Numerical shape optimization using CFD

Praveen. C
TIFR Center for Applicable Mathematics, Bangalore, India
Email: praveen@math.tifrbng.res.in

Numerical shape optimization using PDE-based models like CFD is a promising technology. In this paper, we discuss some aspects of shape optimization as applied to aerodynamic problems. Optimization of wing shapes for aerodynamic performance is presented using a combination of particle swarm method and surrogate models. The wing shape deformations are parameterized using free form deformations. The developed strategy is applied to some test cases to show its effectiveness in solving optimization problems.

Keywords: Particle Swarm Optimization, Free Form Deformation, Surrogate Models, Shape optimization

1 Introduction

Numerical optimization techniques can be broadly classified into gradient-based and gradient-free methods. Gradient-based methods are rooted in differential calculus which provides a way to characterize a local optimum solution in terms of the gradient of the function, which must vanish at a local optimum provided the function is differentiable. The most basic of these methods is the steepest-descent method which takes steps in the direction opposite to the gradient in order to decrease the function value. It is necessary to compute the gradient of the function of interest which can be a difficult task for problems governed by PDE. The engineering approach is to use finite differencing while treating the numerical PDE model as a black-box. There are many problems with this approach, chief of them being the accuracy problem of the finite difference formula and noisy objective functions. The other, more mathematical approach is to use adjoint techniques to compute the gradient. While this is a good approach when it can be accomplished, it involves considerable code development to generate an adjoint solver. Moreover, there are consistency issues of adjoint method when dealing with empirical models like turbulence models and discontinuous solutions. Gradient-based methods have the drawback of giving only a local optimum solution, which may be undesirable especially in a preliminary design phase.

The second class of optimization methods are gradient-free, i.e., they operate with only the function values and can be characterized as stochastic search methods. The best known example is the Genetic Algorithm (GA), with many other techniques being added to this class recently, like Particle Swarm Optimization (PSO). These methods have the potential to find the global optimum. Since only the function value is required, the PDE-based analysis tools like CFD

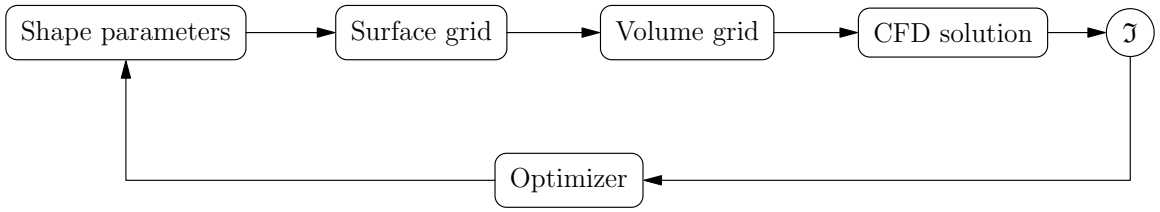


Figure 1: Shape optimization chain

and FEM can be used as black-boxes to easily develop an optimization approach. Most of these methods are population-based, i.e., they work with a collection of candidate designs in any iteration. These designs are then evolved under certain rules into new populations; the evolution rules are designed so that a good search of the design space is performed. Due to the use of population of designs and search character, they have a high computational cost and are slow to converge. Their efficiency can be improved using surrogate models and parallel computing, which makes them feasible to use in solving practical engineering problems.

Optimization problems in which shape plays an important role occur in many areas including aerodynamics. The most common example is the problem of designing a wing shape in order to minimize drag or maximize lift, under some other constraints. Any shape optimization problem based on CFD as the analysis tool involves the following elements.

1. A method to parameterize the shape, i.e., represent the shape in terms of a set of parameters.
2. A method to generate a grid on the shape and in the interior volume
3. A CFD tool to compute the flow around the shape
4. And finally an optimizer which changes the shape to improve the design

These four tools must be coupled in an automatic chain as shown in figure (1). Once the optimization is started the user cannot interfere in this chain and hence it is important that all the elements in this chain must be robust. In the rest of the paper, we discuss some aspects of these four tools that are required to perform shape optimization. Finally, some examples will be presented to illustrate the capabilities of the shape optimization framework presented.

2 Shape parameterization

When dealing with airfoils and wings, the standard approach is to characterize them in terms of some fundamental parameters like leading edge radius, maximum camber, location of maximum camber, taper ratio of wing, etc. These shapes and also more complicated ones can be represented using CAD approaches like BSplines/NURBS. A problem with all these methods is that once the shape is modified, it is necessary to re-generate the surface/volume grids for the new shape. This task can be difficult to automate if one is dealing with complex problems like RANS, where it is difficult to automatically generate good quality grids without user intervention. Instead of re-generating the grid for every new shape, it is also possible to deform a reference grid to correspond to the new shape using radial basis functions. In this case one

obtains a method that is independent of the shape parameterization and type of grid, and is usually referred to as a free-form approach.

An extreme version of this approach is the free-form deformation (FFD) method which was originally developed in computer graphics [7]. The basic idea is to embed the shape inside a box and deform the box. One can think of the box as being made of rubber with the shape inside it. When the rubber is squeezed the shape is also squeezed and deformed. This technique can be applied directly to the CFD grid and is independent of the way the original shape is represented. In fact the shape and the CFD grid are simultaneoulsy deformed, the technique being independent of the type of grid also. In a comparison of various shape parameterization techniques, Samareh shows that FFD scores well on most of the desirable properties of a parameterization technique.

In practice, the FFD technique is implemented as follows. There is a reference shape, which is usually the starting shape, for which a CFD grid is generated. For any grid point P, let $X^0(P)$ denote its coordinates in the reference grid. Under the free-form deformation, its coordinates are given by

$$X(P) = X^0(P) + \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} Y_{ijk} B_i^{n_i}(\xi_p) B_j^{n_j}(\eta_p) B_k^{n_k}(\zeta_p)$$

where $B_i^{n_i}$ etc. are Bernstein polynomials. Note that any set of basis functions that span the space of continuous functions can be used in place of Bernstein polynomials. The design variables in FFD are the coefficients Y_{ijk}, also refered to as control points. In case of wing shape optimization, the control points are moved only normal to the chord direction. For wing shapes, additional design variables like geometric twist can be added separately. Note that in this approach we are not parameterizing the shape, but only the shape deformations. Since Bernstein polynomials are smooth functions, the shape deformations are smooth; the shape retains its original smoothness even under deformation.

3 Particle Swarm Optimization

PSO was inspired by the behaviour of swarms of animals like ants, bees and birds. In nature, such swarms are found to be capable of accomplishing complex tasks like finding the source of a food, building complex structures, avoiding predators, etc. This complex behaviour is made possible due to the cooperative interaction of a large number of individuals through simple rules. This idea of swarm intelligence is used to create an algorithm for minimizing a function as follows. Consider the following problem:

$$\min_{x \in D} J(x), \quad D \subset \mathbb{R}^d$$

In PSO, a number N_p of particles are initially randomly distributed in the design space D. The position coordinates $x_i \in D, \quad i = 1, ..., N_p$ of each particle denotes one design, and has an associated value of the objective function J. Each particle is assigned a velocity vector $v_i \in \mathbb{R}^d, \quad i = 1, ..., N_p$ with which it wanders around in the design space looking for better positions, i.e., positions with smaller value of the objective function. Each particle remembers the best position it has discovered in its lifetime, called local memory, at time t the local memory is given by

$$p_i^t = \underset{0 \le s \le t}{\operatorname{argmin}} J(x_i^s)$$

There is also a global memory, i.e., the best position discovered by the whole swarm which is given by

$$p^t = \underset{i}{\operatorname{argmin}}\, J(p_i^t)$$

and is known to all the other particles; this brings in the cooperative nature of the algorithm. The velocity of each particle is updated as follows:

$$v_i^{t+1} = \omega v_i^t + \underbrace{c_1 r_1^t \otimes (p_i^t - x_i^t)}_{Local} + \underbrace{c_2 r_2^t \otimes (p^t - x_i^t)}_{Global}$$

This leads to a competition between local exploration of the design space through the local memory term and global exploration through the global memory term. The velocity of each particle is updated in an obvious way

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

This simple algorithm can be easily implemented, though the computational cost can be quite high, especially for PDE-based analysis tools like CFD. For high dimensional problems, a large number of particles may be required to perform an effective search. The use of parallel computing can reduce the wall time to some extent, but it still requires large computational resources.

4 Surrogate models

The use of PDE-based models is computationally expensive due to the need to generate a grid and solve the PDE using an iterative technique. For optimization, we only need a small number of integral quantities like lift, drag, etc. The idea of surrogate models is to build a cheap model only for these quantities of interest. Initially, the design space is sampled at a number of locations and the objective function value is evaluated at these locations, which are then stored in a database. This information is used to construct an approximation to the function using various techniques like response surfaces, neural networks, radial basis functions, kriging, etc. This is however a non-trivial task when dealing with large number of design variables, and the available information is usually very sparse. Polynomial based models do not scale well to large number of variables, and are not good at modeling highly non-linear functions. Also, a good model must be able to provide an estimate of the error in the predicted value, so that the model can be improved in regions of large error.

4.1 Kriging

Kriging, also known as Gaussian random process model, is a very promising technique that is capable of constructing useful models in high dimensional spaces and also provides an error estimate. In the following, we adopt the Bayesian viewpoint of Gaussian processes [2, 8]. The problem is to find an approximation $\tilde{f}(x)$ to an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ given its values $f_n = f(x_n)$, $n = 1, \ldots, N$ at N distinct points; we denote the data as $F_N = \{f_1, f_2, \ldots, f_N\} \subset \mathbb{R}$ and $X_N = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^d$. The vector of known function values F_N is assumed to be one realization of a multivariate Gaussian process with joint probability density

$$p(F_N|X_N) = \frac{\exp\left(-\frac{1}{2} F_N^{\mathrm{T}} C_N^{-1} F_N\right)}{\sqrt{(2\pi)^N \det(C_N)}} \tag{1}$$

where C_N is the $N \times N$ covariance matrix. The element C_{mn} of the covariance matrix gives the correlation between the function values f_m and f_n. This is expressed in terms of a covariance function c, i.e., $C_{mn} = c(x_m, x_n)$. When adding a new point x_{N+1}, the resulting vector of function values F_{N+1} is assumed to be a realization of the $(N+1)$- variable Gaussian process with joint probability density

$$p(F_{N+1}|X_{N+1}) = \frac{\exp\left(-\frac{1}{2}F_{N+1}^{\mathrm{T}}C_{N+1}^{-1}F_{N+1}\right)}{\sqrt{(2\pi)^{N+1}\det(C_{N+1})}} \tag{2}$$

Using the rule of conditional probabilities $p(A|B) = \frac{p(A,B)}{p(B)}$ we can write the probability density for the unknown function value f_{N+1}, given the data (X_N, F_N) as

$$p(f_{N+1}|X_{N+1}, F_N) = \frac{p(F_{N+1}|X_{N+1})}{p(F_N|X_N)} \tag{3}$$

In order to simplify this expression we notice that the $(N+1)$-variable covariance matrix C_{N+1} can be written as

$$C_{N+1} = \begin{bmatrix} C_N & k \\ k^{\mathrm{T}} & \kappa \end{bmatrix}$$

where

$$k = [c(x_1, x_{N+1}), c(x_2, x_{N+1}), \ldots, c(x_N, x_{N+1})]^{\mathrm{T}}, \qquad \kappa = c(x_{N+1}, x_{N+1})$$

This gives

$$C_{N+1}^{-1} = \begin{bmatrix} M & m \\ m^{\mathrm{T}} & \mu \end{bmatrix}$$

where

$$M = C_N^{-1} + \frac{1}{\mu}mm^{\mathrm{T}}, \quad m = -\mu C_N^{-1}k, \quad \mu = (\kappa - k^{\mathrm{T}}C_N^{-1}k)^{-1}$$

Then, the probability density for the function value at the new point is

$$p(f_{N+1}|X_{N+1}, F_N) \propto \exp\left[-\frac{(f_{N+1} - \hat{f}_{N+1})^2}{2\sigma_{f_{N+1}}^2}\right]$$

where

$$\hat{f}_{N+1} = k^{\mathrm{T}}C_N^{-1}F_N, \qquad \sigma_{f_{N+1}}^2 = \kappa - k^{\mathrm{T}}C_N^{-1}k \tag{4}$$

Thus the probability density for the function value at the new point x_{N+1} is also Gaussian with mean $\hat{f}_{N+1}$ and standard deviation $\sigma_{f_{N+1}}$. The mean value is taken as the estimate given by the model for the function value at the new point x_{N+1}. The availability of the variance of the estimated function value is an important advantage of this method, since it provides an estimate of the error in the approximated function value.

4.2 Merit functions

A surrogate model must be reasonably accurate to represent the global trends in the function. The but the final goal is to find a minimum of the objective function J and not to construct the most accurate surrogate model. Anyway, in large dimensional spaces, it is impossible to construct a uniformly accurate model in a realistic time, since one is faced with the *curse of*

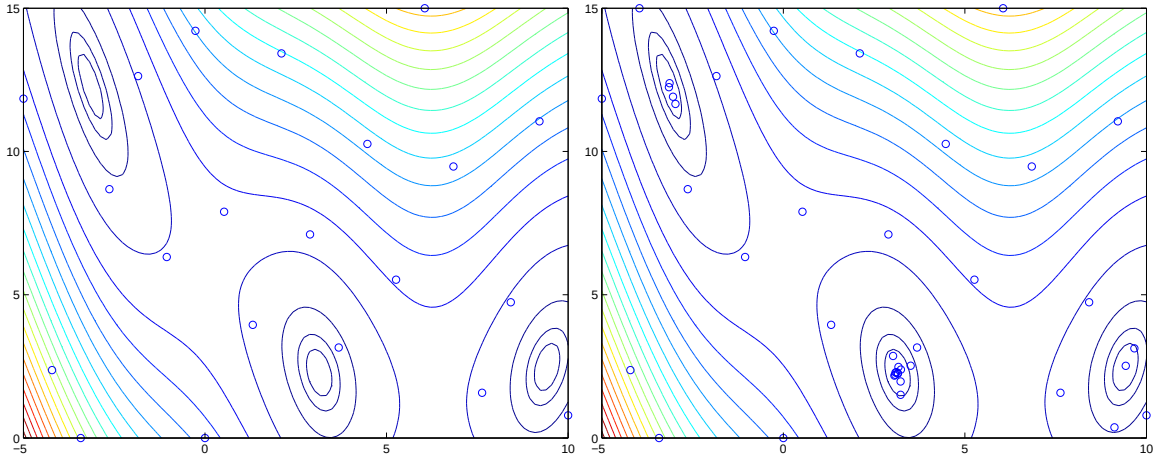


Figure 2: Merit function based minimization of Branin function. Exact function contours are plotted. The initial sampling is shown as circle on left. After 20 iterations, the sampled points are shown on right.

dimensionality. It is more important that the model should be accurate in promising regions of design space where minima could be located. There is thus a need to balance exloration and exploitation. Exploration refers to sampling all regions of design space so that the model accuracy improved everywhere. This could be achieved by adding more samples in regions where the standard deviation σ of the kriging predictor is large. Exploitation refers to performing a greater sampling around promising regions, i.e., regions where $\tilde{J}$ is small. These two objectives can be represented by defining certain merit functions. One particular choice is to minimize a statistical lower bound defined as

$$\min_x J_\kappa(x) := \tilde{J}(x) - \kappa\sigma(x)$$

where κ controls the balance between exploration and exploitation. If $\kappa = 0$, then we are performing exploitation and while if κ is large, we would be performing exploration since more weight is given to error σ. In practice, a range of values of $\kappa = 0, 1, 2, 3$ can be used taking care of both aspects. The four merit functions J_0, J_1, J_2, J_3 are minimized and the resulting design vectors are evaluated on the exact model. The new function values are then added to the database and the metamodel is updated. This iterative process is repeated until the objective function stops decreasing or one has exhausted the computational budget. An example is shown in figure (2) where a 2-D function which has three glocal minima is used. Note that the optimization has added most of the new samples in the regions of the global minima. There are other merit functions that can be defined and the reader is referred to [4] for more information.

5 Examples of shape optimization

Two examples of shape optimization are discussed in this section. The first one uses FFD for shape changes while the second one uses radial basis function for grid deformation.

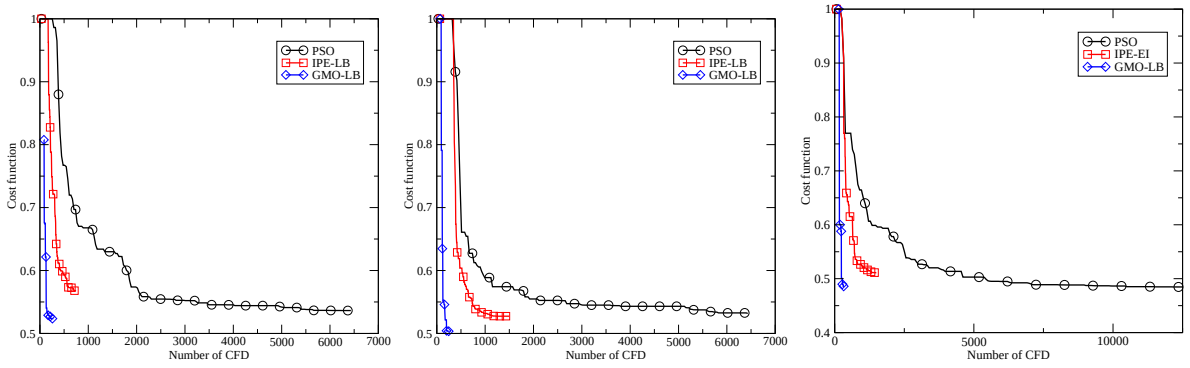
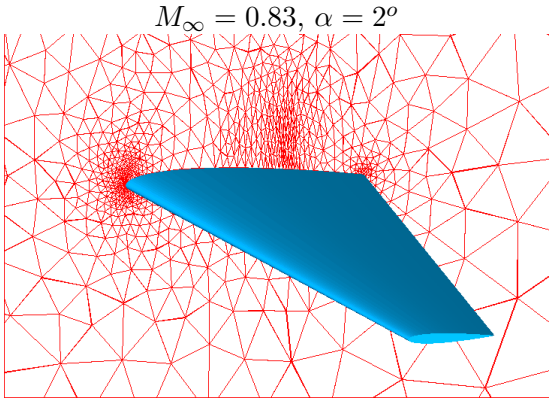


Figure 4: Convergence of objective function for transonic wing optimization

5.1 Transonic wing shape optimization

A transonic wing with inviscid flow model is considered in this case. The CFD consists of a finite volume code on unstructured tetrahedral grids and uses Roe flux. The grid consists of 31124 nodes and is shown in figure (3). The optimization problem is to minimize the drag under a constraint of lift. The constraint is implemented using penalty functions; the cost function to be minimized is defined as

$$\mathcal{J} = \frac{C_d}{C_{d_0}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_0}}\right)$$



(Piaggio Aero. Ind.)
Grid: 31124 nodes

Figure 3: Transonic wing and view of unstructured grid

An FFD parameterization of different sizes is used to study the performance of the metamodels. Figure (4) shows the convergence of the objective function for 8, 16 and 32 design variables. We see that when CFD is used for all the computations, the number of CFD evaluations is quite high, of the order of 1000s, while the use of metamodels leads to similar results using only about 100-300 CFD computations. An example of the change in flow structure is shown in figure (5). We notice that the strong shock in the initial wing is almost completely eliminated due to shape optimization. This example is discussed in more detail in [4].

5.2 Turbulent airfoil optimization

Transonic airfoils usually have a shock on the upper surface. The shock is to be mitigated by adding a bump to the upper airfoil surface. The size, shape and location of the bump can be determined in an optimal way so as to achieve maximum reduction in drag. In order to study this problem, we choose a test case of a laminar flow airfoil RAE5243 [5] and solve the problem

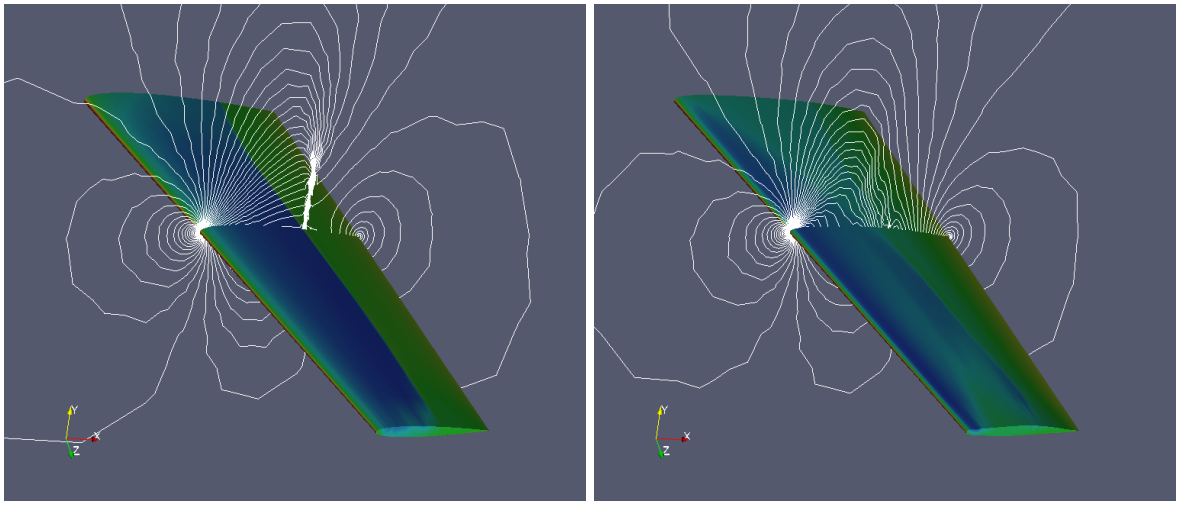


Figure 5: Pressure around transonic wing: initial shape (left) and optimized shape (right)

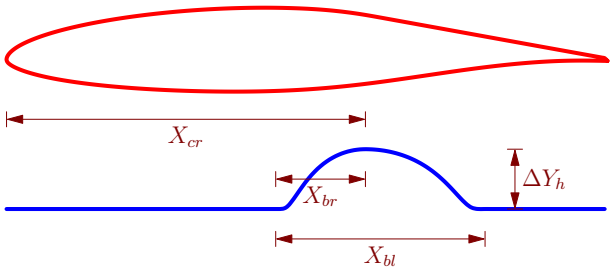


Figure 6: RAE5243 airfoil and parameterization of cubic bump

of minimizing the drag under a lift constraint

$$\min C_d \quad \text{subject to} \quad C_l = 0.82$$

The Reynolds number and Mach number for the test case are 19 million and 0.68 respectively, and we consider the case of fully turbulent flow. The turbulent flow is computed using a RANS code called NUWTUN [3] which is based on finite volume method on block structured grids. The inviscid fluxes are computed using Roe scheme together with MUSCL approach for higher order accuracy, while the turbulence is modeled with $k - \omega$ model.

5.2.1 Bump shape parameterization and grid deformation

The bump is assumed to be a cubic curve which is added to the upper surface of the airfoil in such a way that continuity of the first derivative of the airfoil surface is maintained. The bump shape and location is parameterized using four variables as shown in figure 6. The angle of attack α is also taken as a design variable since it is useful to change α to enforce the lift constraint $C_l = 0.82$.

The addition of a bump to the RAE5243 airfoil changes its shape; it is then necessary to modify the CFD grid to conform to the new shape. We use radial basis function (RBF) interpolation [1]

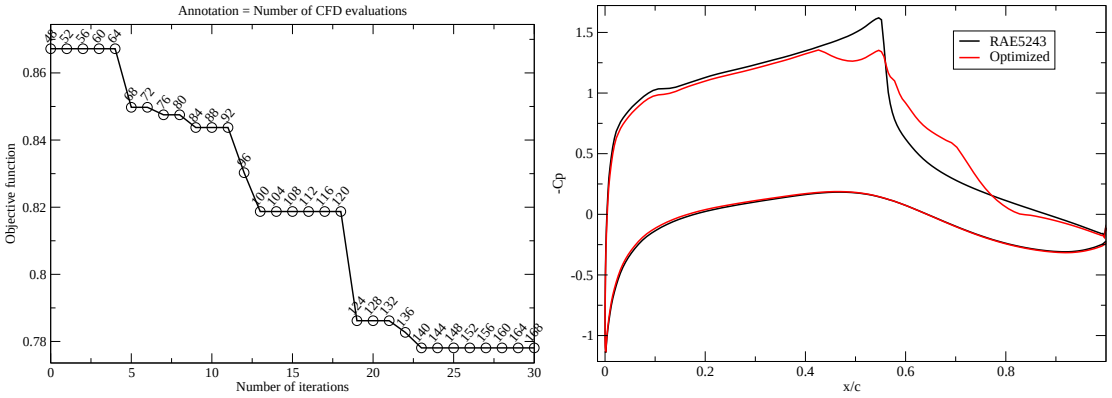


Figure 7: Convergence of objective function for shock control bump optimization, and the initial and final pressure distribution

to perturb the initial grid for RAE5243 airfoil to conform to the new shape. The displacement of the grid points on the airfoil are known by the addition of the bump, while the points on the outer boundary are held fixed. Using these displacements, an RBF interpolation for the displacement in the x and y coordinates is constructed using thin plate spline functions. The displacement for any interior grid point is then obtained using the RBF interpolation and leads to a smooth deformation of the grid.

5.2.2 Bump shape optimization

The drag force is minimized under lift constraint using the four shape parameters of the cubic bump and the angle of attack (five design variables). The constraint is enforced using penalty function approach; the function to be minimized is

$$\min \frac{C_d}{C_{d_0}} + 10^4 \max\left(0, 1 - \frac{C_l}{C_{l_0}}\right)$$

where $C_{l_0} = 0.82$ and C_{d_0} is the corresponding drag coefficient for the RAE5243 airfoil. We first construct a database of 48 design variables using LHS. At each optimization iteration, four different control parameters sets are tested, that correspond to $\kappa = 0, 1, 2, 3$, with a total of 30 iterations. The convergence of the objective function and the initial and final pressure distribution on the airfoil are shown in figure 7. The objective function is seen to converge in about 25 iterations and requiring about 140 RANS computations. The pressure distribution shows the weakening of the strong shock on the upper surface by the introduction of a weak compression zone, which is indicated by a small dip in the optimized pressure distribution. This is also shown in figure 8 where the strong shock is seen to be replaced by a compression wave at the foot of the shock, with the shock being broken into two weaker shocks.

Table 1 shows the forces for the optimized airfoil and these are also compared with the results from Qin et al. [6]. We note that the lift constraint is well satisfied and the drag coefficient is reduced by about 22%. This can be compared with the reduction of 18% obtained by Qin et al. [6] using a gradient-based method. The larger reduction in the drag obtained in the present work might be attributed to the use of a global optimization method. The drag is decomposed into pressure drag C_{d_p} and viscous drag C_{d_v}. Since the shock is weakened by the addition of a

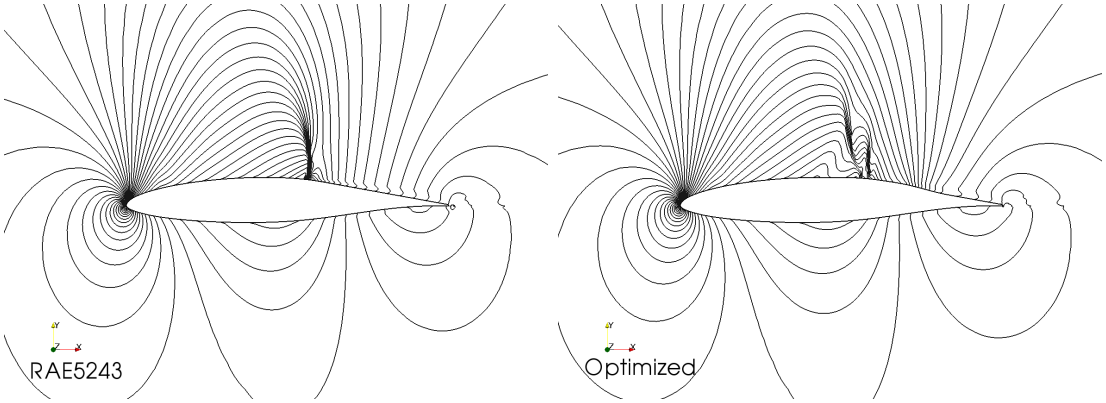


Figure 8: Pressure contours for RAE5243 airfoil and for optimized airfoil with bump

Case	C_d	ΔC_d	C_{d_p}	C_{d_v}	C_l	AOA
Present	0.01266	-22.2%	0.00680	0.00586	0.8204	2.19
Qin et al.	0.01326	-18.2%	0.00756	0.00570	0.82	-

Table 1: Forces for the optimized airfoil

bump, we see that the pressure drag is reduced by a large extent while the viscous drag increases slightly. This is consistent with the results of Qin et al. [6]. The bump shape parameters are shown in table 2 together with those from Qin et al. [6]. The two sets of parameters are different due to the different drag reductions; the higher drag reduction of 22% in our case corresponds to a taller bump which is expected. Figure 9 shows the initial and optimized shapes, and also shows a closeup view of the grid corresponding to the optimal shape.

6 Summary

Shape optimization for aerodynamic problems is discussed using gradient-free methods and surrogate models. The use of kriging which provides an estimate of error in predicted value leads to an efficient method to enrich the model in interesting regions of design space. For shape parameterization, FFD is an attractive alternative since it achieves both shape deformation and grid deformation in one single tool. Some examples of shape optimization using the proposed methodology is presented, demonstrating a reasonably efficient algorithm with realistic computational effort.

Case	X_{cr}	X_{bl}	X_{br}	$\Delta Y_h \times 10^{-3}$
Present	0.688	0.399	0.257	8.578
Qin et al.	0.597	0.313	0.206	5.900

Table 2: Optimized bump parameters

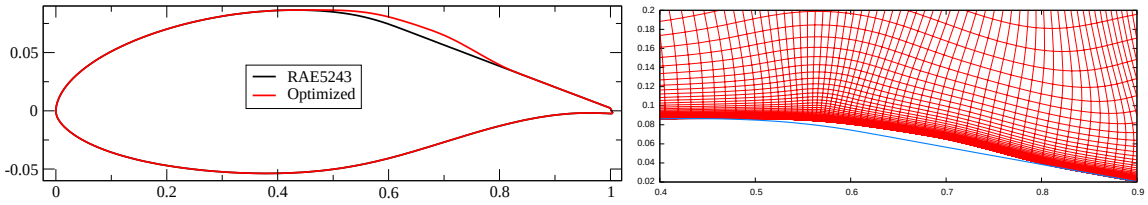


Figure 9: Initial and optimized shapes for shock control bump and grid corresponding to optimal shape

References

[1] K. Kumar and M. T. Nair. A mesh deformation strategy for multiblock structured grids. Technical report, National Aerospace Laboratories, Bangalore, 2007.

[2] D. Mackay. Gaussian processes: a replacement for supervised neural networks. Tutorial lecture notes for NIPS, 1997.

[3] NUWTUN. http://nuwtun.berlios.de.

[4] C. Praveen and R. Duvigneau. Study of some strategies for global optimization using Gaussian process models with application to aerodynamic design. Research Report RR-6964, INRIA, 2009.

[5] C. Praveen and R. Duvigneau. Ta5 test case using famosa, database workshop for multi-physics software validation, December 2009.

[6] N. Qin, W. S. Wong, and A. L. Moigne. Three-dimensional contour bumps for transonic wing drag reduction. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 222(5):619–629, 2008.

[7] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[8] C. K. I. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. Technical Report NCRG/97/012, Dept. of Computer Science and Applied Mathematics, Aston University, Birmingham, 1997.