Evolutionary Algorithms in Engineering and Computer Science

Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications

Edited by

K. Miettinen
University of Jyväskylä, Finland

P. Neittaanmäki
University of Jyväskylä, Finland

M. M. Mäkelä
University of Jyväskylä, Finland

J. Périaux
Dassault Aviation, France

JOHN WILEY & SONS, LTD
Chichester • Weinheim • New York • Brisbane • Singapore • Toronto

Invited lecture, **EUROGEN'99 - Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications**, Editors: K. Miettinen, M. M. Makela, P. Neittaanmaki and J. Periaux, John Wiley & Sons, Ltd., Jyvaskyla, Finland, May 30 - June 3, 1999.

Multidisciplinary Hybrid Constrained GA Optimization

G.S. Dulikravich, T.J. Martin, B.H. Dennis and N.F. Foster
Department of Aerospace Engineering, 233 Hammond Building
The Pennsylvania State University, University Park, PA 16802, USA

Introduction

Realistic engineering problems always involve interaction of several disciplines like fluid dynamics, heat transfer, elasticity, electromagnetism, dynamics, etc. Thus, realistic problems are always multidisciplinary and the geometric space is typically arbitrarily shaped and three-dimensional. Each of the individual disciplines is governed by its own system of governing partial differential equations or integral equations of different degree of non-linearity and based on often widely disparate time scales and length scales. All of these factors make a typical multidisciplinary optimization problem highly non-linear and interconnected. Consequently, an objective function space for a typical multidisciplinary problem could be expected to have a number of local minimums. A typical multidisciplinary optimization problem therefore requires the use of optimization algorithms that can either avoid the local minimums or escape from the local minimums. Non-gradient based optimizers have these capabilities. On the other hand, once the neighborhood of the global minimum has been found, the non-gradient based optimizers have difficulty converging to the global minimum. For this purpose, it is more appropriate to use gradient-based optimizers.

Addition of constraints of both equality and inequality type to a typical multidisciplinary optimization problem reduces significantly the feasible domain of the objective function space. To find such often-small feasible function space, the optimizer should be able to search as large portion of the objective function space as possible. Again, non-gradient based optimizers are capable of performing this task. When constraints of equality type are to be enforced, the gradient-based optimizers can perform this task very accurately.

One of the primary concerns of any optimization algorithm is the computational effort required to achieve convergence. Except in the case of certain sensitivity based optimization algorithms and genetic algorithms with extremely large populations, the computer memory is not an issue. Typical constrained optimization problems in engineering require large number of objective function evaluations. Each function evaluation involves a very time-consuming computational analysis of the physical processes involved. The real issue is the reduction of the overall computing time required to perform the optimization. Therefore, an efficient optimization algorithm should require the least

number of objective function evaluations and should utilize the most efficient disciplinary (or multidisciplinary) analysis algorithms for the objective function evaluations.

This suggests that it might be beneficial to utilize several different optimization algorithms during different phases of the overall optimization process. The use of several optimization algorithms can be viewed as a backup strategy (Dulikravich, 1997) so that, if one optimization method fails, another optimization algorithm can automatically take over. This strategy of using a hybrid optimization approach and performing the switching among the optimizers will be demonstrated and discussed in this text.

In addition, it might be beneficial to perform disciplinary (or multidisciplinary) analysis by using simplified or surrogate physical models during the initial stages of optimization. Because of their reduced degree of non-linearity, such models typically require significantly less computing time to evaluate than the full non-linear model of the physical process. This strategy of using progressively more accurate (and computationally expensive) objective function evaluation models will be demonstrated and discussed.

Finally, the flexibility of the parameterization of the design variable space (for example, parameterization of geometry in the case of shape optimization) can affect the convergence rate of the optimization process and the quality of the final result. The effects of the degree of inherent flexibility of the discretization algorithm on the optimization convergence rate will be discussed in this text.

Hybrid Constrained Optimization

Various optimization algorithms have been known to provide faster convergence over others depending upon the size and shape of the mathematical design space, the nature of the constraints, and where they are during the optimization process. This is why we created a hybrid constrained optimization software. Our hybrid optimizer incorporates four of the most popular optimization modules; the Davidon-Fletcher-Powell (DFP) (Davidon, 1959; Fletcher and Powell, 1963) gradient search method, a genetic algorithm (GA) (Goldberg, 1989), the Nelder-Mead (NM) (Nelder and Mead, 1965) simplex method, and simulated annealing (SA) (Press et al., 1986) algorithm. Each algorithm provides a unique approach to optimization with varying degrees of convergence, reliability, and robustness at different stages during the iterative optimization process. A set of rules and heuristics were coded into the program to switch back and forth among the different optimization algorithms as the process proceeded. These rules will be discussed in this text.

The evolutionary hybrid optimizer handled the existence of equality and inequality constraint functions in three ways: Rosen's projection method, feasible searching, and random design generation. Rosen's projection method (Rao, 1996) provided search directions that guided the descent direction tangent to active constraint boundaries. In the feasible search (Foster and Dulikravich, 1997), designs that violated constraints were automatically restored to feasibility via the minimization of the active global constraint functions. If at any time this constraint minimization failed, random designs were generated about the current design until a new feasible design was reached.

Gradients of the objective and constraint functions with respect to the design variables, also called design sensitivities, were calculated using either finite differencing formulas, or by the much more efficient method of implicit differentiation of the governing equations (Hafka and Malkus, 1991). The population matrix was updated every iteration with new

designs and ranked according to the value of the objective function. During the optimization process, local minimums can occur and halt the process before achieving an optimal solution. In order to overcome such a situation, a simple technique has been devised (Dulikravich and Martin, 1994; 1996). Whenever the optimization stalls, the formulation of the objective function is automatically switched between two or more functions that can have a similar purpose. The new objective function provides a departure from the local minimums and further convergence towards the global minimum.

As the optimization process converges, the population evolves towards the global minimum. The optimization problem was completed when one of several stopping criterion was met; (1) the maximum number of iterations or objective function evaluations was exceeded, (2) the best design in the population was equivalent to a target design, or (3) the optimization program tried all four algorithms but failed to produce a non-negligible decrease in the objective function. The latter criterion was the primary qualification of g is convergence and it usually indicated that a global minimum had been found.
Following is a brief discussion of the most important features of each optimization module that was used in our hybrid constrained optimizer.

Gradient Search Algorithm

Optimizers based on a gradient search concept require that the negative gradient of a scalar function in the design variable space be multiplied by the optimum line search step size, α^* before adding it to the vector of design variables, $\vec{V}$. Unfortunately, the search direction is only first-order accurate and it is slow in minimizing the objective function, especially near a local or global minimum. Also, this method alone does not support logic to ensure that the constraints are not violated. The simplest way to introduce this capability would be to ensure that the line search does not extend into an infeasible region. Such a technique is very simple, but it can often stall at a stationary point located on a constraint boundary before reaching the global minimum.

The following several sections describe how one can improve convergence and handle constrained gradient-based optimization problems.

Improving Convergence of Gradient Based Optimizers

The DFP algorithm uses quasi-Newtonian or rank-two updates and yields second-order accuracy without excessive calling of the analysis program. The DFP is the most computationally expensive method in our hybrid optimization algorithm, but its convergence is both maximized and guaranteed. This algorithm is susceptible to local minimums and it can get stuck on a constraint boundary. Depending on how this procedure stalls, the hybrid optimizer switches to another optimization routine.

The optimal line search parameter, α^*, is that value that minimizes the objective function along the line search direction. Many one-dimensional minimization algorithms have been published for the purposes of line searching (Golden Search, Fibonacci, quadratic interpolation,etc.). practically all of these line search techniques have difficulties with the constrained non-linear optimization when the objective function variation along the line search direction can have multiple local minimums.

Therefore, a more effective method has been developed for this sub-optimization procedure (Dulikravich and Martin, 1994). A small number (five to ten) of design points are generated between the current design ($\alpha = 0$) and the end of the line search in the searching direction, $\vec{S}$. The end of the line search is determined by calculating the distance to the closest design variable bound in the parametric space. The objective (or constraint) function is evaluated at each of these points. Then, a spline is passed through these control points and interpolated at a large number (for example, 1000) equidistantly spaced points. By sorting these interpolated values, the minimum of this new curve is found and the corresponding α^* is determined. The minimum of the interpolated spline is, in general, different from the true objective function minimum. Therefore, the newly found point, (α^*, F*), is added to the original set of design points, a new spline curve is fitted and interpolated through this enlarged set, the minimum of this new curve is found, and the corresponding α^* is determined. The process is repeated several times until the new value of α^* is very close to the previous one indicating that the optimal α^* has been found in the particular search direction.

Enforcement of Constraints

A common method of dealing with constraints is to use a penalty function. Penalty functions are added to the scalar objective function with appropriate weighting factors for scaling of the constraint functions with respect to the objective function value. The use of penalty functions is highly discouraged. Not only do they waste the computing time on evaluating the objective function for infeasible designs, they also change the nature of the design space, often converging to minimums that are nowhere near the global minimum. Penalty methods were not used in the constrained hybrid optimization system because other, more effective procedures were implemented (Rosen's projection method and feasible search).

Rosen's projection method is a constraining approach that was designed to handle the existence of inequality and equality constraint functions (Rao, 1996). It is based on the idea of projecting the search direction into the subspace tangent to any active constraints. After the new search direction has been determined, any standard line search can be performed to update the design. After a line search has been employed and the design updated, the resulting design may become infeasible. A restoration move is then required from the new design point, back to the constraint boundary at the point. The effect of this restoration is to reduce all active constraint functions to zero.

Rosen's restoration is valid only for simple problems that have a local linear behavior. This formula tends to become unstable when nonlinear optimization problems are attempted. In these cases and for the multidisciplinary optimization problems, the feasible search method is highly recommended to restore an infeasible design back to the boundary of the feasible domain. The simplest way to accomplish this task is to employ a sub-optimization problem that minimizes the sum of active constraint functions. Notice that the equality constraints are always active. The evolutionary hybrid optimization algorithm uses DFP method to minimize this function.

Genetic Algorithm

The GA's evolutionary approach utilizes its random nature to escape local minimums. When the average cost function of the new generation is not improved, the GA becomes an inefficient optimizer. This most often occurs when its random nature is prevalent, producing several bad and infeasible designs. The GA will be switched to the NM algorithm because the NM works efficiently upon these worst designs. When the variance in the objective function values of the population are very small, the population begins to contract around a possible global minimum. At this point, the optimization switches to the DFP gradient-based algorithm because DFP has the ability to quickly zoom in on that minimum. The GA develops new population members with each iteration, but only those members whose fitness is higher than that of the worst member will be allowed to enter the population.

The GA can handle constraints on the design variable bounds, but it is inherently unable to handle constraint functions. The new set of designs may not, in general, be feasible. Therefore, the feasibility of each generated design is checked and, if any constraints are violated, a feasible search is performed on the new design. If the feasible search fails, a new design is generated randomly about the best design in the population until a satisfactory one is found. Random designs were generated using a Gaussian shape probability density cloud centered on the current design, $\vec{V}^0$, with a randomly generated number $0 < R < 1$.

$$V_i = V_i^0 \pm \sqrt{-2\sigma^2 \left(V_{i,max} - V_{i,min}\right) \ln R}$$

The non-dimensional variance, σ^2, in this function was determined by the conditions of the optimization process. For example, the user specifies the maximum variance in the input of the optimization algorithm.

Nelder-Mead Simplex Algorithm

For high-dimensional problems, it is known that the sequential simplex-type algorithms are more efficient and robust than gradient based algorithms in minimizing classical unconstrained test functions. The NM method is a zeroth order method that utilizes a simplex generated by the population of previously generated designs. The NM begins by defining a group of solution sets, which, when mapped, form a geometric figure in the N_{var}-dimensional design space, called a simplex. The simplex does not need to be geometrically regular, so long as the distribution of the vertices remains fairly balanced. The NM then becomes a downhill method by obtaining a search direction which points from the worst design in the population through the centroid of the best designs. This algorithm is very easy to program and it has the least amount of objective function evaluations per optimization cycle. The existing population matrix of previously generated feasible designs makes the NM even cheaper to employ. It improves only the worst design in the population with each iteration.

The population of feasible designs, which has been ranked in ascending order according to its objective function values, is utilized such that the centroid of the best designs (omitting the worst design in the population) is computed

$$\vec{V}_{mean} = \frac{1}{N_{var}} \sum_{j=1}^{N_{var}} \vec{V}^{j}$$

A search direction can now be defined as the vector from the worst design to the centroid of the remaining best designs

$$\vec{S} = \frac{\vec{V} - \vec{V}_{N\,var}}{\left|\vec{V}_{mean} - \vec{V}_{N\,var}\right|}$$

Once this search direction is computed, it is projected into the subspace tangent to the active constraints using Rosen's projection method, and then a line search is employed. The initial guess step size of the line search should be set to the average design variable range in the current population. If the line search succeeds, the new design may not be feasible. Then, a restoration move is employed. Eventually, a new feasible design will be obtained that should improve the worst design in the population.

Simulated Annealing Algorithm

The SA method is analogous with thermodynamic annealing. As a liquid is slowly cooled, thermal mobility of the molecules is slowly lost so that the atoms are able to line themselves up and form a pure crystal without defects. A pure crystal is the state of minimum energy for this system. SA provides a slow reduction in its random searching capabilities that it uses to produce search directions. The continuous minimization SA algorithm uses a modification to the downhill NM simplex method. This SA wanders freely through the local minimum neighborhood so it is used when the optimization slows down or stalls. Unfortunately, it can worsen the objective function in the later optimization cycles. The SA is suitable for large scale optimization problems, especially when the desired global minimum is hidden among many, poorer local extrema. The nature of the SA lends itself to the early optimization cycles. It is also useful for escaping from a local minimum after the DFP algorithm gets stuck in a local minimum. The basic idea of continuous minimization of an objective function using SA is to find an appropriate design variable change, like that of a steepest descent or downhill simplex method. The NM algorithm is used to obtain a search direction. Logarithmically distributed random numbers, proportional to the cooling temperature, $\tilde{T}$, are added to the function values at the vertices of the simplex.

$$\tilde{F} = F - \tilde{T}\ln(R)$$

Here, R is a random number between 0.0 and 1.0. Another random variable is subtracted from the objective function value of every new point that is tried as a replacement design. The simplex will expand to a size that can be reached at this temperature and then executes a stochastic, tumbling Brownian motion within that region. The efficiency with which the region is explored is independent of its narrowness or aspect ratio. If the temperature is reduced slowly enough, it is likely that the simplex will contract about the lowest minimum encountered.

The cooling scheme that was employed in the hybrid optimizer reduced the temperature $\tilde{T}$ to $(1-\gamma)\tilde{T}$ every $\tilde{K}$ optimization cycles, where the optimum γ and $\tilde{K}$ are determined empirically for each problem. When the cooling temperature has reduced to a significantly small value, the program switches to a different optimizer.

Automated Switching Among the Optimizers

A set of rules has been added to the hybrid constrained optimization system in order to make the switching among the algorithms automatic, as well as to utilize some of the heuristic understanding of each algorithm's behavior. The purpose of this switching was to increase the hybrid optimizer's robustness and improve upon its convergence. Each rule was based upon and incorporated with the unique behavior of each numerical optimization algorithm. The timing of the switching among the algorithms was forced to occur during those instances in which the particular algorithm performed badly, stalled, or failed. The first algorithm that the optimization process chose to switch to was determined by reasoning and by trial and error. If the subsequent algorithm also failed in its processes, the opportunity was made for every one of the algorithms in the system to have a try at the problem. When all the available algorithms had been tried on a particular population of feasible designs, and all failed at their given tasks, then the program was terminated. The rules for switching will now be discussed for each algorithm in the hybrid optimization system. Figure 1 demonstrates the major characteristics of this switching process in flowchart form.

Rules for Switching Optimization Modules

Local Minimum Rule

This rule has been developed for gradient-based methods. The gradient search optimization algorithm is switched whenever the change in the objective is less than a user-specified tolerance, $\overline{\Delta F}$.

$$\frac{|F^* - F_K|}{|F^* - F_0|} < \overline{\Delta F}$$

Here, F^* is the value of the objective function at the new optimum, F_0 is the value of the objective function for the design at the start of the gradient search, and F_K is the value of the objective function from the previous optimization cycle. The GA is the first algorithm chosen, then comes the SA and finally the NM. The genetic algorithm is chosen first because its random searching capabilities are useful in escaping local minimums. This rule is also applicable whenever the program stalls on a constraint boundary.

Descent Direction Rule

When the DFP search direction is not a descent direction, the dot product of the search direction, $\tilde{S}$, and the gradient, ∇F, is greater than zero. The search direction may be different from the negative gradient direction because of the DFP update formula and because that search direction is projected onto the subspace of active constraints (Rosen's projection method).

$$\vec{S} \bullet \nabla F > 0$$

When this condition is met, the inverse Hessian matrix is re-initialized to the identity matrix. If the inverse Hessian is already equal to the identity due to re-initialization, the program is switched to simulated annealing (SA – NM – GA). The randomness and simplex searching methods of the simulated annealing process provide quick and effective ways of navigating through the irregular design spaces. New designs (optimum along the line search direction) created by the DFP are added to the population matrix and the DFP always works on the best member in the population matrix.

The GA has several criterions that can qualify its performance and so several switching rules have been developed. The most often used switching criterion is based upon the variance in the population. As the GA proceeds, it tends to select members in the population with like qualities to breed more often. The result is that the population tends to acquire a similar set of characteristics and the variation in the population reduces. This can happen too quickly when the specified mutation is too infrequent. In the ideal situation, the design variables of the population will tend to collect around the global minimum, but may have difficulty in finding it.

Design Variance Limit Rule

This is the first rule for switching from the GA. It is defined as

$$\sigma_V = \frac{1}{N_{VAR} N_{POP}} \sum_{i=1}^{N_{VAR}} \frac{\sqrt{\sum_{j=1}^{N_{POP}} \left(P_{i,j} - \overline{V}_i\right)^2}}{\left(V_{max,i} - V_{min,i}\right)} < \sigma_{V\,min}$$

In this equation, the non-dimensional standard deviation, σ_V, for all design variables in the population is measured with respect to the average design variable in the population.

$$\overline{V}_i = \frac{1}{N_{POP}} \sum_{j=1}^{N_{POP}} P_{i,j} \qquad \text{for } i = 1,\ldots,N_{VAR}$$

When the design variance in the population becomes less than the limit, the GA is switched to the DFP. The reasoning is that the population variance is contracting around a minimum and the DFP can be used to quickly home in on that minimum. The order of the switching is DFP – NM – SA.

Objective RMS Limit Rule

This is similar to the aforementioned rule, except that the variance in the objective function is computed rather than the variance in the design variables.

$$\sigma_F = \frac{1}{N_{POP}} \frac{\sqrt{\sum_{j=1}^{N_{POP}} \left(F_j - \bar{F}\right)}}{\left(F^* - F^0\right)}$$

Here, the F_j is the objective function value of the jth population member and the average objective function of the population is computed from them.

$$\bar{F} = \frac{1}{N_{POP}} \sum_{j=1}^{N_{POP}} F_j$$

The difference between this rule and the design variance limit is that the population may be dispersed over a wide design space but each may have very similar objective function values. This can occur is the objective function space is a large flat area with little or no gradient. The program is then switched first to the SA method (SA – NM – DFP).

Bad Mutation Rule

The average objective function value of the population is also used as a switching criterion of its own for the GA. The bad mutation rule causes the GA to switch to the NM if the average objective function increases from the previous optimization cycle with the GA. This will most likely occur if the random mutation rate is too large or if it produces one or more really bad designs. Since the NM specializes in bringing the poorest design within the centroid of the best designs, it is the most obvious first choice (NM – SA – DFP).

Lost Generation Rule

The GA develops sequential populations of new 'child' designs that are entered into the population only if the population size is allowed to increase, or if the 'child' design is better than the worst member in the population. If no new designs are entered into the population, the GA fails. This lost generation causes the program to switch to the SA algorithm (SA – NM – DFP).

Stall Rule

The NM simplex searching algorithm has only one failure mode, stalled. The NM is said to stall whenever the line search (produced by the direction from the worst design in the population through the centroid of the best designs) fails to improve itself so that it is better than the second worst member in the population.

$$F^*_{N_{POP}} \geq F_{N_{POP}-1}$$

This rule causes the hybrid optimizer to switch to the DFP method (DFP – GA – SA). If the best design in the population was generated by the DFP, then the DFP is passed by and the GA takes over.

Insufficient Random Energy Rule

The purpose of the SA algorithm is to add energy into the optimization system and allow it to escape the local minimums. The objective functions of the design variables in the population might get worse. Also, the random energy added to the system allows for the modification of any member in the population. Therefore, the capabilities of the SA would be wasted if the optimization process was switched to some other algorithm in case when any one objective function value became worse. It was also found that the SA terminates prematurely when the worse average objective criterion was met. The SA has, therefore, been programmed to end whenever the cooling protocol did not add sufficient energy into the system. The insufficient random energy criterion can be stated as follows.

$$\sqrt{\sum_{j=1}^{Npop} \left(\tilde{F}_j - \bar{\tilde{F}}\right)^2} < \Delta F_{min}$$

Here, the algorithm ends whenever the variance of the objective function values with added random energy, $\tilde{F}$, are less than a user-specified limit. The program is switched from the SA to the DFP method (DFP – GA – NM). After several cycles, the random energy in the SA may have been reduced to a negligible amount, while the insufficient random energy criterion might not be met because of the large variance in the population. Therefore, the SA switches to the NM method (NM – GA – DFP) after $\tilde{K}_{max}$ optimization cycles.

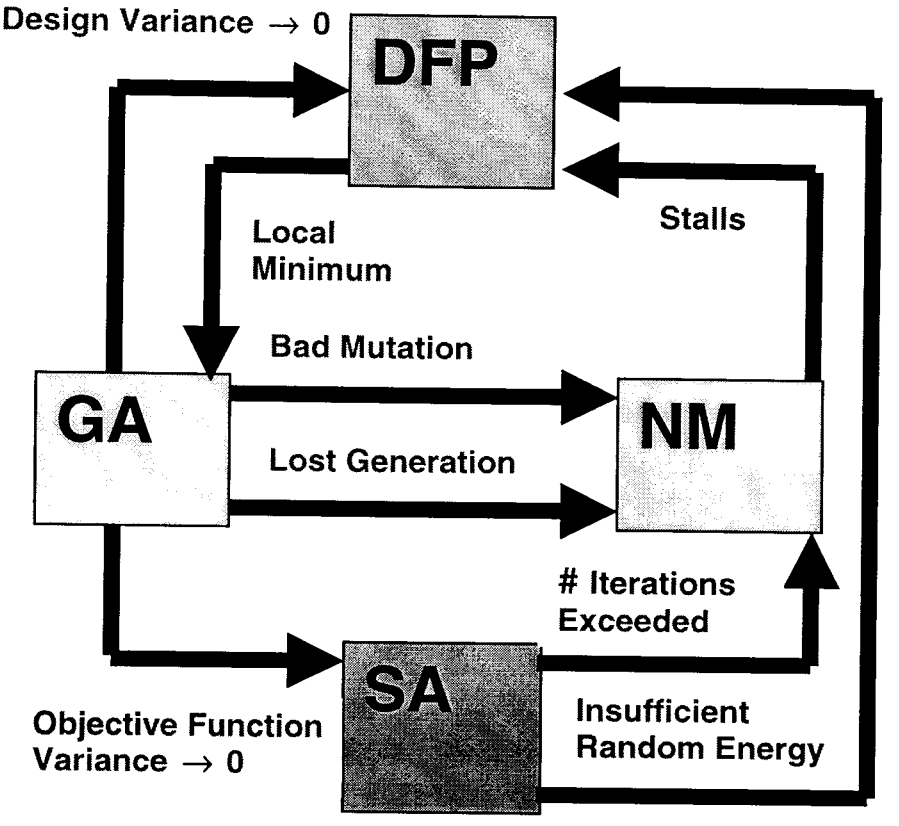


Figure 1. Flowchart of automatic switching among modules in a hybrid optimizer.

Aero-Thermal Optimization of a Gas Turbine Blade

Internal cooling schemes of modern turbojet and turbofan engines bleed air from the compressor and pass this air into the serpentine coolant flow passages within the turbine blades. The maximum temperature within a turbine blade must be maintained below a certain value in order to avoid thermal creep, melting, and oxidation problems. Increased coolant heat transfer and increased coolant flow rate directly decrease the amount of air delivered to the combustor and increase specific fuel consumption. Thus, the coolant mass flow rate and the coolant supply pressure should be minimized, while maintaining the inlet temperature of hot gases as high as possible and temperature in the blade material below a specified limit. These objectives can be met (Dulikravich and Martin, 1995; 1996; Martin and Dulikravich, 1997; Dulikravich et al. 1998; Martin et al., 1999) by the constrained optimization of the coolant passage shapes inside the turbine blade.

Geometry Model of the Turbine Blade Coating and Coolant Flow Passages

The outer shape of the blade was assumed to be already defined by aerodynamic inverse shape design or optimization. It was kept fixed during the entire thermal optimization procedure. The thermal barrier coating thickness was described by a wall thickness function versus the airfoil contour following coordinate, s. The metal wall thickness variation around the blade was also defined by a piecewise-continuous beta-spline (Barsky, 1988). The number of coolant flow passages in the turbine blade was kept fixed.

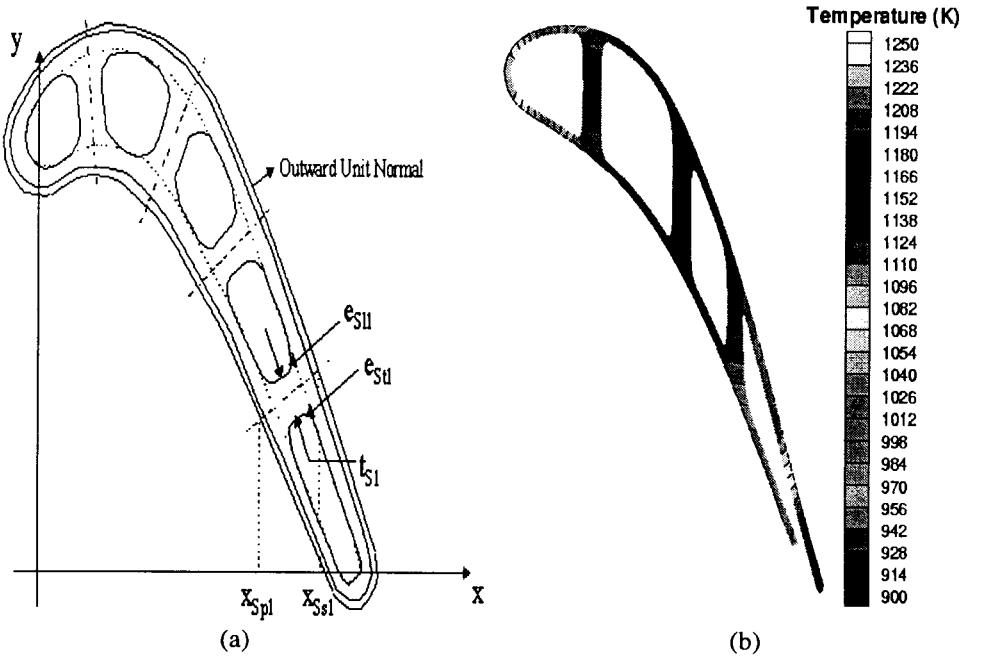


Figure 2. (a) A sketch of turbine airfoil, coating and coolant passage geometry. (b) Temperature field computed on the initial guess geometry used for the minimization of coolant temperature at the triling edge ejection location.

The x-coordinates of the intersections of the centerlines of each of the internal struts with the outer turbine airfoil shape were defined as x_{Ssi} and x_{Spi}, for the suction and pressure sides of the blade, respectively. The range over which each strut could vary was specified. In addition to the coordinates of the strut intersections, the strut thickness, t_{Si}, and a filleting exponent on either the trailing or leading edge sides, e_{Sti} and e_{Sli}, respectively, were used to complete the geometric modeling of each strut (Figure 2a). The strut fillets were described by a super-elliptic function that varied from a circular fillet ($e_{Si} = 2$) to an almost sharp right angle ($e_{si} \rightarrow \infty$).

The boundary conditions and constraints were: hot gas pressure at the inlet to the blade row (588131 Pa), hot gas pressure at the exit of the blade row (134115 Pa), hot gas inlet Mach number (0.1772), initial guess for the coolant mass flow rate (0.025 kg/s), initial guess for the hot gas inlet temperature (1592.6 K), thermal conductivity of thermal barrier coating (1.0 W/m K), thermal conductivity of the blade metal (30.0 W/m K), thermal barrier coating thickness (100 microns), maximum allowable temperature in the blade material (1250.0 K).

Total number of design variables per section of a three-dimensional blade was 27. These variables were: eight beta-spline control points defining coolant passage wall thickness, six strut end-coordinates (two per strut), three strut thicknesses (one per strut), six strut filleting exponents (two per strut), four relative wall roughnesses (one per each coolant flow passage). Two additional global variables were: one mass flow rate, and one inlet turbine hot gas temperature. The initial guess geometry is depicted in Figure 2b.

Turbine Cooling Scheme Optimization for Minimum Coolant Ejection Temperature

The uniform temperature and heat flux extrema objectives were not entirely successful (Dulikravich et al. 1998a; Martin et al., 1999) at producing internal blade geometry that would be considered economical or even physically realizable. The minimization of the integrated hot surface heat flux objective tried to eliminate the coolant passages (Martin and Dulikravich, 1997). The maximization of the integrated hot surface heat flux thinned the walls and produced an extreme range of temperatures that questioned the validity of the use of heat transfer coefficients on the outer surface of the blade.

Therefore, another objective was formulated (Martin et al., 1999) that minimizes the coolant temperature at the very end of the coolant flow passage (the ejection slot at the blade trailing edge). This is an indirectly formulated objective since mass flow rate was used as a design variable and could not be simultaneously used as the objective function. The reasoning was that reduced heat transfer coefficients require lower surface roughness on coolant passage walls therefore resulting in lower coolant supply pressure requirements. Thus, compressor bleed air can be extracted from lower compressor stages in which the air is cooler. This in turn should lead to the lower coolant mass flow rate requirement.

First, a turbulent compressible flow Navier-Stokes solver was used to predict the hot gas flow-field outside of the blade subject to specified realistic hot surface temperature distribution. As a byproduct, this analysis provides hot surface normal temperature gradients thus defining the hot surface convection heat transfer coefficient distribution. This and the guessed coolant bulk temperature and the coolant passage wall convection heat transfer coefficients creates boundary conditions for the steady temperature field prediction in the blade and thermal barrier coating materials using fast boundary element technique. The quasi-one-dimensional flow analysis (with heat addition and friction) of the coolant fluid dynamics was coupled to the detailed steady heat conduction analysis in the turbine blade material. By perturbing the design variables (especially the variables defining the internal blade geometry) the predicted thermal boundary conditions on the interior of the blade will be changing together with the coolant flow parameters. As the optimization algorithm ran, it also modified the turbine inlet temperature. Once the turbine inlet temperature changed significantly, the entire iterative procedure between the thermal field analysis in the blade material and the computational fluid dynamic analysis of the external hot gas flow-field was performed again to find a better estimate for thermal boundary conditions on the blade hot surface. This global coupling process was performed only a small number of times during the course of the entire optimization. This semi-conjugate optimization uses sectional two-dimensional blade hot flow-field analysis and a simple quasi one-dimensional coolant flow-field analysis. Consequently, it requires considerably less computing time than would be needed if a full three-dimensional hot gas flow-field and coolant flow-field analysis (Stephens and Shih, 1997) would be used.

Two different minimizations of the coolant ejection temperature were performed; one with the maximum temperature equality constraint, $T_{max} = \overline{T_{max}}$, and the other with the inequality constraint $T_{max} < \overline{T_{max}}$. The design sensitivity gradients were calculated using finite differencing and the optimization program was initiated with the DFP. The optimization with the equality constraint required 15 cycles and 794 objective function evaluations. The large number of function evaluations was needed because forward finite differencing was used to obtain the sensitivity gradients. After 4 optimization cycles, the program switched to the GA, switching finally to the NM in the 14th cycle.

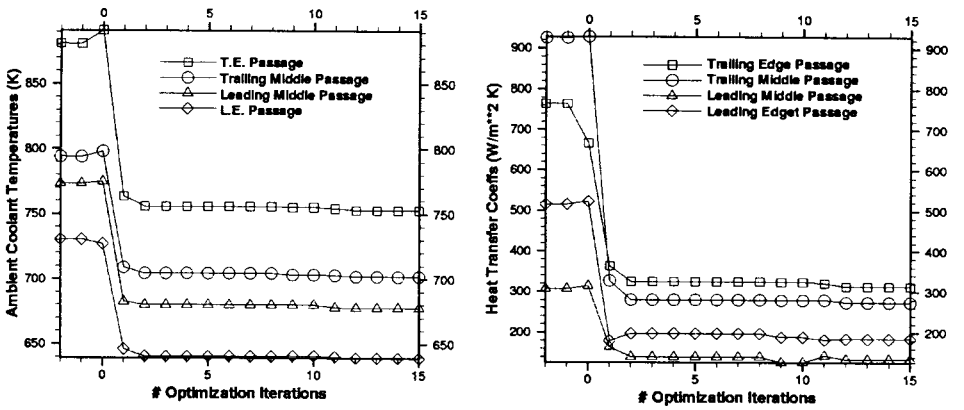


Figure 3. Evolution of optimum bulk coolant temperatures (a) and the coolant passage wall heat transfer coefficients (b) in each of the four coolant flow passages for the minimization of coolant ejection temperature when using the maximum temperature equality constraint.

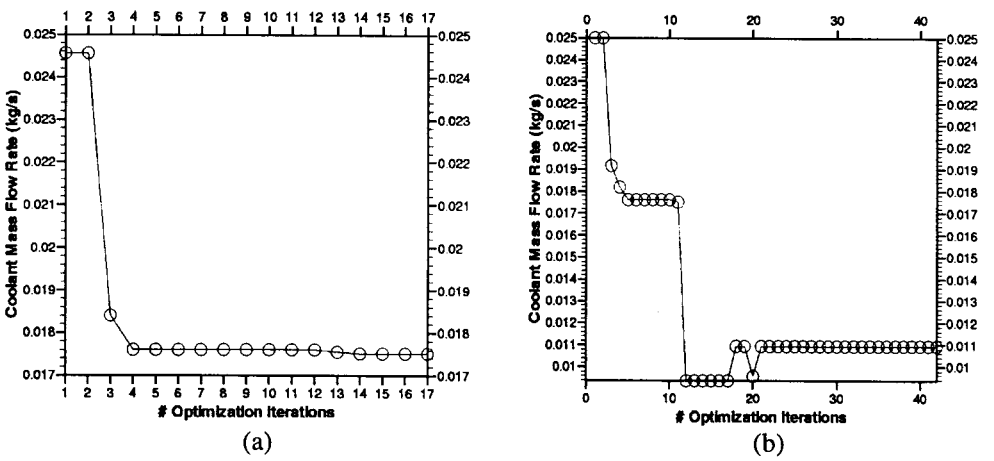


Figure 4. Evolution of coolant mass flow rate for the minimization of coolant ejection temperature using: a) the maximum temperature equality constraint, and b) the maximum temperature inequality constraint.

The reduction in coolant bulk temperatures (Figure 3a) also significantly reduced the coolant pressure losses and the coolant convection heat transfer coefficients (Figure 3b). This suggests that it might be possible to remove the heat transfer enhancements (mechanical wall turbulators) such as trip strips and impingement schemes from the coolant passages thus leading to a substantial reduction in turbine blade manufacturing costs. The ultimate goal (reduction in the coolant mass flow rate) was achieved (Figure 4a) by reducing the heat transfer coefficients and by making the passage walls thinner (Figure 5a). It should be pointed out that the turbine inlet temperature changed very little when the maximum temperature equality constraint was enforced with this objective.

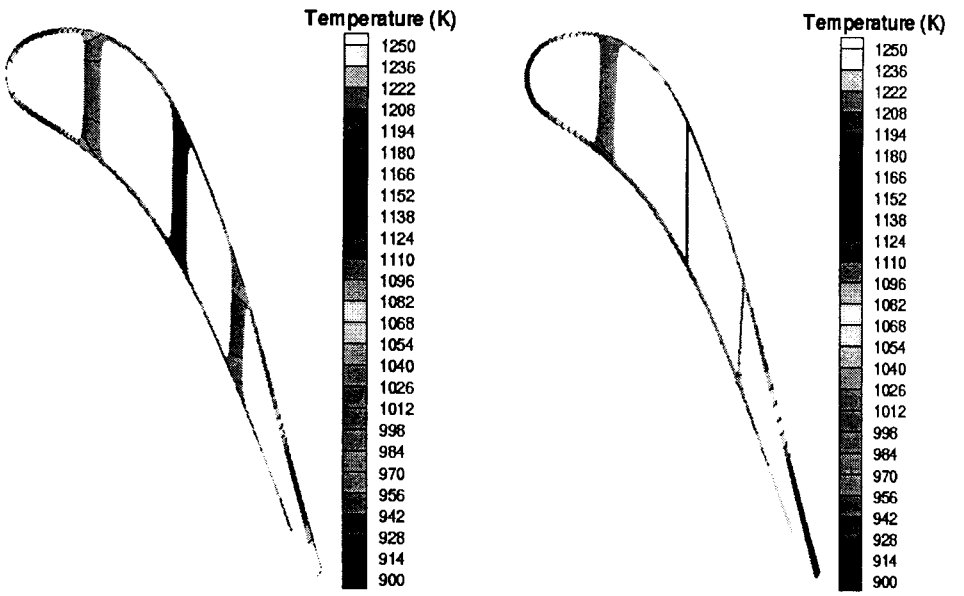


Figure 5. Optimized blade interior geometries and temperatures for minimized coolant ejection temperature design of an internally cooled turbine blade using; a) the maximum temperature equality constraint, and b) the maximum temperature inequality constraint.

But, when the maximum temperature inequality constraint was enforced, the coolant mass flow rate was reduced even more dramatically (Figure 4b) but the turbine inlet temperature decreased from 1600 K down to 1340 K which is unacceptable. The final optimized configuration had extremely thin walls and struts (Figure 5b) which were nearly at the lower limits enforced by the bounds on their design variables. This configuration is clearly unacceptable because of the reasonable doubts that such a thin walled blade could sustain the mechanical stresses. It is interesting to note that the optimization with the maximum temperature inequality constraint ran for more cycles (40), but required fewer objective function evaluations (521). This was because of the fewer number of gradients needed of the inequality constraint functions. That is, the equality constraint was always active, but the inequality constraint was only active when the maximum temperature in the blade was equal to or greater than the target maximum temperature, $\overline{T_{max}}$.

It can be concluded that even a seemingly minor change in the constrained multidisciplinary (aero-thermal) optimization can have a profound influence on the acceptability of the optimized design. It can also be concluded that the hybrid constrained optimizer switching logic proved to be robust and efficient when solving a realistic multidisciplinary problem.

Hypersonic Shape Optimization

Two main objectives during shape optimization of hypersonic missile shapes are lowering of the aerodynamic drag and lowering of the aerodynamic surface heating. The aerodynamic analysis was performed using an extremely simple model known as modified Newtonian impact theory (MNIT). This is an algebraic equation which states that the local coefficient of aerodynamic pressure on a body surface is linearly proportional to the square of the sine of the angle between the body tangent at that point and the free stream velocity vector. The MNIT is known to give embarrassingly accurate predictions of integrated aerodynamic lift and drag forces for simple body shapes at hypersonic speeds. It is often used instead of more appropriate (and orders of magnitude more computationally expensive) physical models based on a system of non-linear partial differential equations known as Navier-Stokes equations. In the following examples, the MNIT was used for the flow field analyses. Two optimization techniques were included: the constrained DFP and the constrained GA technique, both using Rosen's projection methodology for improved equality constraint treatment.

Optimum Ogive Shaped Missile

First, results from the program were verified against analytically known solutions. Specifically, the geometry of an axisymmetric body was optimized to reduce compression wave drag at zero angle of attack. Optimal bodies of revolution that minimize drag have previously been analytically determined. Two such solutions are known as the Von-Karman and Sears-Haack bodies (Ashley and Landahl, 1965). These two bodies yield the minimum wave drag under two different sets of constraints. The Von-Karman body assumes that the body terminates with a flat plane, that the base area in this plane is known, and that the total length of the body is specified. The Sears-Haack body assumes that the body is pointed at both ends, and that the total volume and length of the body are given.

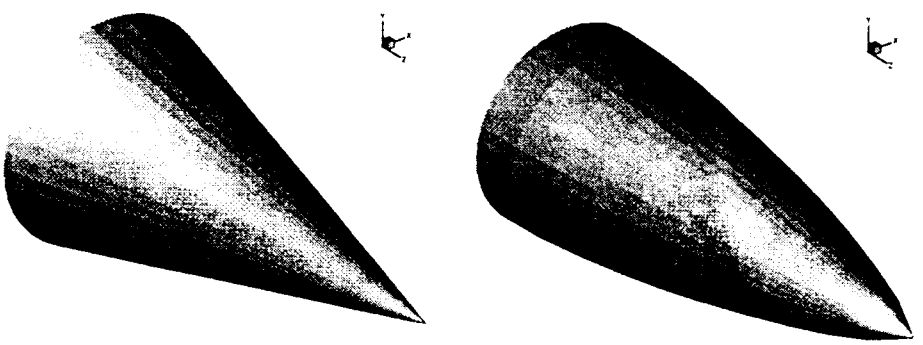
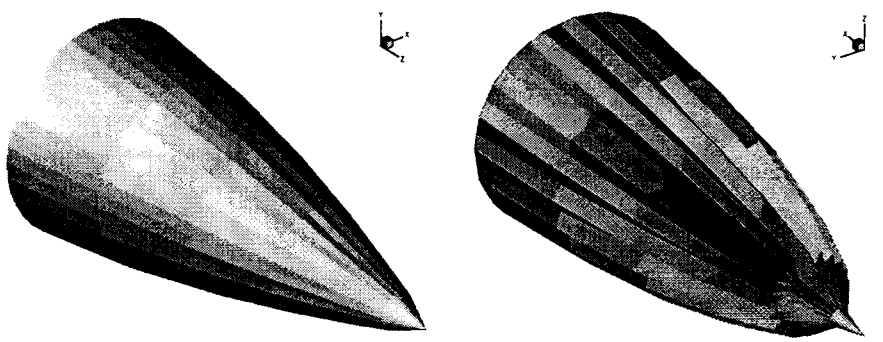


Figure 6. Initial cone configuration (a) and optimized ogive shape (b) at the onset of longitudinal 'ridges'. Constrained DFP optimizer was used with MNIT as the analysis code.

The constrained DFP optimizer was used to determine computationally the body of revolution that minimizes wave drag at Mach = 10 and an altitude of 18 km. Initially, the body was specified to be a 10-meter-long, 15-degree angle right-circular cone (Figure 6a). The design variables for this exercise were specified to be the radii of the body at 10 cross sections. Each design variable (the cross sectional radii) was allowed to vary from 0 to 10 meters. During the optimization process, the length was kept fixed and the total volume of the body was constrained (with an equality constraint) not to change by more than 1.0 cubic meter from its initial value of 75.185 cubic meters. The constrained optimization process converged to the 'bulged' axisymmetric body called an *ogive* (Figure 6b). The base area of the optimized body, and the total volume (fixed) were then used to compute Von-Karman and Sears-Haack bodies from analytical expressions (Anderson, 1989). The numerically optimized body was in excellent agreement with the analytically optimized body shapes (Foster and Dulikravich, 1997).

Optimum Star Shaped Missile

All of the body surface nodes on the first cross section could move together radially and were controlled by one design variable. On the other five cross section planes, all of the 38 surface nodes had two degrees of freedom except for the two 'seam' points whose x-coordinate is zero (the points on the vertical plane of symmetry). These 'seam' points were allowed to move only vertically (in the y-direction) in their plane. Thus, there were 78 design variables per each of the five cross sections and one design variable (radius) at the sixth cross section giving a total of 391 design variable in this test case.



(a) (b)

Figure 7. The star shaped hypersonic missile obtained when using a constrained GA optimizer and MNIT as a flow-field analysis code: a) front view, and b) axiometric view.

The constrained DFP algorithm could not converge to anything better than the smooth surface ogive shape which happened around 30^{th} iteration. Therefore, the optimization was switched to a constrained GA algorithm. After the completion of the 45^{th} design cycle which is towards the end of the 'leveled-off' portion of the convergence history (approximately from design cycle 25 to cycle 48), a slightly perturbed ogive geometry was obtained (Figure 7a). Due to the use of GA, the convergence after that point again dramatically increased leading to the deepening of the body surface 'ridges', or 'channels', and the narrowing of the spiked nose. It is understandable that these channels appeared because the flow solver that was used for the optimization process was the MNIT.

According to the MNIT, the pressure on the surface of the body is solely a function of the local inclination angle with respect to the free stream. Therefore, the optimizer was able to reduce the pressure on all of the body surface panels (after the first cross section) by creating the star-shaped body (Figure 7b) which caused the outward normal vectors on each panel to rotate away from the free stream direction.

Since the use of MNIT in this shape optimization example causes every second point defining the body cross section to become a 'ridge', the number of optimized 'ridges' will always equal the half of the grid points defining the body cross section. This is strictly the result of the fact that an oversimplifying flow-field analysis model (MNIT) was used. It is interesting to note that star-shaped hypersonic projectile shapes have been studied experimentally for several decades, but the number of 'ridges' or 'fins' on these bodies was considerably smaller. That is, a more physically appropriate non-linear flow-field analysis code would have created a number of optimized 'ridges' that is considerably smaller than half the number of all the grid points defining a body cross section. This illustrates the possible dangers of using overly simplistic surrogate models for the evaluation of the objective function in the latter stages of the optimization process.

The final star cross section missile configuration obtained with the MNIT reduced the aerodynamic drag by 77 percent. When the optimized star missile shape was analyzed using the parabolized Navier-Stokes flow-field analysis code, the total drag of the optimized body was found to be 53 percent lower than the drag of the initial conical geometry. The difference between 77 and 53 percent in the reduction of the missile drag can be represented as a relative error of $(77 - 53)/53 = 45.3$ percent that is strictly due to the use of an overly simplistic surrogate model (MNIT) for the flow-field analysis. The shape optimization effort took 75 optimization cycles, and called the MNIT flow solver 60001 times. The execution took 4282 seconds on a Cray C-90 computer.

Maximizing Aerodynamic Lift/Drag

Next, an initial 10-meter long 15-degree cone was optimized to maximize the lift-to-drag (L/D) ratio at zero angle of attack using the hybrid gradient optimizer and the MNIT flow solver. The shape was optimized holding its length and volume fixed. Six cross sectional planes with forty surface nodes described the geometry. Every surface node was allowed to vary only radially on its cross sectional plane thus creating a total of 240 design variables. The DFP optimizer was executed until convergence was reached. Execution terminated after 40 design iterations that consumed 1458 CPU seconds on a Cray C-90 and required 19961 objective function analyses. The converged optimal lift-to-drag ratio was L/D = 1.29. Figures 8a shows the final optimized body that is cambered and has ridges that have formed on its upper surface. The optimizer cambered the body so that a greater surface area on the underside faced the free stream so as to increase lift, and formed ridges on top of the body so that downward pressure was minimized. Yet the body still has an ogive appearance, which helps to reduce overall drag. At this point (after the DFP optimizer had converged), the optimization method was changed to the GA optimizer which continued to further reduce the cost function. The execution of the GA was stopped after it had performed an additional 52 design cycles that required only an additional 748 objective function analyses. The converged optimized lift-to-drag ratio was L/D = 1.54. The important fact learned from this case is that the constrained GA optimization technique was able to significantly reduce the cost function after the gradient based constrained DFP had converged and could go no further. Not only did the constrained GA optimizer reduce the cost function, but it did so while performing fewer objective function analyses.