Aerodynamic Shape Optimization Using A Real-Number-Encoded Genetic Algorithm

Terry L. Holst and Thomas H. Pulliam
Numerical Aerospace Simulation Systems Division
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

A new method for aerodynamic shape optimization using a genetic algorithm with real number encoding is presented. The algorithm is used to optimize three different problems, a simple hill climbing problem, a quasi-one-dimensional nozzle problem using an Euler equation solver and a three-dimensional transonic wing problem using a nonlinear potential solver. Results indicate that the genetic algorithm is easy to implement and extremely reliable, being relatively insensitive to design space noise.

Background

Numerical methods for optimizing aerodynamic performance have been studied for many years. Most approaches can be classified into one of three general categories: 1) inverse methods, 2) gradient-based methods and 3) genetic algorithms. The inverse method in aerodynamic design seeks to determine the aerodynamic shape for a specified surface pressure distribution, i.e., the "inverse" of the normal analysis approach. An advantage of this approach is that it offers direct control over aerodynamic forces and moments (through specification of the surface pressure). In addition, by utilizing proper constraints on the adverse pressure gradient, a degree of control on boundary layer separation is also available, even for inviscid implementations. The biggest difficulty of the inverse design method is selecting a pressure distribution that will achieve the best aerodynamic performance for a given set of constraints. Clearly, experience helps in this area, but knowing what is good aerodynamically and (at the same time) does not over constrain the inverse method and prevent convergence can be a problem.

Numerical optimization using gradient-based methods has received much attention in recent years. The reliability and success of gradient methods is based on and requires a smooth design space and the existence of only a single global extremum. A good review of gradient methods used in aerodynamic design is presented by Reuther.[1] The general idea associated with this broad class of methods consists of the following steps. First, determine the optimization's objective, e.g., minimization of the drag-to-lift ratio or minimization of the least-square error between the actual and a prescribed pressure distribution, etc. Second, the geometry to be optimized must be parameterized. This parameterization must completely describe the geometry (or the portion that is to be optimized) and must lend itself to discrete variations that can be independently modified. In many (but not all) gradient method implementations it is advantageous to parameterize the geometry with the minimum number of parameters that will still completely describe the applicable design space. Examples of aerodynamic shape parameterizations are given in Hicks and Henne[2] where a series of "bump" functions is used or Burgreen and Baysal[3] where a series of B-spline control points is used. The third step is to compute the direction in the design space (away from a specified initial condition) that minimizes the objective using a steepest descent, conjugate gradient, Newton or quasi-Newton approach (see Luenberger[4] or Reuther[1] for general details in this area).

The technique for determining sensitivity derivatives is a key item in any gradient-based design approach and has received much attention. The simplest approach, often called the "brute-force" or finite-difference method, consists of using CFD flow solver solutions to determine the effect of each design perturbation on the objective function. Sensitivities are then constructed using finite-difference formulas. If a design problem using the above approach has K decision variables, the sensitivity derivative computation for each design iteration will require $K+1$ function evaluations, specifically, one CFD solution for the unperturbed or baseline geometry and K solutions corresponding to the K decision variable perturbations. After the sensitivity derivatives have been computed and the steepest descent direction is determined, a "line search" is required to determine values for the step size vector, which in turn requires a number of additional CFD function calls. Overall, gradient-based methods for typical aerodynamic optimization problems require from several iterations to several tens of iterations to converge (depending on the method used and the number of decision variables). Thus, the total number of CFD solutions required for this type of optimization approach can easily number in the hundreds for a single design.

Other methods for evaluating sensitivity derivatives that seek to reduce the large computational cost associated with the finite-difference method have been developed, e.g., the ADIFOR (Automatic Differentiation of Fortran) approach,[5] the quasi-analytic method of El-banna and Carlson[6,7] and Arslan and Carlson[8] and the adjoint method of Jameson.[9-10] In the latter approach, an adjoint equation, derived from control theory, is solved to determine design space sensitivity derivatives. This approach is superior to the finite-difference approach for generating sensitivities because all the sensitivities are obtained (no matter how many there are) by solving one adjoint equation, with a cost on the order of one flow field solution. Thus, for design problems containing a large number of decision variables, the cost savings for this approach over the finite-difference approach is significant.

The last optimization method discussed is called the genetic algorithm (GA) approach and is the method used in the present paper. The basic idea associated with this approach is to search for optimal solutions using the theory of evolution. During solution iteration (or "evolution" using GA terminology) the decision variables or "genes" are manipulated using various operators (selection, combination, crossover, mutation) to create new design populations, i.e., new sets of decision variables. General GA details can be found in Goldberg,[11] Davis[12] and Beasley, et al.[13] Each design is evaluated using an objective-like "biological fitness function" to determine survivability. Constraints can easily be included in this approach. If a design violates a constraint, its fitness function is set to zero, i.e., it doesn't survive to the next evolution level. Because GA optimization is not a gradient-based optimization technique, it does not need sensitivity derivatives. It theoretically works well in non-smooth design spaces containing several or perhaps many local extrema. It is also an attractive method for multi-point design applications. A disadvantage of the GA approach is expense. In general, the number of function evaluations required for a GA algorithm exceeds the number required by a finite-difference-based gradient optimization (see the results presented in Obayashi and Tsukahara[14] and Bock[15]). Example applications utilizing potential-based flow solvers in the context of GA optimization can be found in Quagliarella and Della Cioppa[16] for airfoil applications, Vicini and Quagliarella[17] for multi-point and multi-objective airfoil applications and Obayashi *et al.*[18] for multi-disciplinary optimization of transonic wings. Examples of multi-design-point wing optimization using Euler and Naver/Stokes flow solvers can be found in Sasaki, et al.[19] and Oyama.[20,21]

GA Algorithm

The GA used in the present study to perform all optimizations is described in this section. As mentioned above the general idea behind any GA optimization is to discretely describe the optimization space using a number of genes, x_i^n. In this notation the i-subscript is the gene number and the superscript corresponds to the generation. Each set of genes that leads to the complete specification of an individual (or an individual design) is called a chromosome, which is given by

$$\mathbf{X}_j^n(x_{1,j}^n, x_{2,j}^n, \cdots, x_{i,j}^n, \cdots, x_{NG,j}^n) \tag{1}$$

In this notation $\mathbf{X}_j^n$ is the jth chromosome for the nth generation that consists of NG genes. A second subscript j has been added to each gene to show which chromosome it is identified with.

For aerodynamic shape optimization the set of genes associated with a single chromosome must fully describe the geometry being modified. In many GA applications genes are computationally represented using bit strings and the operators used are designed to manipulate bit string data. In the present approach, following the arguments of Oyama,[21] Houck, et al.[22] andMichalewicz,[23] real-number encoding is used to represent all genes. The key reason for using real number encoding is that they have been shown to be more efficient in terms of CPU time relative to binary encoded GAs.[23] In addition, real numbers are used for all parameters defining an aerodynamic surface or shape, e.g., sweep, twist, thickness, camber. Thus, using a real number encoding eliminates the need for binary-real number conversions.

Once the design space has been defined in terms of a set of real-number genes, the next step is to form an initial generation, $\mathbf{G}^0$, which is represented by

$$\mathbf{G}^0 = (\mathbf{X}_1^0, \mathbf{X}_2^0, \cdots, \mathbf{X}_j^0, \cdots, \mathbf{X}_{NC}^0) \tag{2}$$

where NC is the total number of chromosomes. Each gene within each chromosome is assigned an initial numerical value using a process that randomly chooses numbers between fixed user-specified limits. For example, the tenth gene in an arbitrary chromosome is computed using

$$x_{10} = R(0,1)(x\max_{10} - x\min_{10}) + x\min_{10}$$

where $x\max_{10}$ and $x\min_{10}$ are the upper and lower limits for the tenth gene, respectively, and $R(0,1)$ is a uniform random number generator that delivers an arbitrary numerical value between 0.0 and 1.0.

After the initial generation is established, fitness values, F_j^0, are computed for each chromosome using a suitable function evaluation. This is analogous to the objective function evaluation in gradient methods and symbolically is represented using

$$F_j^0 = F(\mathbf{X}_j^0) \tag{3}$$

In the case of aerodynamic shape optimization the function F represents a suitable CFD flow solver analysis. Fitness determination is followed by a ranking process where the most fit individual is given a number one ranking, the second most fit individual is ranked number two, and so on. This is simply determined for the initial GA generation using

$$\left.\begin{array}{rl} ic = 1 & \\ if\ (F_j^0 < F_{jj}^0)\ ic = ic + 1 & jj = 1, NC \\ IR_j^0 = ic & \end{array}\right\} j = 1, NC$$

This completes the GA initialization process. The next several subsections describe how the GA progresses from generation to generation using a variety of special operators that manipulate the chromosomes and allow evolution to occur.

Selection

The first operation required to determine the $(n+1)$st generation is selection. The chromosomes that will be used by the other GA operators (to be discussed shortly) must be selected from the nth generation chromosomes. Simply stated, the selection operation used in the present study is given by

```
$jj = 1$
$if\ (IR_j^n \le jj)$ then          ⎫          ⎫
    $\mathbf{X}_{jj}^* = \mathbf{X}_j^n$           ⎬ $j = 1, NC$ ⎬ $it = 1, NC$
    $jj = jj + 1$                  ⎭          ⎪
endif                                          ⎪
$if\ (jj > NC)$ stop                           ⎭
```

Once all the genes in each chromosome have been selected they are placed in a temporary holding array given by

$$\mathbf{G}^* = (\mathbf{X}_1^*, \mathbf{X}_2^*, \cdots, \mathbf{X}_i^*, \cdots, \mathbf{X}_{NC}^*)$$

Note how the fittest individuals in the *n*th generation are selected multiple times, the average individuals are selected a small number of times, and the least fit individuals are not selected at all. This biasing toward the fittest individuals is a key element in any GA. The chromosomes represented by the $\mathbf{G}^*$ quantities will be used by the succeeding operators to produce the final value for $\mathbf{G}^{n+1}$.

Passthrough

The simplest operator used in the present study is "passthrough." As the name implies, a certain number of the fittest chromosomes are simply "passed through" to the next generation. Because passthrough is always performed first, it operates on the chromosomes that have the highest fitness. This guarantees that the maximum fitness never drops from generation to generation. The number of chromosomes that are passed through to the next generation is controlled by the parameter p_B. For example, if $p_B = 0.1$ then ten percent of the chromosomes—those with the highest rankings—will be passed through to the next generation.

Random Average Crossover

The next GA operator to be discussed is called the random average crossover operator and is implemented gene by gene using the following formula:

$$x_{i,j}^{n+1} = \frac{1}{2}(x_{i,j1}^* + x_{i,j2}^*) \quad i = 1, 2, \cdots, NG \tag{4}$$

where the *j*1 and *j*2 subscripts are randomly chosen between 1 and NC. Values for $x_{i,j1}^*$ and $x_{i,j2}^*$ are taken from $\mathbf{G}^*$ and the newly computed values $x_{i,j}^{n+1}$ are then stored in $\mathbf{G}^{n+1}$. Once all the genes in a particular chromosome have been operated on the algorithm turns to the next chromosome. The number of chromosomes modified using random average crossover is determined by the parameter p_A. For example, if $p_A = 0.2$, then twenty percent of the chromosomes will be determined for the (*n*+1)st generation using random average crossover.

Perturbation Mutation

The next GA operator is called perturbation mutation and is implemented by first selecting a random chromosome $\mathbf{X}_j^*$ from $\mathbf{G}^*$. Then a single gene $x_{i,j}^*$ is randomly selected from $\mathbf{X}_j^*$ and modified using the following formula:

$$x_{i,j}^{n+1} = x_{i,j}^* + (x\max_i - x\min_i)[R(0,1) - 0.5]\beta \tag{5}$$

where β is a user-specified tolerance. Because this operator can cause the value of a particular gene to exceed one of its limits, additional checks to make sure this doesn't happen are required each time Eq. (5) is implemented. The number of chromosomes modified using perturbation mutation is determined by the parameter p_P. For example, if $p_P = 0.3$, then thirty percent of the chromosomes will be determined for the (n+1)st generation using the perturbation mutation operator.

Mutation

The last GA operator used in the present study is called the mutation operator and is implemented similarly to the perturbation mutation operator. First, a random chromosome $\mathbf{X}_j^*$ is chosen from $\mathbf{G}^*$. Then a single gene $x_{i,j}^*$ is randomly selected from $\mathbf{X}_j^*$ and supplied with a completely different value using the following formula:

$$x_{i,j}^{n+1} = (x\max_i - x\min_i)R(0,1) + x\min_i \tag{6}$$

Like the other operators used in the present study the mutation operator is controlled by a parameter, p_M. For example, if $p_M = 0.4$, then forty per cent of the chromosomes will be determined for the (n+1)st generation using the mutation operator.

General Algorithm Comments

For consistency the parameters (p_B, p_A ,p_P ,p_M) must sum to one. The passthrough operator is always performed first and always passes through the top p_B chromosomes from $\mathbf{G}^*$ to $\mathbf{G}^{n+1}$. Otherwise the order in which each operation is performed is immaterial. Once all values of $\mathbf{G}^{n+1}$ have been established, the algorithm proceeds to fitness evaluations using Eq. (3), ranking and on to succeeding generations until the optimization is sufficiently converged. The next section presents results where the algorithm just described is exercised.

Results

Case 1(Two-Gene Hill-Climbing Problem)

The first problem used to evaluate the GA just presented is a simple hill-climbing problem. It utilizes a continuously differentiable analytic function of the form $z = f(x, y)$ that has several peaks and valleys. An isometric graphical view of this function is displayed in Fig. 1 over the range $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$. The object of this exercise is to find the maximum value of z using the GA and in so doing gain insight into the workings of the GA process. Thus, the x and y values are the genes, each (x,y) pair is a chromosome and the value of z for each chromosome is the fitness.

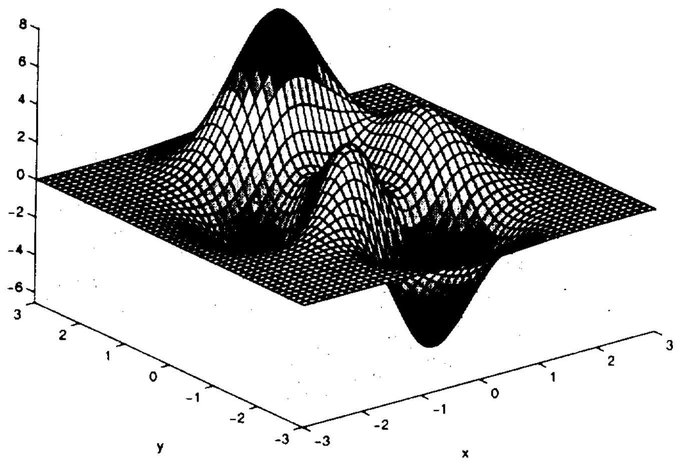


Fig. 1 Isometric view of the function used in the hill climbing problem.

Two typical GA convergence histories for the hill-climbing problem are presented in Fig. 2. For each curve the number of chromosomes, *NC*, was 20 and the β parameter in the perturbation mutation operator was 0.01. For each of these computations the GA procedure was continued until the error in the solution was reduced below 0.00001, i.e., CONV = 10^{-5}. The P vector notation appearing in the Fig. 2 caption is defined by P=(p_B, p_A, p_P, p_M). The fitness value at the end of each GA generation is plotted. As can be seen there is not that much difference between the two convergence histories, although the case with more mutation (higher values of p_P and p_M) and less passthrough (smaller value of p_B) does have a slightly superior convergence.

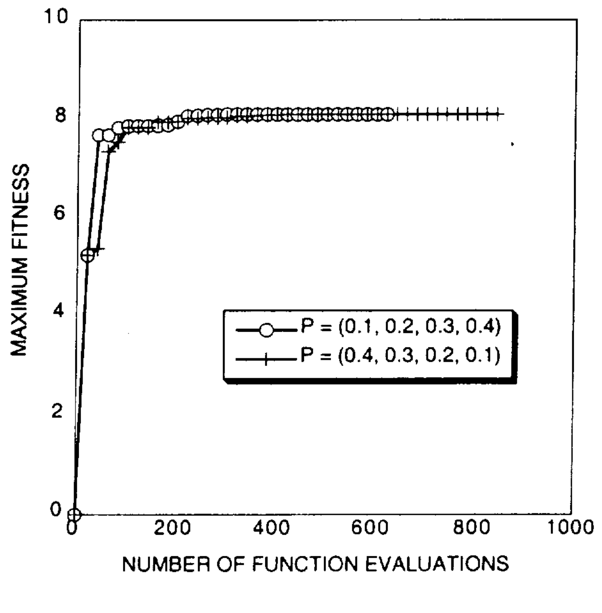


Fig. 2. Sample GA convergence history for the hill-climbing problem, β = 0.01, CONV = 10^{-5}, *NC* = 20.

Using the hill-climbing problem to study the effect of P, *NC* and β on GA convergence is handy because of the speed with which each complete GA optimization can be performed. Such a study is presented in

Figs. 3 and 4. For these results a value of CONV=10^{-5} was used for each computation. Each curve was generated by performing a separate GA over a large number of *NC* values ranging from 12 to 1024 in increments of 4. The resulting data were curve fit to produce each of the curves displayed.

Figure 3 presents the effect of wide variations in *NC* and β on GA convergence. Generally, the most optimal convergence occurs when the number of chromosomes is small, on the order of 12-20, and when β is between 1.0 and 0.001. Except for extremely small values (10^{-4} and smaller) β has little effect on GA performance. This is understandable since the perturbation mutation operator serves the role of providing small arbitrary corrections to the convergence process. As β is reduced in size the corrections become too small and GA convergence is delayed.

For moderate values of β the effect of the number of chromosomes is each generation has a relatively significant effect on convergence. This is due to the fact that a reasonable number of generations must be selected and operated upon in order to achieve convergence. As the number of chromosomes per generation is reduced, the number of generations that are enabled, for a fixed number of function evaluations, increases. This is the mechanism that produces faster convergence for smaller values of *NC*.

Figure 4 shows the effect of several different P vector values on GA performance as a function of chromosome size. As can be seen the different values of P generally have little influence on convergence. The only minor conclusion from Fig. 4 is that the curve generated when p_B = 0.4 is slightly poorer for smaller values of *NC*. Thus, it is concluded that the use of the passthrough operator in the present GA implementation should be limited.

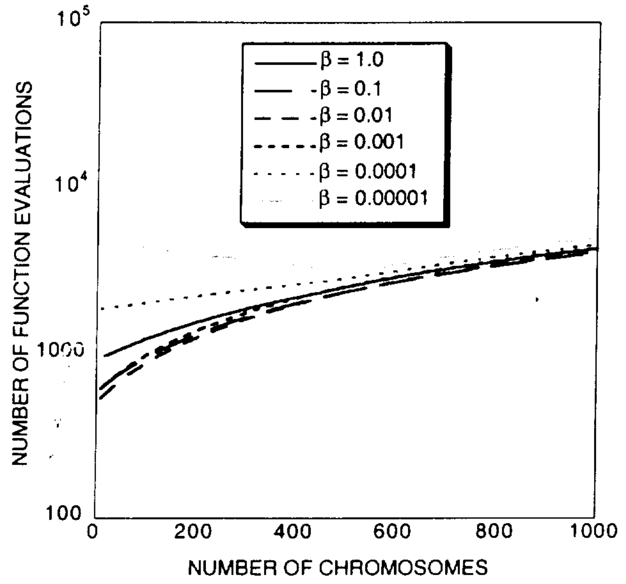


Fig. 3. Genetic algorithm convergence as a function of the number of chromosomes used in each generation and the size of perturbation mutations (β), CONV = 10^{-5}, P = (0.1, 0.2, 0.3, 0.4).

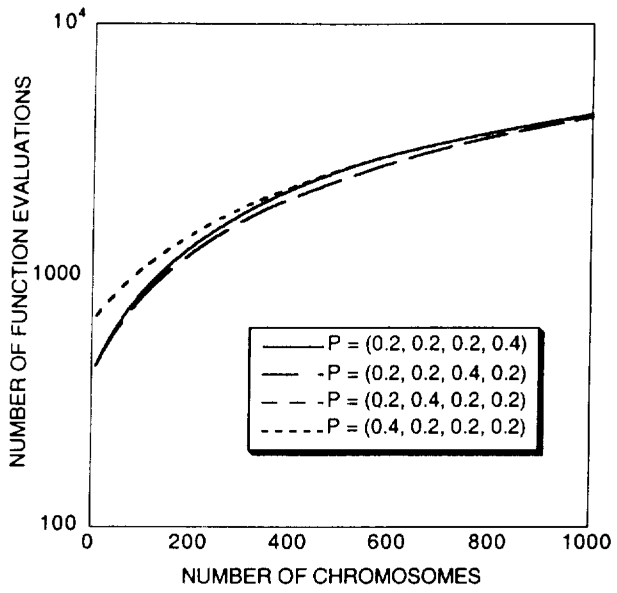


Fig. 4. Genetic algorithm convergence as a function of the number of chromosomes used in each generation and the type of operators used, CONV = 10^{-5}, $\beta = 0.01$.

Case 2 (Multiple-Gene Nozzle Problem)

An inverse design case is presented for the quasi-one-dimensional inviscid Euler equations. A one dimensional Newton solver (ARC1D) employing a central differencing / artificial dissipation scheme is used as the flow solver, i.e., function evaluation. The details of the flow solver and method are not pertinent to this study. The method converges typically in 10-20 nonlinear iterations for flows ranging from subsonic/transonic to supersonic nozzle shapes. The design parameterization is a set of knots, representing the height of the nozzle at various axial locations, which are used in a spline fit of the geometry. The case presented here used seven knots to define the 21grid point nozzle shown in Figure 5. The target nozzle shape is shown along with a sample of initial shapes created from a random set of initial genes. The vertical position of the knots is the gene encoding and each gene is bounded between 0.8 and 2.0 which includes the final design target. Given the exact diverging nozzle shown in Fig. 5, a target density (target ρ) is computed using ARC1D and is used to define the fitness for the GA. In particular, the fitness is defined as the inverse of the L2 norm of (target ρ - ρ), where ρ is computed from ARC1D using the best chromosome from the GA. Figure 5 also shows computed ρ for the initial nozzle shapes with the corresponding initial fitness for each shape.

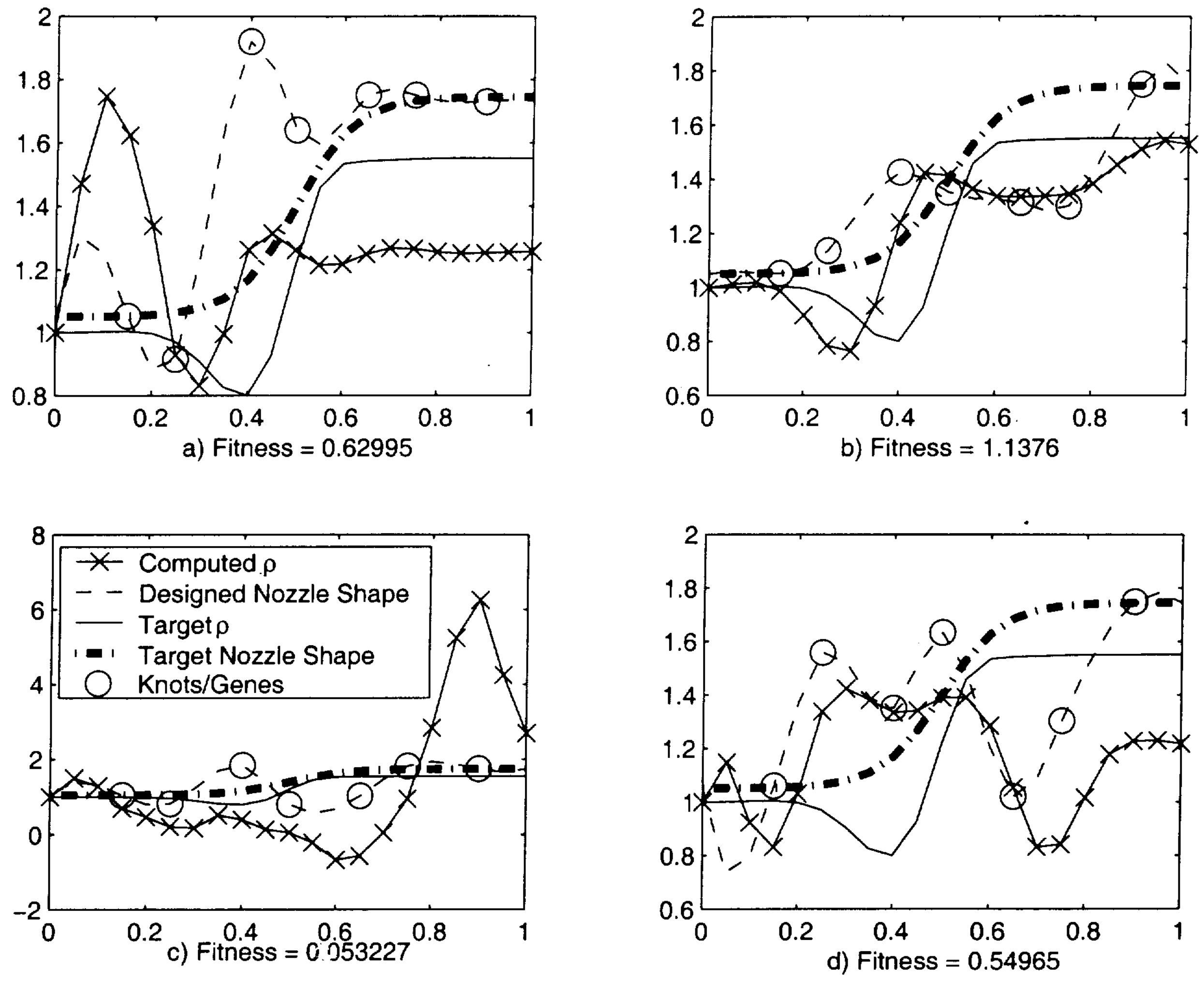


Fig. 5 Initial Chromosomes and Fitness for One- Dimensional Nozzle Design

The GA was run with P = (0.1, 0.2, 0.4, 0.3), β = 0.5 and a population of 20 chromosomes. Figure 6 shows a sample of various nozzle shapes and computed densities at different points in the GA convergence history. Figure 6a corresponds to an early time solution after three generations with a low fitness value and a poor design. Figures 6b and 6c show later time designs where the GA has produced by generation 100 a good fitness and design. Finally in Figure 6d, after 200 generations the nozzle design is fully converged and the design process has reproduced the target nozzle shape. Note that the last 100 generations would normally not be necessary in a real design process, as the iteration would probably be terminated earlier when the fitness based on the difference between the target and computed density reaches some prescribed tolerance.

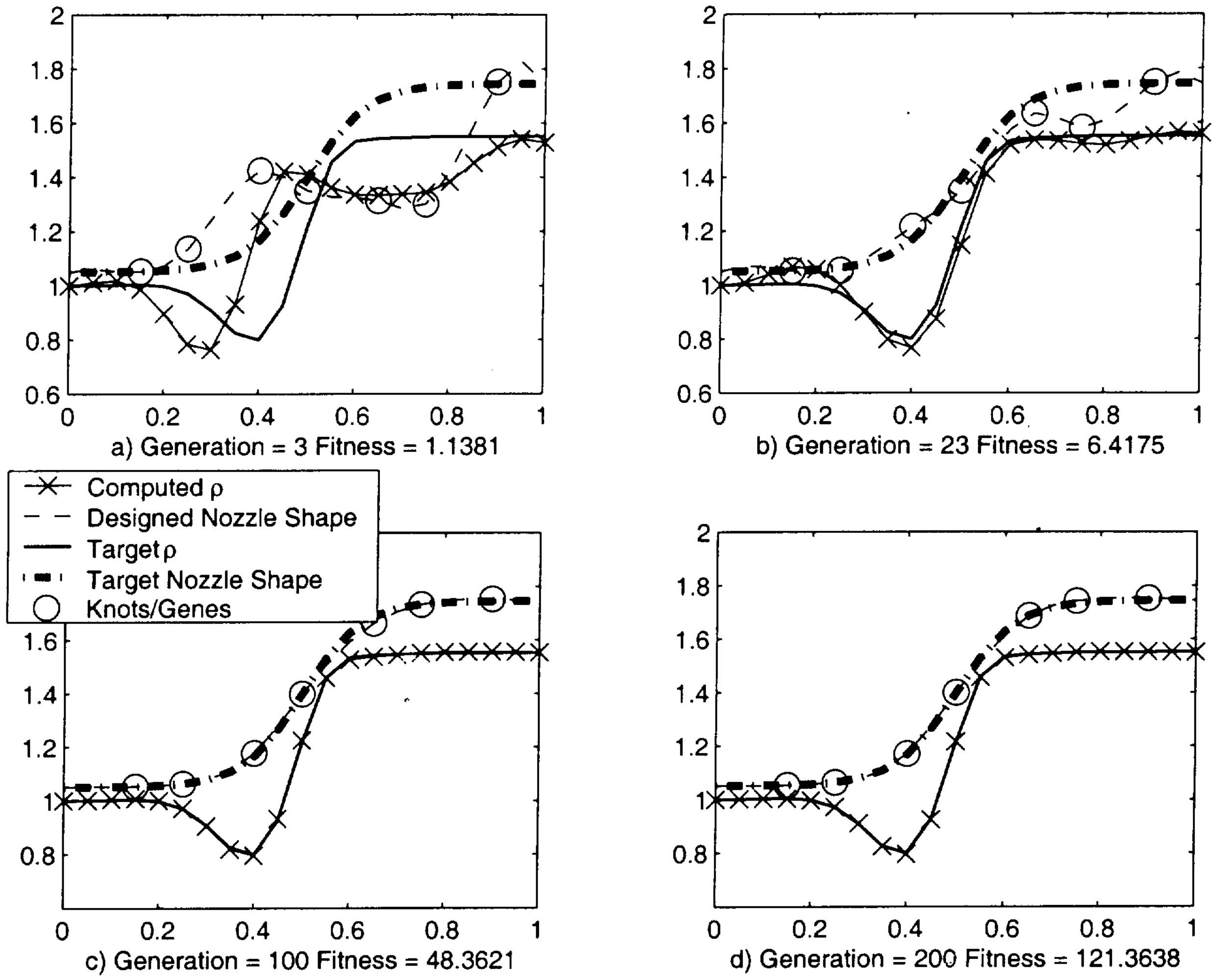


Fig. 6 Various Designs and Fitness for One- Dimensional Nozzle

Figure 7a shows the GA fitness verses generation, and Figure 7b shows the GA fitness as a function of function evaluations, (i.e. calls to ARC1D). In addition, Fig. 7 shows the average fitness convergence histories, which is quite noisy because of mutation, but still indicates convergence of the GA process. These results are typical of GA convergence for an inverse design case, (e.g. Tse and Chan[24] show inverse airfoil designs requiring 10,000 or more function evaluations), requiring on the order of 2000 to 4000 function evaluations to reach a design criteria. High fitness is achieved on the order of 2000 function evaluations, but the nozzle design requires a lot more evaluations to settle on the final shape. This can be improved but a modification of the fitness measure or the inclusion of a more constrained optimization. In this study we did not employ any smoothness criteria or other constraints, which would eliminated the more volatile designs achieved by the high values of p_p and p_m.

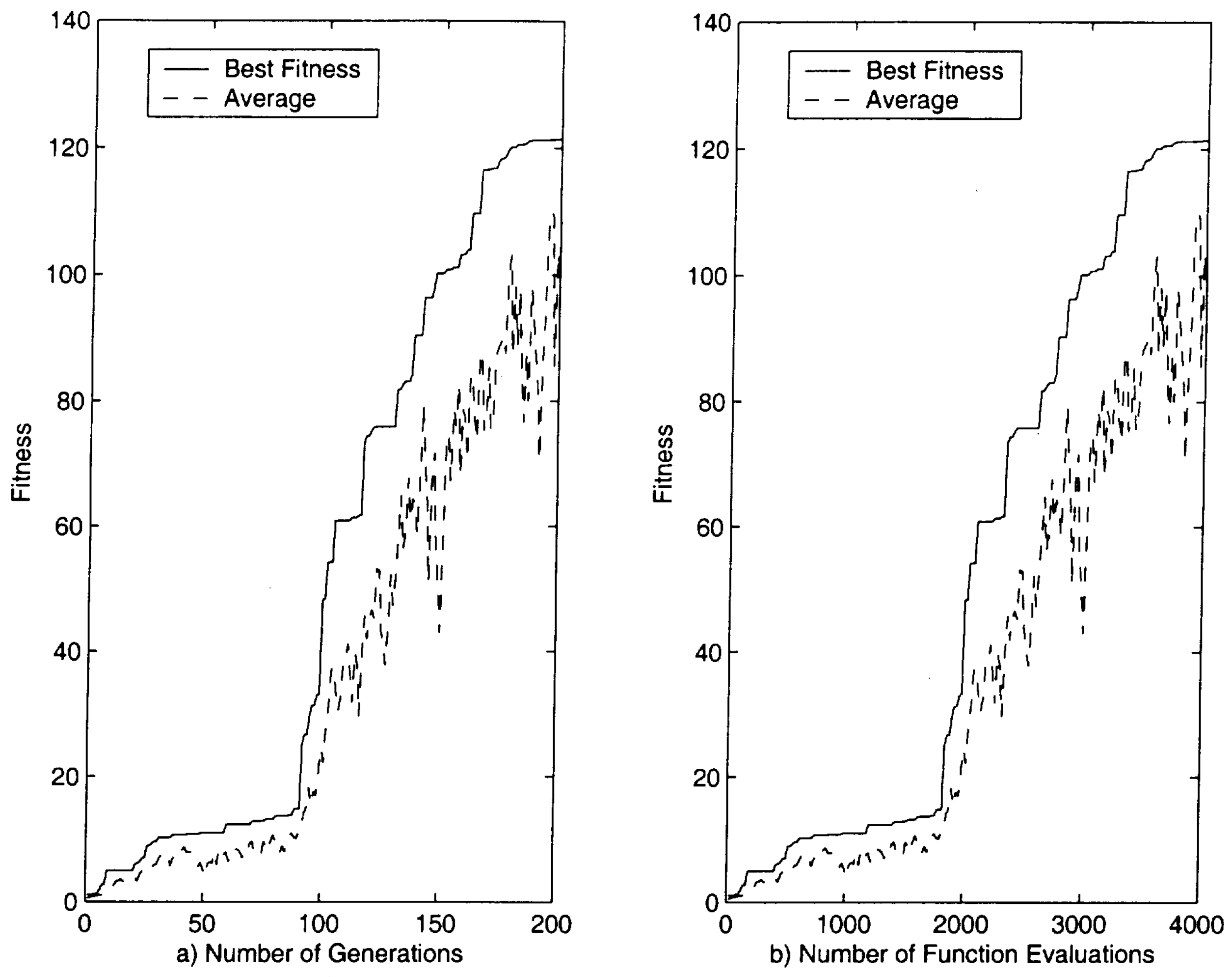


Fig. 7 Fitness Convergence for One- Dimensional Nozzle

Case 3 (Multiple-Gene Wing Optimization)

The last case presented demonstrating the use the GA optimization involves a three-dimensional transonic wing. The baseline geometry used for this set of computations is the RAE Wing A[25] mounted on a symmetry plane. This geometry involves a symmetric wing with a 9% maximum thickness-to-chord ratio, a taper ratio of 0.333, a leading edge sweep of 36.65° and an aspect ratio of 6.0. The freestream flow conditions used for this set of computations were held fixed at $M_\infty = 0.84$, $\alpha = 4^\circ$.

The TOPS[26,27] CFD code is utilized for this case to perform all function evaluations. This code is a three-dimensional full potential solver that utilizes a chimera zonal grid approach for handling complex geometries. It has the capability of producing complete numerical solutions about wing configurations (~115K grid points) in 1-2 min on a single SGI Origin 2000 processor. Each run consists of surface and volume grid generation; chimera hole cutting, donor cell search, and interpolation coefficient computation; and, finally, the flow solver step. Each of these steps is automatically coupled and executed without user intervention. This makes the GA/TOPS coupling easy and efficient.

Each grid used consists of two grid zones, an inner C-H topology grid fitted to the wing surface and an outer sheared-stretched Cartesian grid. The inner grid is generated using the HYPGEN grid generation program.[28] During GA iteration another option for generating the inner grid is available that first reads in a baseline-HYPGEN grid and then modifies it according to the wing geometry perturbations that have just been computed via the GA process. This saves a small amount of computer time in that the grid does not have to be recomputed from scratch at the beginning of each function evaluation.

For all computations used during GA iteration a coarse grid consisting of 115K points is used. This provides adequate accuracy for the optimization to proceed while allowing efficient code operation. Once the design optimization is complete, flow solutions for the initial and final geometries are recomputed using a finer grid (494K points), thus allowing a more accurate assessment of the performance improvement actually achieved.

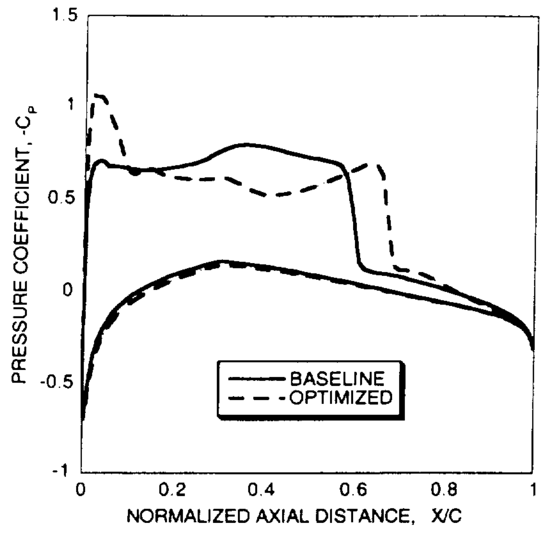
The design space discretization for this wing optimization problem consists of ten genes. Eight of the genes are associated with the wing upper surface thickness—four thickness variables at two span stations each, wing root and tip. These thickness values are implemented using bump functions[2] located at x/c = 0.15, 0.35, 0.45, 0.65. The value of wing twist at the root and tip are the two remaining genes. The maximum and minimum gene values used for all thickness functions are ±0.01, respectively. The wing root twist maximum and minimum values are 3° and 0°, respectively and the wing tip twist values are, 0° and -3°, respectively. The simple definition of the design space described above is chosen because the emphasis in this study is on GA optimization—it's implementation strategy and efficiency—not on aerodynamic efficiency. All computations in this section have been performed on an SGI workstation with a single R10000 (250MHz) processor using Fortran 77.

The fitness function for all wing optimizations reported in this section is given by

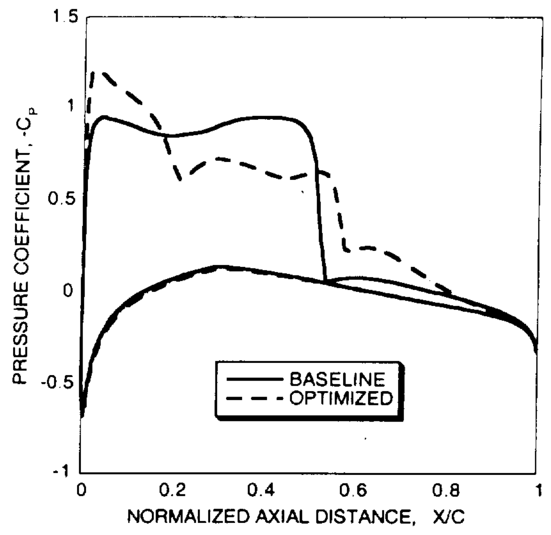
$$F = \frac{1.0}{\dfrac{C_D + 0.025}{C_L} + (C_L - 0.451)^2}$$

In this equation C_L and C_D are the wing inviscid lift and drag coefficients, respectively. The 0.025 constant is added to the drag coefficient to approximate viscous drag. The 0.451 constant in the denominator's second term is the coarse-grid baseline solution lift coefficient and provides a simple constraint on the optimization process keeping the lift nearly fixed.

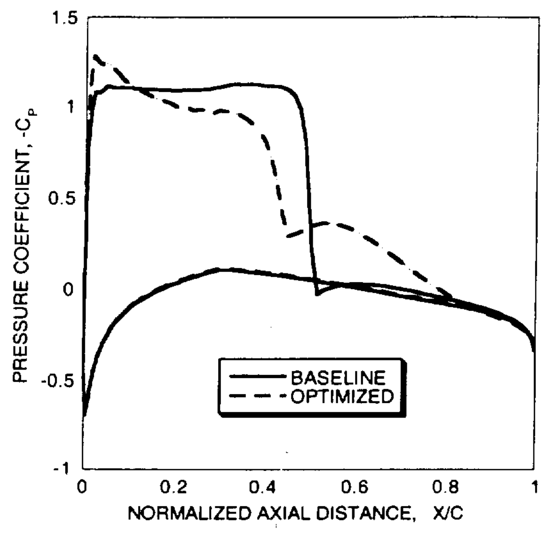
A typical result obtained using the GA optimization and design space discretization described above is presented in Figs. 8 and 9. This GA optimization was computed with 20 chromosomes and utilized a β value of 0.3 and P values of (0.1, 0.2, 0.3, 0.4). Figure 8 shows pressure coefficient distributions at several span stations for both the baseline and optimized wing solutions. Figure 9 shows Mach number contours for the upper wing surface for the same two solutions. Note that the optimization has produced a solution with significantly reduced shock strength, especially outboard of mid span. In addition, the single shock characteristic of the baseline solution has been replaced with a lambda shock pattern in the optimized solution. The inviscid drag for this computation was reduced by 67 counts while the lift decreased by only 0.0016.



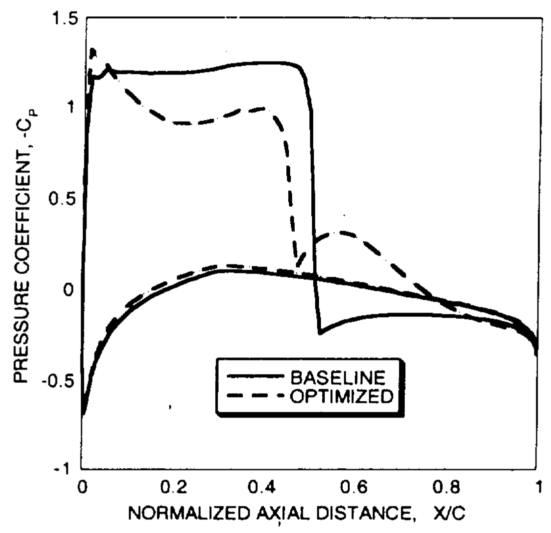
a) y/b = 0.21



b) y/b = 0.39

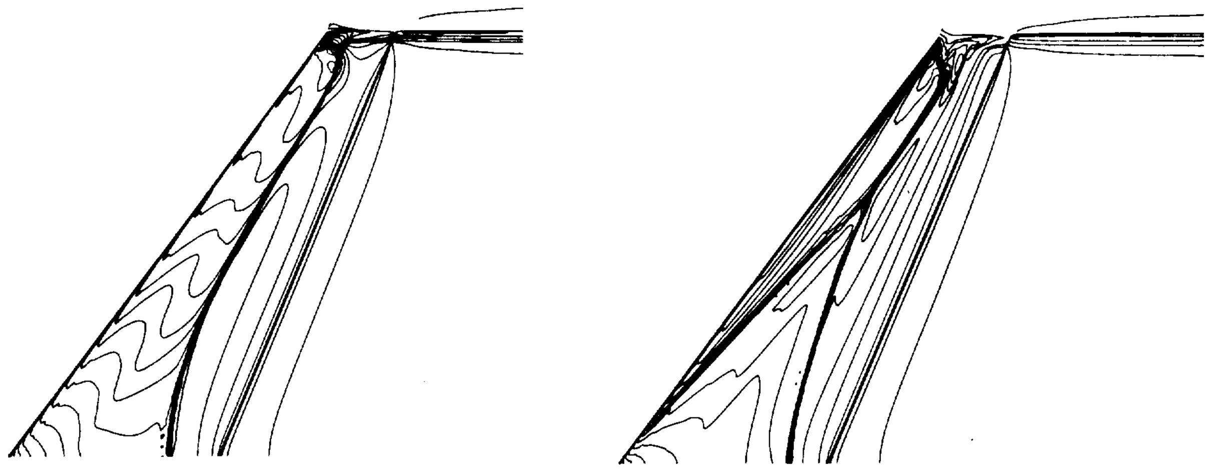


c) y/b = 0.62



d) y/b = 0.82

Fig. 8. Pressure coefficient distributions at selected span stations showing the baseline and optimized solutions, $M_\infty = 0.84$, $\alpha = 4^\circ$, $NC = 20$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4).



a) Baseline solution b) Optimized solution

Fig. 9. Mach number contours plotted on the upper wing surface for the baseline and optimized solutions, $M_\infty = 0.84$, $\alpha = 4^\circ$, $NC = 20$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4).

The next several results focus on GA convergence performance. Figure 10 shows the effect of flow solver convergence level on GA convergence. There are three curves plotted in Fig. 10, each showing how the maximum fitness converges with the number of CFD flow solver function evaluations over 30 generations. The first curve shows GA convergence when each flow solver solution maximum residual (RMAX) is reduced below 10^{-4}. Since the initial maximum residual is on the order of 0.01, this roughly corresponds to a two order of magnitude reduction in maximum residual for each solution—a level of convergence that does not achieve plottable accuracy. As can be seen in Fig. 10, the GA process barely improves the maximum fitness above the baseline level.

The second curve in Fig. 10 shows GA convergence when each flow solver solution maximum residual is reduced below 10^{-5}. This roughly corresponds to a three order of magnitude reduction in maximum residual and produces plottable accuracy for all but the most difficult areas to converge, e.g., the solution around the shock may not be completely converged for this level of maximum residual reduction. The third curve in Fig. 10 shows GA convergence when each flow solver solution maximum residual is reduced below 10^{-6}. This roughly corresponds to a four order of magnitude reduction in maximum residual and produces solid plottable accuracy over the entire solution. Note that for the last two curves displayed in Fig. 10, the GA convergence histories are nearly identical. However, the third curve—corresponding to the more tightly converged flow solutions—does produce a peak maximum fitness value that is about 13% higher. It is interesting to note that the GA optimization performance displayed in Fig. 10 (second and third curves) is comparable to convergence of the finite-difference GM optimization reported in Ref. 29, which was used to optimize a similar problem to that of the present section.

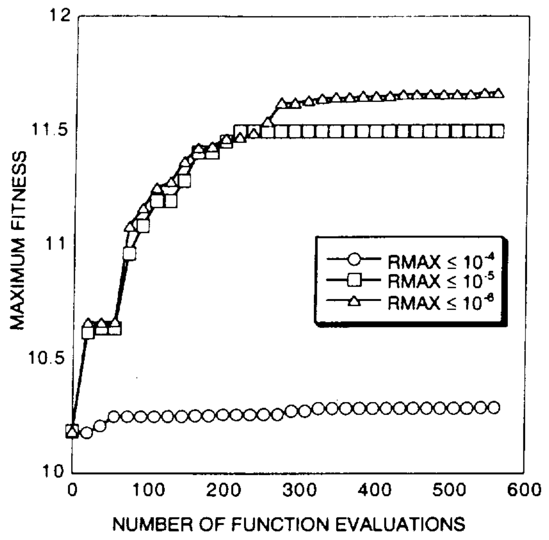


Fig. 10. Genetic algorithm convergence history comparison showing the effect of flow solver convergence level on GA performance, $M_\infty = 0.84$, $\alpha = 4^o$, $NC = 20$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4), NG = 30.

Another important way to view GA efficiency is to measure convergence against computer time and not against the number of function evaluations. The results presented in Fig. 10 are replotted and displayed versus CPU time in Fig. 11. Note that the level of flow solver solution convergence—as expected—has a profound effect on computation cost. The second curve, while producing only 87% of the maximum fitness that the third curve produces costs only about one-third as much in terms of computer time.

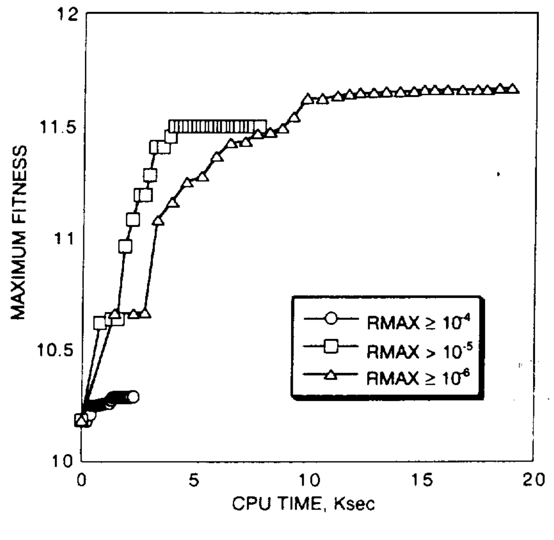


Fig. 11. Genetic algorithm convergence history comparisons showing the effect flow solver convergence level on GA CPU time, $M_\infty = 0.84$, $\alpha = 4^o$, $NC = 20$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4), NG = 30.

In the hill climbing problem the effect of number of chromosomes on GA convergence was presented in Figs. 3 and 4. For the present wing optimization problem it's more difficult to present such a complete study because of the computational expense involved. Nevertheless, a comparison between two results is possible and is presented in Fig. 12. The two curves correspond to a baseline computation utilizing 20 chromosomes and a separate computation utilizing 100 chromosomes. Both computations utilized flow solutions converged to 10-6, $\beta = 0.3$ and P = (0.1, 0.2, 0.3, 0.4). The 20-chromosome computation utilized 100 generations and the 100-chromosome computation utilized 20 generations, thus producing curves with similar numbers of function evaluations. For this ten-gene aerodynamic shape optimization problem it is clear that the use of 20 chromosomes produces superior GA convergence relative to the 100-chromosome computation.

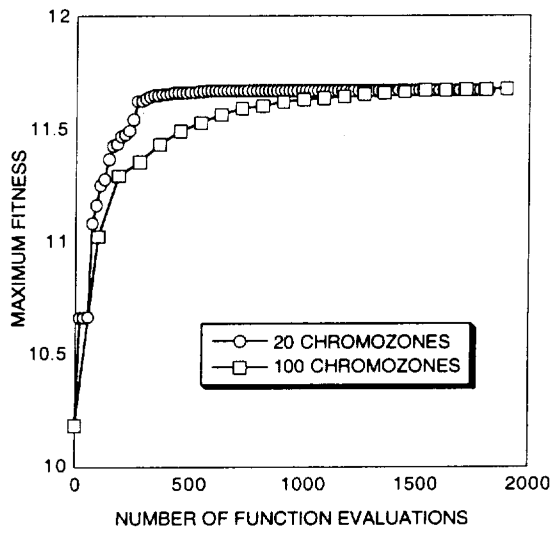


Fig. 12. Genetic algorithm convergence history comparison showing the effect of number of chromosomes on GA performance, $M_\infty = 0.84$, $\alpha = 4^\circ$, RMAX $\leq 10^{-6}$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4).

For all computations presented in this section a special solution initialization feature has been used in the CFD flow solver. Instead of using a freestream initial solution, the solution is initialized from a file that contains the most recent maximum fitness solution. Whenever the GA procedure encounters a new chromosome having a fitness that exceeds the previous maximum, the flow solution is saved and used to initialize each succeeding solution until it is replaced with a superior solution. Because changes in many chromosomes are small—being perturbations away from the several fittest individuals—use of this philosophy greatly reduces the amount of computer time required for a GA computation. A quantification of this improvement is presented in Fig. 13. The first curve shows a GA convergence history that uses the solution restart option, and the second curve shows an identical GA convergence history that does not use solution restart. Each flow solution for both curves is converged to RMAX $\leq 10^{-6}$. For this optimization the use of restart saves a factor of 2.7 in computer time.

The successful use of solution restart and less well-converged solutions during GA iteration is a testament to the robustness of the GA approach for optimization. Both these shortcuts introduce error into the GA process, but, if suitably controlled, do not hamper overall GA convergence.

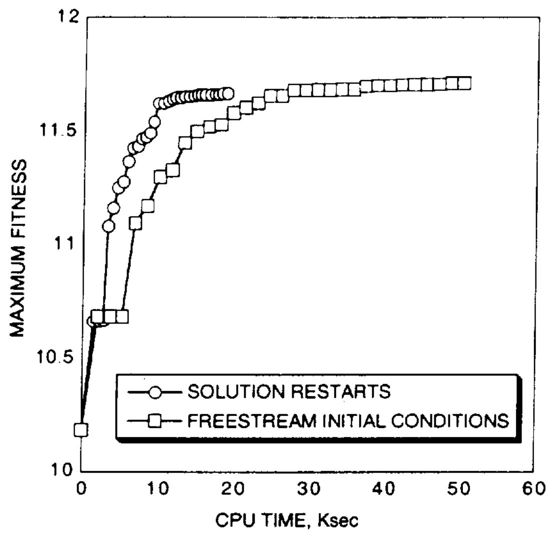


Fig. 13. Genetic algorithm convergence history comparisons showing the effect solution restarts on GA CPU time, $M_\infty = 0.84$, $\alpha = 4^\circ$, $NC = 20$, $\beta = 0.3$, P = (0.1, 0.2, 0.3, 0.4), NG = 30, RMAX $\leq 10^{-6}$.

Conclusions

A genetic algorithm (GA) procedure suitable for performing optimizations, including aerodynamic shape optimizations is presented. It uses real-number encoding to represent all geometric parameters as genes in the GA. Four operators are presented enabling advancement from one generation to the next. They include passthrough, random average crossover, perturbation mutation and mutation. The results indicate that the GA approach is robust and continues to converge even with significant amounts of lack-of-convergence error in each function evaluation.

References

1. Reuther, J., "Aerodynamic Shape Optimization Using Control Theory," *RIACS Rep. 96.09*, 1996.

2. Hicks, R. and Henne, P., Wing Design by Numerical Optimization, *J. Aircraft*, **Vol. 15**, 1978, pp. 407-412.

3. Burgreen, G. W. and Baysal, O., "Three-Dimensional Aerodynamic Shape Optimization of Wings Using Sensitivity Analysis," AIAA Paper 94-0094, 1994.

4. Luenberger, D. G., *Linear and Nonlinear Programming*, 2nd Ed., Addison-Wesley Publishing Co., Inc., Reading, MA, 1984.

5. Bischof, C., Carle, A., Corliss, G., Griewank, A. and Hovland, P., "ADIFOR—Generating Derivative Codes from Fortran Programs," *Scientific Programming*, No. 1, 1992, pp. 1-29.

6. El-banna, H. M. and Carlson, L. A., "Determination of Aerodynamic Sensitivity Coefficients Based on the Transonic Small Perturbation Formulation," *AIAA J.*, Vol. 27, 1990, pp. 507-515.

7. El-banna, H. M. and Carlson, L. A., "Aerodynamic Sensitivity Coefficients Using the Three-Dimensional Full Potential Equation," *J. Aircraft*, Vol. 31, 1994, pp. 1071-1077.

8. Arslan, A. E. and Carlson, L. A., "Integrated Determination of Sensitivity Derivatives for an Aeroelastic Transonic Wing," *J. of Aircraft*, Vol. *33*, 1996, pp.224-231.

9. Jameson, A. (1988) Aerodynamic Design via Control Theory, *J. of Sci. Comp.* **3**, 233-260.

10. Jameson, A. (1990) Automatic Design of Transonic Airfoils to Reduce the Shock Induced Pressure Drag, *31st Israel Annual Conf. on Aviation and Aeronautics*, Tel Aviv, 5-17.

11. Goldberg, D. E., "*Genetic Algorithms in Search, Optimization and Machine Learning*," Addison-Wesley, Reading, MA, 59-88, 1989.

12. Davis, L., "*Handbook of Genetic Algorithms*," Van Nostrand Reinhold, New York, 1991.

13. Beasley, D., Bull, D. R. and Martin, R. R., "An Overview of Genetic Algorithms: Part 1, Fundamentals," *University Computing*, Vol. 15, No. 2., 1993, pp. 58-69.

14. Obayashi, S. and Tsukahara, T., "Comparison of Optimization Algorithms for Aerodynamic Shape Design," *AIAA J.*, Vol. 35, 1997, 1413-1415.

15. Bock, K.-W., "Aerodynamic Design by Optimization," Paper 20, *AGARD CP-463*, 1990.

16. Quagliarella, D. and Della Cioppa, A., "Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils," *AIAA Paper 94-1896-CP*, 1994.

17. Vicini, A. and Quagliarella, D., "Inverse and Direct Airfoil Design Using a Multiobjective Genetic Algorithm," *AIAA J.*, Vol. 35, 1997, pp. 1499-1505.

18. Obayashi, S., Yamaguchi, Y. and Nakamura, T., (1997) "Multiobjective Genetic Algorithm for Multidisciplinary Design of Transonic Wing Planform," *J. of Aircraft*, Vol. 34, 1997, pp. 690-693.

19. Sasaki, D., Obayashi, S., Sawada, K. and Himeno, R., "Multiobjective Aerodynamic Optimization of Supersonic Wings Using Navier-Stokes Equations," European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2000, Barcelona, Spain, Sept. 11-14, 2000.

20. Oyama, A., "Multidisciplinary Optimization of Transonic Wing Design Based on Evolutionary Algorithms Coupled with CFD Solver," European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2000, Barcelona, Spain, Sept. 11-14, 2000.

21. Oyama, A., "Wing Design Using Evolutionary Algorithms," PhD Thesis, Dept. of Aeronautics and Space Engineering, Tohoku University, Senadi, Japan, March 2000.

22. Houck, G. R., Joines, J. A. and Kay, M. G., "A Genetic Algorithm for Function Optimization: A Matlab Implementation,"

23. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, AI Series, Springer-Verlag, New York, 1994.

24. Tse, D. and Chan, L., "Transonic Airfoil Design Optimization using Soft Computing Methods," *Canadian Aeronautics and Space J.*, Vol. 46, No. 2, June 2000, pp. 65-73.

25. Treadgold, A. F., Jones, A. F. and Wilson, K. H., "Pressure Distribution Measured in the RAE 8ft X 6ft Transonic Wind Tunnel on RAE Wing 'A' in Combination with an Axi-Symmetric Body at Mach

Numbers of 0.4, 0.8 and 0.9," *Experimental Data Base for Computer Program Assessment*, AGARD-AR-138, May 1979.

26. Holst, T. L., "Full Potential Equation Solutions Using a Chimera Grid Approach," AIAA Paper No. 96-2423, June, 1996.

27. Holst, T. L., "Multizone Chimera Algorithm for Solving the Full-Potential Equation," *J. of Aircraft*, Vol. 35, No. 3, May-June 1998, pp. 412-421.

28. Chan, W. M., Chiu, I., and Buning, P. G., " User's Manual for the HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface," NASA TM 108791, Oct. 1993.

29. Cheung, S. and Holst, T., "Aerodynamic Shape Optimization Using a Combined Distributed/Shared Memory Paradigm," Proceeding of the Fourth CAS Workshop, Moffett Field, Calif., 1998.