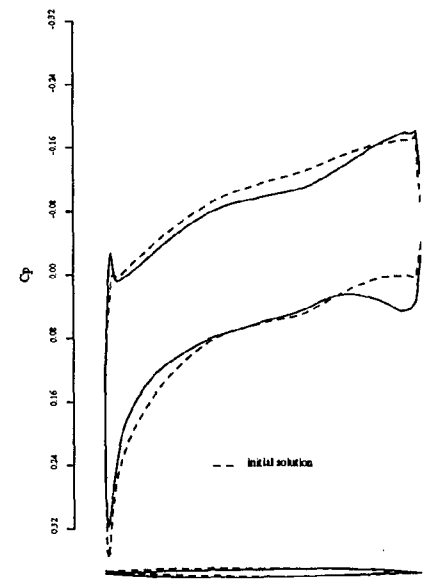
8a: span station $\eta = 0.116$



8b: span station $\eta = 0.374$



8c: span station $z = 0.632$



8d: span station $z = 0.871$

Figure 8: SYN87-MB Fixed Lift Drag Minimization.
180-Block Mesh, 1,500 K mesh cells, $M = 2.2$
90 Camber Hicks-Henne variables.
- - - - -, Initial Pressure Coefficient
—, Final Pressure Coefficient.

Figure 4

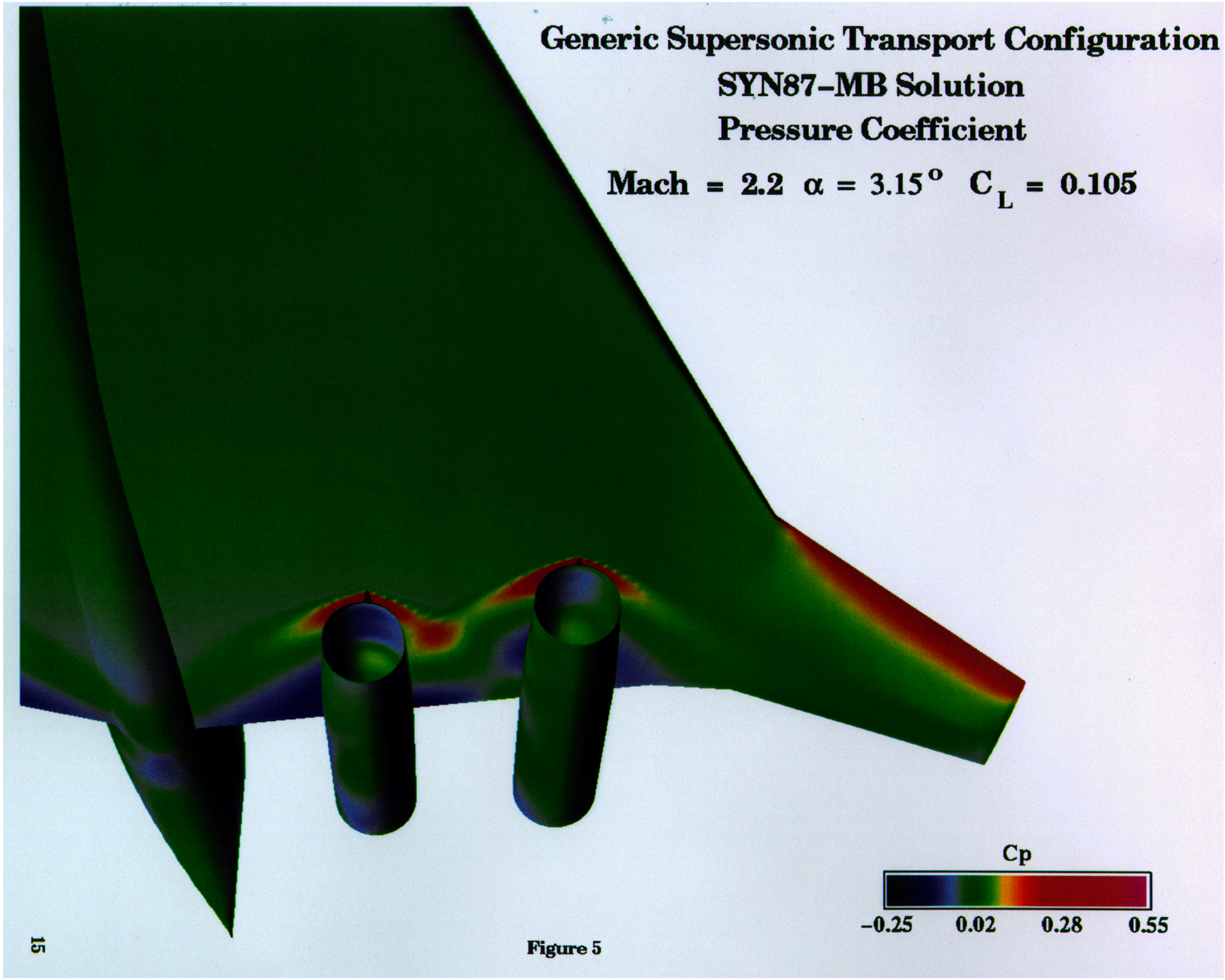


Figure 5

Generic Supersonic Transport Configuration

SYN87-MB Solution

Pressure Coefficient

Mach = 2.2 $\alpha = 3.15^{o}$ $C_L = 0.105$

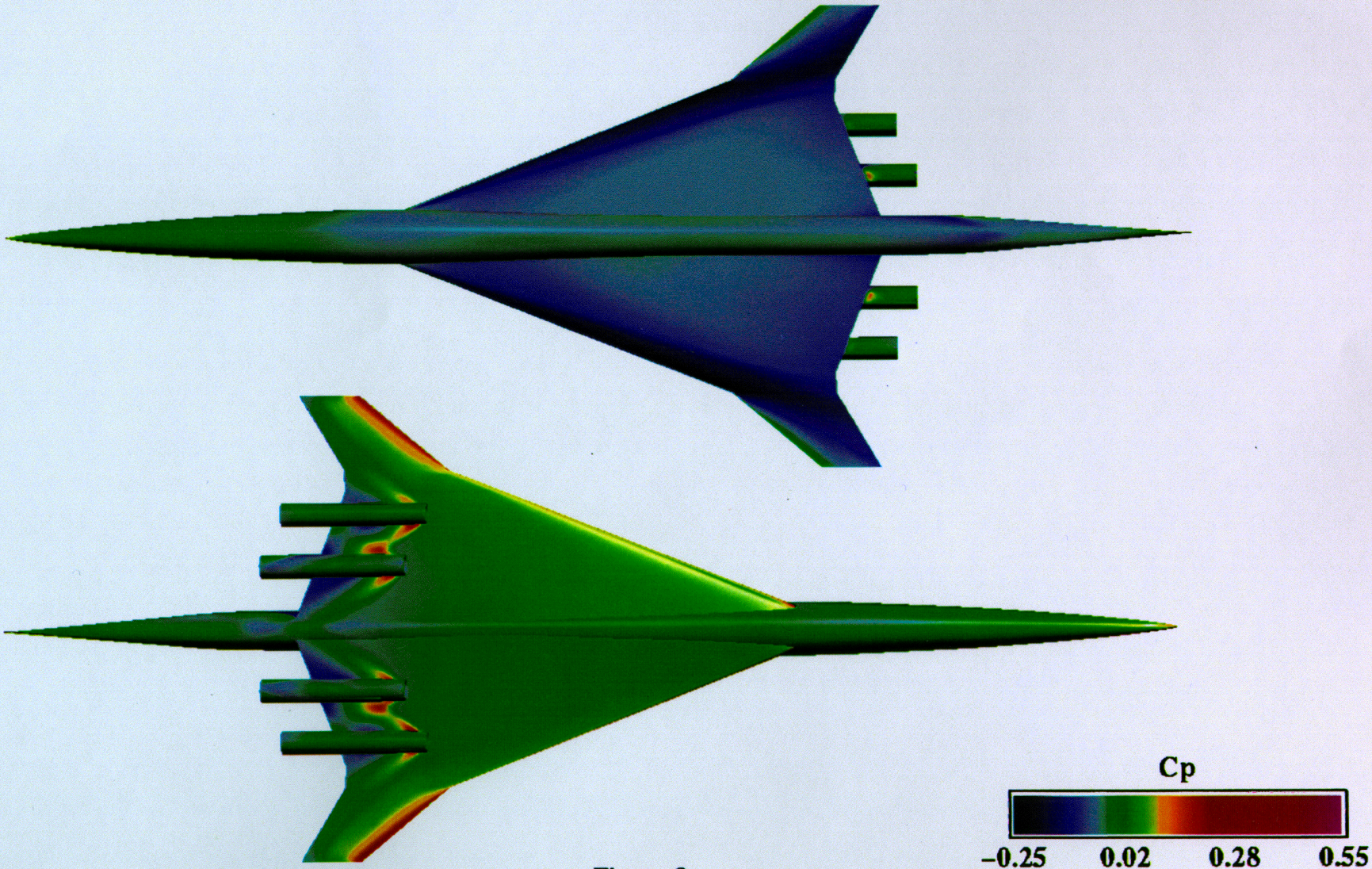


Figure 6

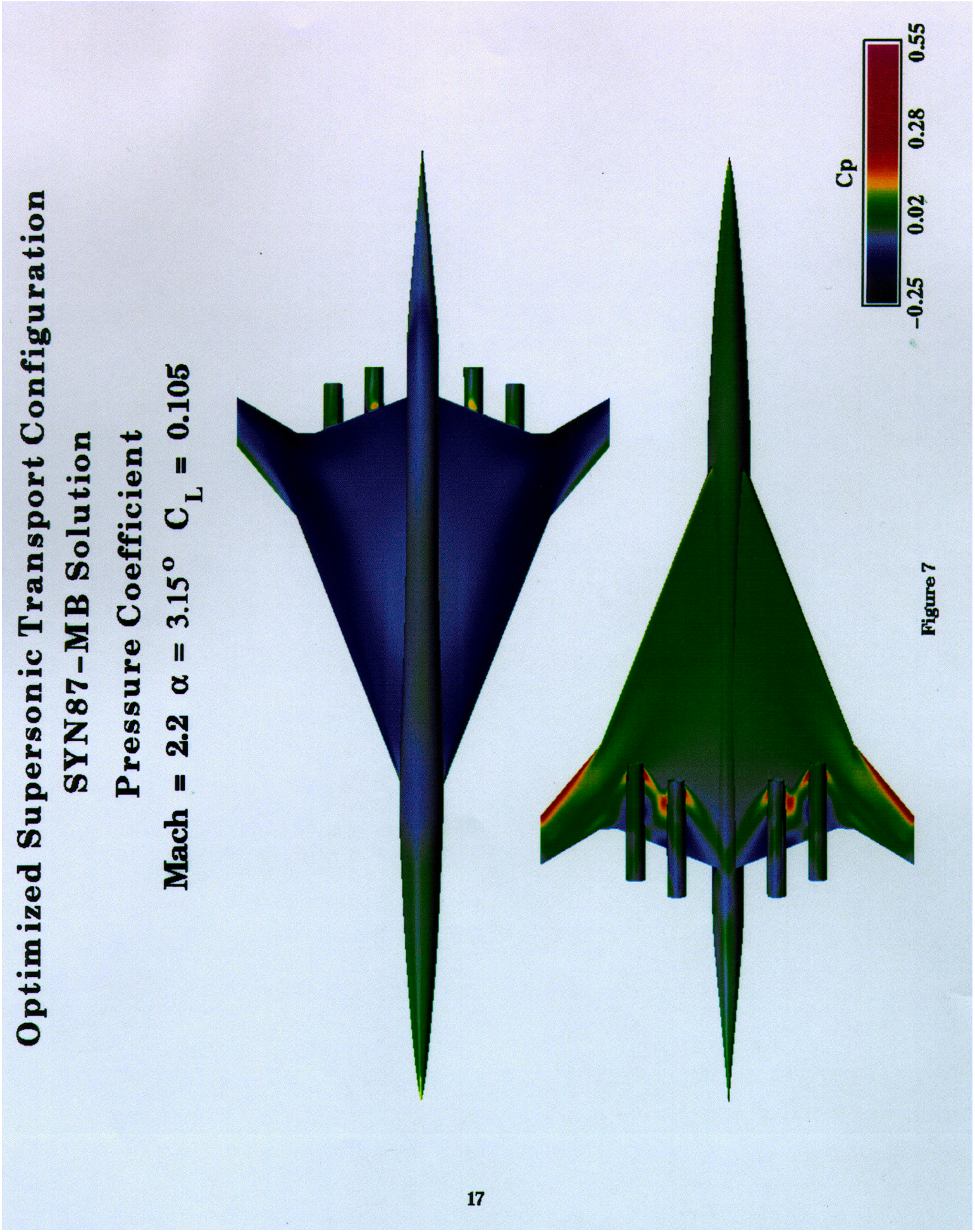


Figure 7