HAL open science

Study of some strategies for global optimization using Gaussian process models with application to aerodynamic design

C. Praveen, Régis Duvigneau

▶ To cite this version:

C. Praveen, Régis Duvigneau. Study of some strategies for global optimization using Gaussian process models with application to aerodynamic design. [Research Report] RR-6964, INRIA. 2009, pp.38. inria-00396926

HAL Id: inria-00396926

https://inria.hal.science/inria-00396926v1

Submitted on 19 Jun 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Study of some strategies for global optimization using Gaussian process models with application to aerodynamic design

Praveen. C — Regis Duvigneau

N° 6964

June 2009

Thème NUM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6964--FR+ENG

Study of some strategies for global optimization using Gaussian process models with application to aerodynamic design

Praveen. C* , Regis Duvigneau†

Thème NUM — Systèmes numériques
Projet OPALE

Rapport de recherche n° 6964 — June 2009 — 35 pages

Abstract: Aerodynamic shape optimization using CFD and global optimizers like PSO is a computationally expensive process. To make the method more efficient, metamodels must be incorporated into the optimization algorithm. Metamodels can be either local or global in their approximation ability. We use local models based on kriging in a pre-evaluation strategy to screen a small number of promising designs, which are then evaluated on the exact model. We also construct and use global metamodels as a complete replacement of the exact model, with various enrichment methods based on merit functions. The two methods are applied to inviscid, transonic drag reduction of a wing under lift and volume constraints, and their performance is compared. In all the tests, the use of metamodels reduces the computational expense. Global metamodels were found to give the maximum reduction in computational expense.

Key-words: Particle swarm optimization, metamodels, kriging, merit functions, aerodynamic shape optimization

* TIFR-CAM, Bangalore 560065, India
† Projet OPALE, INRIA, Sophia Antipolis, France

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Etude de quelques stratégies pour l'optimisation globale utilisant des modèles de processus Gaussiens avec application à la conception aérodynamique

Résumé : L'optimisation de forme aérodynamique basée sur des simulations CFD et des optimiseurs globaux tels PSO est une procédure coûteuse. Pour rendre ces méthodes plus efficaces, des métamodèles doivent être incorporés à l'algorithme d'optimisation. Ceux-ci peuvent être locaux ou globaux, pour leur domaine de validité. On utilise des modèles locaux basés sur le krigeage dans une stratégie de pré-évaluation pour sélectionner un nombre réduit de designs prometteurs, qui sont évalués par le modèle exact. On construit également des métamodèles globaux pour remplacer complètement le modèle exact, avec des méthodes d'enrichissement variées basées sur des fonctions de mérite. Les deux méthodes sont appliquées à la réduction de trainée d'une aile, en régime transsonique et non-visqueux, avec des contraintes de portance et de volume, et leurs performances sont comparées. On trouve que les métamodèles globaux permettent d'obtenir la réduction maximale en terme de coût de calcul.

Mots-clés : Optimisation par essaim de particules, Métamodèles, krigeage, fonctions de mérite, Conception optimale en aérodynamique

Contents

1 Introduction **4**

2 Kriging approximation **5**

2.1 Covariance function 7

2.2 Optimization of hyper-parameters 8

3 Merit functions **9**

3.1 Minimizing statistical lower bound 9

3.2 Maximizing probability of improvement 10

3.3 Maximizing expected improvement 10

4 Particle swarm optimization **11**

5 Optimization using local metamodels (IPE) **13**

6 Optimization using global metamodels (GMO) **16**

6.1 Generation of initial database 18

6.2 Construction of global metamodel 18

6.3 Minimization of global metamodel 18

6.4 Failure handling 19

6.5 Constraint handling 19

7 Numerical test case **19**

7.1 Transonic wing optimization 19

7.2 Parameterization using the Free-Form Deformation approach 20

7.3 Aerodynamic fitness evaluation using CFD 21

8 Results of optimization **23**

8.1 8 design variables 24

8.2 16 design variables 25

8.3 32 design variables 28

8.4 Discussion of results 30

9 Summary and conclusions **32**

1 Introduction

The ability to locate the globally optimal solution is an important consideration in many applications, especially in the preliminary design phase. Classical gradient based methods are not capable of finding the global optimum except in some special cases, e.g., when the function is strictly convex. A theorem by Torn and Zilinskas [24] states that for a general continuous function, an optimization method converges to the global optimum if the sequence of iterates it generates are dense in the design space. This means that the method has to search in all parts of the design space to find the global optimum, which makes sense intuitively. Modern optimization methods like genetic algorithms (GA) and particle swarm optimization (PSO) have the potential to locate the global optimum, though a rigorous proof is not available. These methods are however very expensive since they use a *population/swarm* of candidate solutions in every iteration, requiring the computation of the objective function several times in each iteration. The computational cost of these methods is further increased by the use of high fidelity analysis tools like Computational Fluid Dynamics (CFD) which are based on expensive numerical solution of partial differential equations. In such cases it is important to have optimization methods that can work with few function evaluations.

Consider the problem of minimizing a *cost* function $\mathcal{J}(x) := J(x, u(x))$

$$\min_{x \in D} \mathcal{J}(x), \quad D \subset \mathbb{R}^d \tag{1}$$

where the state u is obtained by the solution of some partial differential equations (expensive model), implicitly depending on x, and subject to some constraints

$$\mathcal{C}_i(x, u) \geq 0, \quad i = 1, ..., N_c \tag{2}$$

An obvious way to reduce the computational cost is to replace the costly analysis tool with a cheaper *surrogate* or *meta* model. The first step consists in building a database of design variables and corresponding function values $(x_m, \mathcal{J}_m, \mathcal{C}_{im})$, $m = 1, ..., M$, where the set of design variables $(x_m)_m$ is possibly obtained through a design of experiments procedure. The database can then be used to construct a metamodel $\hat{\mathcal{J}}$, $\hat{\mathcal{C}}_i$ of $\mathcal{J}$, $\mathcal{C}_i$. Metamodels can be distinguished based on whether they provide a local or global approximation of the cost function. A global model is constructed using all the available data and provides an approximation in the entire design space. A local model however is constructed around a point of interest $x_o \in D$ where the function value is to be approximated, and uses a small subset of the data in the neighbourhood of that point.

When global models can be constructed sufficiently accurately, they can help to locate the global optimum with fewer function evaluations. Instead of taking small steps as in classical

gradient based methods, using a global model can help to quickly reach more promising regions of the design space due to the ability of the metamodel to provide global trends in the cost function. However global models can be difficult and costly to construct when the dimension of the design space is large and the cost function is complex. Local models have the advantage of being cheaper to construct since they typically use a small dataset. In this case, the burden of locating the global optimum is entirely on the optimization algorithm.

In this work, we construct local and global kriging models and use them to solve the optimization problem (1)-(2) with PSO.

- Local models are used to approximate the cost and/or constraint functions and a small subset of the swarm is chosen for evaluation on the exact model. This leads to a reduction in the number of costly function evaluations and makes the algorithm more efficient. The strong coupling between the metamodel, optimization algorithm and the exact model leads to a robust algorithm with substantial reduction in computational cost.
- Global models are constructed using an initial database of design variables and new design points are chosen by minimizing certain merit functions. These points are added to the database and the metamodel is updated leading to a progressive improvement of the metamodel in interesting regions of the design space.

The report is organized as follows. We begin with a description of the kriging approximation in section (2) and explain our specific implementation of the method for determining the parameters in the model. Next in section (3) we explain three merit functions and their significance towards exploration and exploitation of the design space. This is followed by a description of the PSO algorithm, section (4), which is used to solve the minimization problem and also for training metamodels and minimizing the merit functions. Section (5) describes our PSO algorithm using inexact pre-evaluations that require local metamodels. Section (6) describes the global metamodel-based optimization algorithm. Finally, section (7) describes the test case of aerodynamic shape optimization including CFD algorithm and shape parameterization, while section (8) presents the results obtained from different optimization methods.

2 Kriging approximation

Gaussian process models [16, 20, 27] (also called kriging) treat the response of some experiment as if it were a realization of a stochastic process. This stochastic process approach has been developed in three different disciplines: geology, global optimization and statistics. In geology, the approach is known as kriging and was developed to predict the underground concentration of a valuable mineral using the data from core samples taken at different locations. In global optimization, the use of stochastic processes is called 'Bayesian global

optimization' and dates back to the work of Harold Kushner (1964). The mathematical theory of kriging was developed by the French mathematician Georges Matheron based on the work of mining engineer Danie G. Krige, after whom it is known as kriging.

In physical experiments, there are independent random errors which make the assumption of a stochastic process plausible. But in a computer experiment, where the response is the output of a computer code which is deterministic, its plausibility is less clear. The output of a computer code also has some high-frequency, low-amplitude errors due to convergence and stopping criteria, limiters, etc., but the general trends are different[1].

In the following, we adopt the Bayesian viewpoint of Gaussian processes [16, 27]. The problem is to find an approximation $\hat{f}(x)$ 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

$$p(F_N|X_N) = \frac{\exp\left(-\frac{1}{2}F_N^\top C_N^{-1} F_N\right)}{\sqrt{(2\pi)^N \det(C_N)}} \tag{3}$$

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

$$p(F_{N+1}|X_{N+1}) = \frac{\exp\left(-\frac{1}{2}F_{N+1}^\top C_{N+1}^{-1} F_{N+1}\right)}{\sqrt{(2\pi)^{N+1} \det(C_{N+1})}} \tag{4}$$

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

$$p(f_{N+1}|X_{N+1}, F_N) = \frac{p(F_{N+1}|X_{N+1})}{p(F_N|X_N)} \tag{5}$$

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

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

[1] In this regard, Torczon et al. state [23]: *we regard the supposition of an underlying stochastic process as nothing more than a convenient fiction. The value of this fiction lies in its power to suggest plausible ways of constructing useful approximations and we will try to avoid invoking it excessively. When we do invoke it, it should be appreciated that optimality criteria such as BLUP and MLE are defined with respect to the fictional stochastic process and should not be invested with more importance than the practical utility of the approximations in which they result.*

where

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

and

$$\kappa = c(x_{N+1}, x_{N+1})$$

This gives

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

where

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

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

$$\hat{f}_{N+1} = k^\top C_N^{-1} F_N, \qquad \sigma_{f_{N+1}}^2 = \kappa - k^\top C_N^{-1} k \tag{6}$$

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 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.

Note: The kriging model reproduces the input data exactly as long as the covariance matrix has full rank. To see this, let us predict the function value at $x_{N+1} = x_1$. Note that in this case, the vector k is identical to the first column of C_N. Thus, since C_N is symmetric, we have

$$k^\top C_N^{-1} = [1, 0, \ldots, 0]$$

which gives $\hat{f}_{N+1} = k^\top C_N^{-1} F_N = f_1$.

2.1 Covariance function

The covariance function must reflect the characteristics of the output of the computer code. For a smooth response, a covariance function with sufficient number of derivatives is preferable, whereas an irregular response might require a covariance function with no derivatives. In the absence of any knowledge regarding the unknown function, the most commonly used correlation function is an exponential; in this work we take the correlation function of the form

$$c(x, y) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{r_i^2}\right] + \theta_2$$

where $\Theta = (\theta_1, \theta_2, r_1, r_2, \ldots, r_d)$ are some parameters to be determined. The first term is a distance-dependent correlation between the function values at two data points; if their distance is small compared to the length scales r_i, the exponential term is close to one while it decays exponentially as their distance increases. The parameter θ_1 scales this correlation. In the second term, θ_2 gives an offset of the function values from zero. It is also common to use a single length scale r which reduces the number of parameters to three irrespective of the dimension d of the independent variables. This choice may be sufficient for constructing local kriging models.

2.2 Optimization of hyper-parameters

It now remains to find the parameters Θ in the correlation function. These parameters are determined by maximizing the joint probability density $p(F_N|X_N)$. This is equivalent to minimizing the log-likelihood function given by

$$\mathfrak{L} = F_N^\top C_N^{-1} F_N + \log\det(C_N)$$

This function is known to be multi-modal; hence robust techniques like Genetic Algorithms [4] or Particle Swarm Optimization [25] might be preferable. Many researchers have combined such techniques with a final gradient search to locate the minimum more accurately [2]. In this work we have used PSO technique but without any gradient search.

There are many practical issues that must be taken care of so that all the computations are stable; we follow [2] in this respect. The hyper-parameters must be non-negative; hence it is better to work with the logarithm of the parameters so that they always remain positive. The optimization using PSO is thus applied to the logarithm of the hyper-parameters. The correlation matrix can be ill-conditioned in which case the computation of its inverse will not be accurate. If the condition number is above a specified tolerance, then the log-likelihood is taken to be a large positive value. If the condition number is within the specified tolerance, an LU decomposition of C_N is first performed; then $C_N^{-1} F_N$ and $C_N^{-1} k$ are computed using the LU decomposition. The logarithm of the determinant of C_N is also computed using the LU decomposition

$$\log\det(C_N) = \sum_{n=1}^{N} \log L_{nn} + \sum_{n=1}^{N} \log U_{nn}$$

This avoids the under-flow problem associated with multiplying a large number of small numbers which would be the case if the matrix C_N is badly scaled.

Again following [2] we scale the coordinates and function values so that they lie in the interval $[0, 1]$. For each particle, the parameters in the covariance function are initialized randomly in the intervals as given below:

$$\begin{aligned} \theta_1 &\in [10^{-3}, 1] \\ \theta_2 &\in [10^{-3}, 1] \\ r_i &\in [10^{-2}, 10], \quad i = 1, \ldots, d \end{aligned}$$

In the PSO method, if a particle goes outside this range then its previous position is restored and it's velocity is set to zero. If we do not restrict the range in this manner, then we found that the PSO can converge to a solution which does not give an accurate interpolation.

3 Merit functions

The use of metamodels for optimization has to be an iterative process since it is not possible in general to construct a very accurate model in one shot. The model has to be updated using new data in each iteration until some convergence criterion is satisfied. Ideally, for locating the global optimum, one may construct a metamodel which is accurate in all regions of the design space. However this can be very expensive for problems with large number of design variables. Actually, for the optimization problem, it is enough if the model is accurate in the interesting regions where optimum solutions are located. The computational cost and convergence towards the global optimum depends on the way we update the metamodel which in turn depends on the selection of new evaluation points.

The most obvious way to select new evaluation point is to use the optimum solution of the metamodel. However this does not take account of the inaccuracy in the metamodel and can converge to a local optimum or even to some point which is not even a local optimum. In order to improve the metamodel, it is necessary to have an estimate of the error in the model. Kriging or gaussian process models provide an intrinsic estimate of the error or variance in the values predicted by the model. This makes it possible to select new evaluation points so that the model accuracy can be improved. The new search point is chosen as the minimizer (or maximizer) of a *merit function*; there are many different merit functions that make use of the variance [12] and are discussed in the following sections.

3.1 Minimizing statistical lower bound

Torczon et al. [5] and Cox and John [3] suggested the use of the lower confidence bound of the prediction as a merit function. If the model gives an estimate of the standard deviation in the approximation $\hat{s}(x)$, then we can construct a statistical lower bound of the cost function as

$$f_M(x) = \hat{\mathcal{J}}(x) - \kappa\hat{s}(x) \tag{7}$$

The merit function is minimized and the new evaluation point is the minimizer of the merit function. Several merit functions with different values of κ are minimized which gives a set of new evaluations points. A small value of κ leads to searching around the current minimum of the metamodel. A large value of κ may be expected to give a good estimate of the lower bound of the cost function and leads to better exploration of the search space where the data is less certain or non-existent. According to [2], in practice using four different values of $\kappa = 0, 1, 2, 4$ is sufficient. However since the standard deviation is only an approximation, it can be very innacurate especially with sparse data, and may fail to generate an evaluation

point in the region of the global optimum [12]. Also, the iterates from minimizing this merit function will not be dense due to deletion of regions of the search space where the lower bound is above the current best function value, which can happen if the variance is estimated very inaccurately.

3.2 Maximizing probability of improvement

The idea of this merit function, first introduced by Harold Kushner [14], is to choose the next evaluation point so that the probability of finding a solution better than the current best is maximized. Let $\mathcal{J}_{\min}$ be the cost function of the best solution found so far. We set a target cost function $T < \mathcal{J}_{\min}$ and look for a point x for which the probability that $\hat{\mathcal{J}}(x) \leq T$ is maximum. We assume that the function value is the realization of a random variable with mean $\hat{\mathcal{J}}(x)$ and standard error $\hat{s}(x)$. Then the probability of finding a value T is

$$\mathrm{PoI}(x) = \Phi\left(\frac{T - \hat{\mathcal{J}}(x)}{\hat{s}(x)}\right) \tag{8}$$

where $\Phi(y) = \frac{1}{2}[1+\mathrm{erf}(y/\sqrt{2})]$ is the cumulative distribution function of the standard normal distribution and the merit function is taken to be the negative of PoI. A disadvantage of this approach is that the user has to specify the target value T. A value of T close to $\mathcal{J}_{\min}$ leads to highly local search around the current best solution while a value of $T \ll \mathcal{J}_{\min}$ leads to a more global search. In practice one can use several values of the target T corresponding to low, medium and high desired improvement. This will enable both global search during the initial stages and a fine local search at later iterations. For the one-dimensional algorithm, Gutmann [11] proves under certain mild assumptions, that the iterates from minimizing this merit function are dense.

3.3 Maximizing expected improvement

This merit function is based on locating the point x at which the maximum reduction in cost function can be expected. We again assume that the unknown function is a realization of a random variable with mean and variance given by $\hat{\mathcal{J}}(x)$ and $\hat{s}(x)$ respectively. If $\mathcal{J}_{\min}$ is the cost function of the current best solution, then the probability of reducing the cost function by an amount I is given by the normal density function

$$\frac{1}{\sqrt{2\pi}\hat{s}(x)} \exp\left[-\frac{(\mathcal{J}_{\min} - I - \hat{\mathcal{J}}(x))^2}{2\hat{s}(x)}\right]$$

The expected improvement is the expected value of I under the above distribution and is given by

$$\mathrm{EI}(x) = \int_{I=0}^{I=\infty} I \left\{\frac{1}{\sqrt{2\pi}\hat{s}(x)} \exp\left[-\frac{(\mathcal{J}_{\min} - I - \hat{\mathcal{J}}(x))^2}{2\hat{s}(x)}\right]\right\} \mathrm{d}I$$

which can be shown to be equal to

$$\mathrm{EI}(x) = \hat{s}(x)[u\Phi(u) + \phi(u)], \quad u(x) = \frac{\mathcal{J}_{\min} - \hat{\mathcal{J}}(x)}{\hat{s}(x)} \tag{9}$$

and where $\phi(y) = \frac{1}{\sqrt{2\pi}} \exp(-y^2/2)$ is the standard normal distribution. The merit function is defined as negative of EI. An advantage of this merit function is that it avoids the need to specify any parameters by the user (like the target T of the previous merit function). Under some assumptions, Locateli [15] has shown that the iterates from minimizing this merit function are dense.

4 Particle swarm optimization

PSO is modeled on the behaviour of a swarm of animals when they hunt for food or avoid predators [18]. In nature a swarm of animals is found to exhibit very complex behaviour and capable of solving difficult problems like finding the shortest distance to a food source. However the rules that govern the behaviour of each animal are thought to be simple. Animals are known to communicate the information they have discovered to their neighbours and then act upon that individually. The individuals cooperate through self-organization but without any central control. The interaction of a large number of animals acting independently according to some simple rules produces highly organized structures and behaviours.

In PSO, a swarm of particles wanders around in the design space according to some specified velocity. The position of each particle corresponds to one set of design variables and it has an associated value of the cost function. Each particle remembers the best position it has discovered in its entire lifetime (local memory) and also knows the best position discovered by its neighbours and the whole swarm (global memory). The velocity of each particle is such as to pull it towards its own memory and that of the swarm. While there are many variants of the PSO algorithm, the one we use is described below and complete details are available in [7]. The algorithm is given for a function minimization problem

$$\min_{x_l \le x \le x_u} \mathcal{J}(x)$$

Constraints are incorporated into the cost function using the penalty function approach.

Algorithm: *Particle swarm optimization*

1. Set $n = 0$
2. Randomly initialize the position of the particles and their velocities $\{x_k^n, v_k^n\}, k = 1, \ldots, K$.
3. Compute cost function associated with the particle positions $\mathcal{J}(x_k^n), k = 1, \ldots, K$

4. Update the local and global memory

$$x^n_{*,k} = \underset{0\le s\le n}{\operatorname{argmin}}\, \mathcal{J}(x^s_k), \quad x^n_* = \underset{0\le s\le n, 1\le k\le K}{\operatorname{argmin}}\, \mathcal{J}(x^s_k) \tag{10}$$

5. Update the particle velocities

$$v^{n+1}_k = \omega^n v^n_k + c_1 r^n_{1,k}(x^n_{*,k} - x^n_k) + c_2 r^n_{2,k}(x^n_* - x^n_k) \tag{11}$$

6. Apply craziness operator to the velocities
7. Update the position of the particles

$$x^{n+1}_k = x^n_k + v^{n+1}_k \tag{12}$$

8. Limit new particle positions to lie within $[x_l, x_u]$ using reflection at the boundaries
9. If $n < N_{\max}$, then $n = n + 1$ and go to step (iii), else STOP.

Apart from the above basic algorithm, we use several additional strategies to enhance the efficiency of PSO. The inertia parameter ω is decreased during the iterations as proposed by Fourie and Groenwold [10]. A starting ω^o is chosen with a large value in order to promote an exploratory search. Then its value is decreased by a factor α if no improved solution is found within h consecutive time steps:

$$\text{If } \mathcal{J}(x^n_*) = \mathcal{J}(x^{n-h}_*) \text{ then } \omega^n = \alpha\omega^{n-1}$$

with $\alpha \in (0, 1)$. Therefore, if the exploratory search fails, then the convergence towards the best locations ever found is promoted. A craziness operator is implemented on the velocity [10] which is inspired by the mutation operator in GAs. A probability of craziness $p_c \in [0, 1]$ is chosen; then, at each time step and for each particle, the velocity direction is randomly modified with the probability p_c, but the velocity modulus is kept constant. Therefore, large random perturbations occur at the beginning of the optimization procedure, which promote random global search, whereas small random perturbations are performed when the swarm is close to the solution, which promote random local search. This approach is inspired from the so-called non-uniform mutation operator in GAs [17]. Finally, an upper limit on velocity as recommended by Shi and Eberhart [22] in order to improve the stability and convergence rate of PSO is also used.

In the original algorithm proposed by Kennedy and Eberhart [13] the random numbers r_1, r_2 are scalars, i.e., one random number is used for all the velocity components of a particle. In practical implementation, it is found that researchers have used both a scalar and vector version of random numbers. In the vector version, a different random number is used for each component of the velocity vector. This is equivalent to using random diagonal matrices for r_1 and r_2. Wilke [26] has investigated the difference in performance of PSO between these versions and concludes that the scalar version is susceptible to getting trapped in a line search while the vector version does not have this problem. The vector version is also preferred for use with metamodels since it has space-filling characteristics.

5 Optimization using local metamodels (IPE)

PSO is a rank-based algorithm; the actual magnitude of cost function of each particle is not important but only their relative ordering matters. An examination of the PSO algorithm shows that the main driving factors are the local and global memories. Most of the cost functions are discarded except when it improves the local memory of some particle. This loss of information can be avoided by storing the function values in a database which can be used to build approximations of the cost function. For each particle, a local dataset is selected from the database comprising of nearest points and a local metamodel is built; the objective and constraint functions are approximated using the local metamodel (in-exact pre-evaluation). Using the approximate information, a small subset of the particles is selected (pre-screening) which are most promising in the sense that they may lead to improvement of the best solution discovered till that time. This small set of particles is evaluated on the exact model and the resulting function values are stored in the database.

When updating the local and global memories, the cost functions are of mixed type; some particles have cost functions evaluated on the metamodel and a few are evaluated using the exact model. If the memories are updated using cost functions evaluated on the metamodel, then there is the possibility that the memory may improve due to error in the cost function. This erroneous memory may cause PSO to converge to it or may lead to wasteful search. Hence the memories are updated using only the exactly evaluated cost functions. We have proposed a metamodel-assisted PSO with inexact pre-evaluation [19] as follows; the first N_e iterations of PSO are performed with exact function evaluations which are stored in a database. In the subsequent iterations the metamodel is used to pre-screen the particles and only a small percentage of particles is evaluated on the exact function.

Algorithm: *Particle swarm optimization with IPE*

1. Set $n = 0$
2. Randomly initialize the position of the particles and their velocities $\{x_k^n, v_k^n\}, k = 1, \ldots, K$.
3. If $n \leq N_e$ compute cost function associated with the particle positions $\mathcal{J}(x_k^n), k = 1, \ldots, K$ using the exact model, else compute the cost function using metamodel $\hat{\mathcal{J}}(x_k^n), k = 1, \ldots, K$.
4. If $n > N_e$, then select a subset of particles S^n based on a pre-screening criterion and evaluate the exact cost function for these particles. Store the exact cost functions into the database.
5. Update the local and global memory *using only the exactly evaluated cost functions*

$$x_{*,k}^n = \underset{0 \leq s \leq n}{\operatorname{argmin}} \mathcal{J}(x_k^s), \quad x_*^n = \underset{0 \leq s \leq n, 1 \leq k \leq K}{\operatorname{argmin}} \mathcal{J}(x_k^s) \tag{13}$$

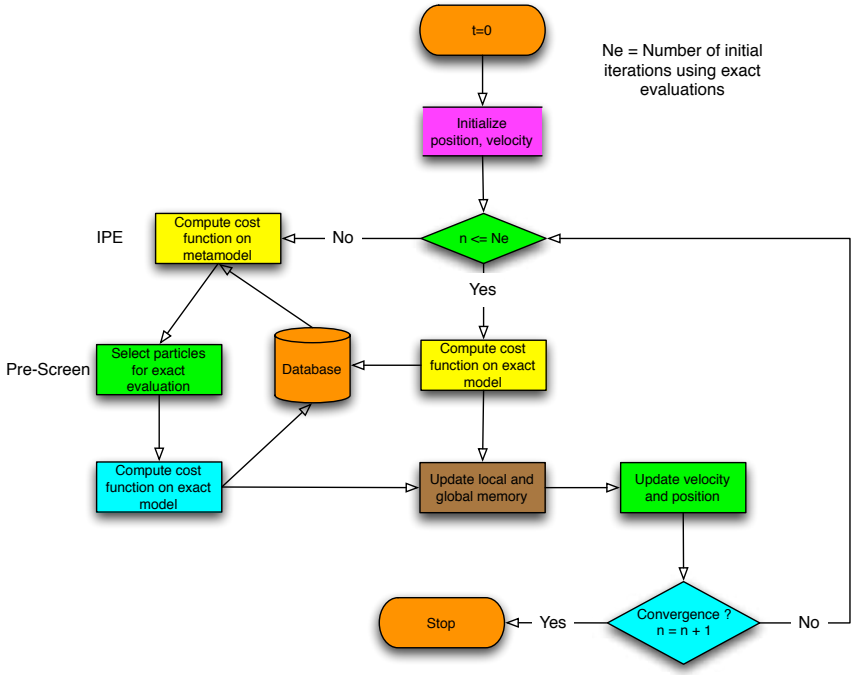


Figure 1: Schematic of the IPE algorithm

6. Store exactly evaluated function values into a database
7. Update the particle velocities

$$v_k^{n+1} = \omega^n v_k^n + c_1 r_{1,k}^n (x_{*,k}^n - x_k^n) + c_2 r_{2,k}^n (x_*^n - x_k^n) \tag{14}$$

8. Apply craziness operator to the velocities
9. Update the position of the particles

$$x_k^{n+1} = x_k^n + v_k^{n+1} \tag{15}$$

10. Limit new particle positions to lie within $[x_l, x_u]$ using reflection at the boundaries
11. If $n < N_{\max}$, then $n = n + 1$ and go to step (iii), else STOP.

This algorithm is illustrated in figure (1).

The important aspect of metamodel assisted optimization is the criterion used to select the set S of particles whose function value will be exactly evaluated. The selection criterion should pick the most promising designs, i.e., those which are likely to lead to further reduction in the cost function. It should also promote exploration of those regions of search space where data is sparse. Giannakoglou [8] discusses several pre-screening criteria based on the estimated fitness function and standard deviation of the estimation whenever available, as in the case of gaussian random process models. The rationale behind the pre-screening criteria are similar to the merit functions. Several pre-screening criteria are discussed below.

1. *Mean value criterion*: This criterion selects all particles whose estimated function value is less than the current minimum function value i.e., all x_k for which $\hat{\mathcal{J}}(x_k) < \mathcal{J}_{\min}$.

2. *Lower bound criterion* (LB): This criterion is similar to the mean value criterion but it also makes use of the standard deviation which indicates error in the metamodel prediction and promotes exploration of unexplored regions of the search space. All particles x_k for which $\hat{\mathcal{J}}(x_k) - \kappa\hat{s}(x_k) < \mathcal{J}_{\min}$ are selected; here $\kappa = 3$ can be used.

3. *Probability of Improvement criterion* (PI): This criterion selects all particles which have a high probability of the cost function being reduced below the current minimum value. The target T is taken to be 10% less than the current minimum function value $\mathcal{J}_{\min}$. There can be two variants of this criterion:

   - a specified percentage of particles with highest probability of improvement.
   - all particles x_k for which $\mathrm{PoI}(x_k) > P_m$, P_m being specified by the user.

4. *Expected Improvement criterion* (EI): This criterion selects all particles for which the expected improvement is high.

   - a specified percentage of particles which have the highest expected improvement.
   - all particles x_k for which $\mathrm{EI}(x_k) > I_m$, I_m being specified by the user. The value of I_m must be adjusted as the optimization progresses.

5. *Local improvement criterion* (A): This criterion was proposed specifically for use with PSO [19]. It selects all particles whose cost function is less than their local memory, i.e., all particles x_k for which $\hat{\mathcal{J}}(x_k) < \mathcal{J}(x_{*,k})$. This is similar to the mean value criterion but the improvement is checked over the local memory of each particle rather than the global memory. It promotes greater exploration of the design space since it searches the regions around the local memory of each particle. This criterion can be modified to take account of the standard deviation information similar to the lower bound criterion: select all particles x_k for which $\hat{\mathcal{J}}(x_k) - \kappa\hat{s}(x_k) < \mathcal{J}(x_{*,k})$. The number of particles to be selected is automatically determined by the algorithm and hence it is said to be *adaptive* (A).

In the present work, we test the IPE-LB, IPE-PI, IPE-EI and IPE-A algorithms on the aerodynamic shape optimization problem. With the LB, PI and EI pre-screening methods, we choose 10% of the particles for evaluation on the exact model (CFD).

6 Optimization using global metamodels (GMO)

A global metamodel provides an approximation to the cost and constraint functions in the entire design space. This allows us to completely replace the costly analysis tools with these cheaper metamodels. The optimization algorithm is applied to the metamodel to predict a better solution. However this cannot be a one-shot process since the metamodel is an approximation, usually very coarse, and the optimum solution predicted by minimizing it may not really be the optimum and/or may not satisfy the constraints. The metamodel must be updated by adding new data points and the optimization applied to the new model. This process is continued until some convergence criterion is satisfied or the computational resources are exhausted. The selection of the new evaluation points is the most crucial aspect of this method. New points must be added in those regions of the design space where there is more likelihood of the existence of an optimum. This is achieved by selecting the new evaluation points as the minimizers of the merit function(s), which are described in section (3). The basic algorithm using global metamodels and merit functions is outlined below.

Algorithm: *Global metamodel-based optimization*

1. Generate an initial database
2. Construct a global metamodel
3. Minimize merit functions using the metamodel
4. Evaluate minimizers on exact model
5. Store the new function values in the database
6. Construct a metamodel using updated database.
7. If not converged, go to step 3.

The above algorithm is illustrated schematically in fig. (2). The minimization of the metamodel must be performed using a global optimization method; in this work we use PSO for this purpose. This step is not expensive since the metamodel can be evaluated cheaply, even using a large swarm size in PSO and performing a large number of iterations.

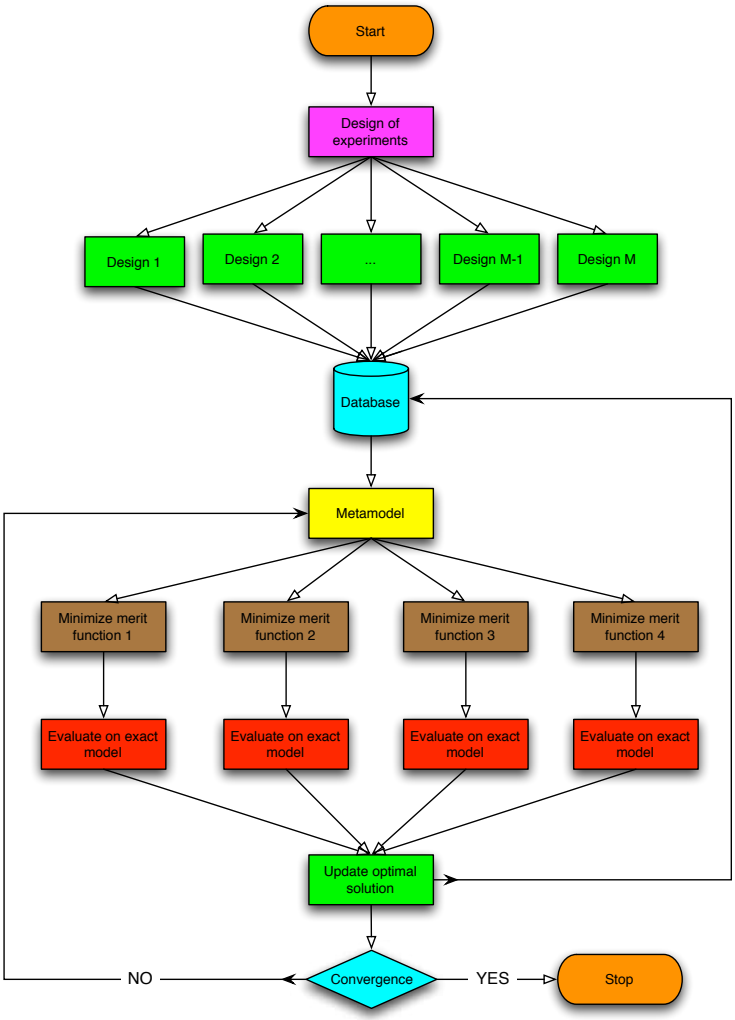


Figure 2: Schematic of global metamodel-based optimization

6.1 Generation of initial database

The design space is initially sampled to generate a set of points that will be used to build the initial metamodel. It is necessary to sample all parts of the design space since a priori we do not know where the optimum solutions are located. However when the dimension of the space is large, it is not possible to sample it in a dense way as it requires a large number of points; this is known as the *curse of dimensionality*. One can only sample it in a very sparse way but still satisfying some space-filling criteria. Here we use the Latin Hypercube sampling technique to generate an initial sample that leads to variations in all design variables between their minimum and maximum range. In practice, due to the high computational expense of simulation tools, it may be possible to only generate an initial database of the order of a few hundred solutions or maybe even just a few tens of solutions, depending on available computational resources and time. The evaluation of all the design variables can be performed in parallel; if there are M initial designs to generate the database, one can use M processors to compute all the designs simultaneously.

6.2 Construction of global metamodel

In this work we use kriging approximations to construct global metamodels of all the functions that are required for optimization. This includes the objective function like drag coefficient and constraint functions like lift coefficient and volume. Separate metamodels are constructed for each of these functions and the resulting models are used to evaluate the cost function involving penalty terms and merit function. The metamodel must be constructed or re-trained after every iteration since new points are added to the database in every iteration. Training a metamodel involves finding the coefficients in the kriging model as described in section (2) using PSO. Whenever the metamodel is being re-trained, the previously determined set of kriging parameters are also used in the PSO to initialize the particle positions, which can accelerate its convergence.

6.3 Minimization of global metamodel

The global metamodel is used to minimize certain merit functions which are described in section (3). Our objective is to find the global minimizers and hence we should use algorithms that are capable of locating global optima. In the present work, we use PSO as described in section (4) to minimize the merit functions. Since the metamodel is cheaper to evaluate, we can use a larger number of particles in the swarm and also perform more number of iterations. The global minimizers of the merit functions are then evaluated on the exact model, and resulting function values are stored in the database. During this step, it is important to ensure that new data points that are added are not too close to any of the existing points in the database.

6.4 Failure handling

The CFD code can sometimes fail to converge due to various reasons. For compressible flows, violation of positivity of some variables like density and pressure can occur. Another difficulty is that the grid deformation algorithm can fail due to creation of negative volumes. When performing global optimization, all regions of the design space may be explored which can lead to very large grid deformations. In these situations, we assign a large cost function value to the corresponding design variable. This enables us to avoid those regions of the design space which leads to failure. However it is preferable to have a robust grid deformation method since otherwise it can exclude important regions of the design space.

6.5 Constraint handling

A straight-forward way to incorporate equality and inequality constraints is through the penalty function approach. When the constraint is violated, the penalty term adds a large numerical value to the cost function. In the context of aerodynamic shape optimization, there are aerodynamic constraints like those on lift and geometric constraints on volume and thickness. Modelling the cost function including the penalty terms is not a good approach since the function will have very large numerical values when constraint is violated and O(1) values when it is not violated. Hence we prefer to model the objective and constraint functions using separate metamodels and then evaluate the cost function.

7 Numerical test case

This section describes the aerodynamic shape optimization problem, the parameterization of the shape using free form deformation (FFD) and the numerical algorithm for solving the flow equations.

7.1 Transonic wing optimization

The test-case considered here corresponds to the optimization of the shape of the wing of a business aircraft (courtesy of Piaggio Aero Ind.), for a transonic regime. The test-case is described in depth in [1]. The overall wing shape can be seen in figure (3). The free-stream Mach number is $M_\infty = 0.83$ and the incidence $\alpha = 2°$. Initially, the wing section is supposed to correspond to the NACA 0012 airfoil.

The goal of the optimization is to reduce the drag coefficient C_d subject to the constraint that the lift coefficient C_l should not be less than C_{l_0}. The constraint is taken into account using a penalization approach. Then, the resulting cost function is :

$$\mathcal{J} = \frac{C_d}{C_{d0}} + 10^4 \max(0, 1 - \frac{C_l}{C_{l0}}) + 10^4 \max(0, \frac{V_0 - V}{V_0}) \tag{16}$$

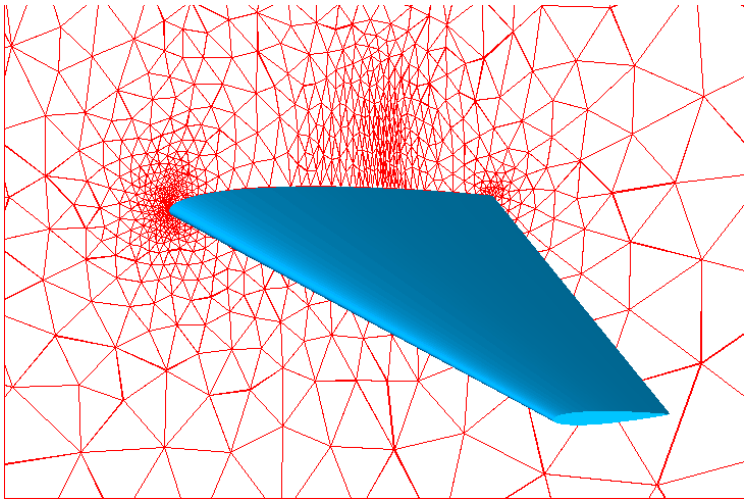


Figure 3: Initial wing shape (blue) and mesh in the symmetry plane (red).

C_{d0} and C_{l0} are respectively the drag and lift coefficients corresponding to the initial shape (NACA 0012 section) and V_0 is the wing volume. For the CFD computations, an unstructured mesh, composed of 31124 nodes and 173 445 tetrahedral elements, is generated around the wing, including a refined area in the vicinity of the shock (figure (3)).

7.2 Parameterization using the Free-Form Deformation approach

The FFD technique originates from the Computer Graphics field [21]. It allows the deformation of an object in a 2D or 3D space, regardless of the representation of this object. Instead of manipulating the surface of the object directly, by using classical B-Splines or Bézier parameterization of the surface, the FFD techniques defines a deformation field over the space embedded in a lattice which is built around the object. By transforming the space coordinates inside the lattice, the FFD technique deforms the object, regardless of its geometrical description. An added advantage is that the computational grid used for CFD can also be deformed simulataneously to conform to the new shape of the object; this is also independent of the type of grid that is used, making it a very versatile method.

More precisely, consider a three-dimensional hexahedral lattice embedding the object to be deformed. Figure (4) shows an example of such a lattice built around a realistic wing. A local coordinate system (ξ, η, ζ) is defined in the lattice, with $(\xi, \eta, \zeta) \in [0, 1] \times [0, 1] \times [0, 1]$. During the deformation, the displacement Δq of each point q inside the lattice is here defined

by a third-order Bézier tensor product:

$$\Delta q = \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} B_i^{n_i}(\xi_q) B_j^{n_j}(\eta_q) B_k^{n_k}(\zeta_q) \Delta P_{ijk}. \tag{17}$$

$B_i^{n_i}$, $B_j^{n_j}$ and $B_k^{n_k}$ are the Bernstein polynomials of order n_i, n_j and n_k (see for instance [9]):

$$B_p^n(t) = C_n^p \, t^p \, (1-t)^{n-p}. \tag{18}$$

$(\Delta P_{ijk})_{0 \le i \le n_i, 0 \le j \le n_j, 0 \le k \le n_k}$ are weighting coefficients, or control points displacements, which are used to monitor the deformation and are considered as design variables during the shape optimization procedure.

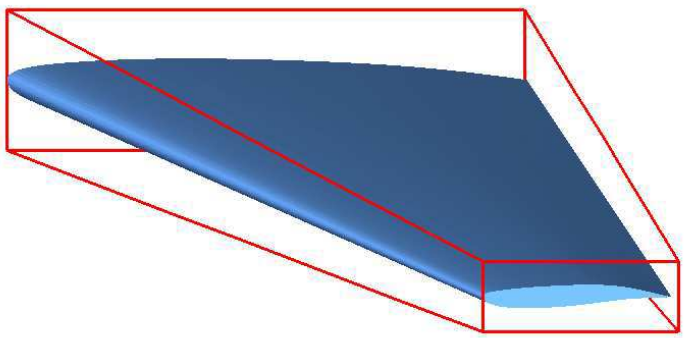


Figure 4: Example of FFD lattice (red) around a wing.

For the aerodynamic optimization, the FFD lattice is built around the wing with ξ, η and ζ in the chordwise, spanwise and thickness directions respectively. The lattice is chosen in order to fit the planform of the wing (see figure 4). Then, the leading and trailing edges are kept fixed during the optimization by freezing the control points that correspond to $i = 0$ and $i = n_i$. Moreover, all control points are only moved vertically. Hence for a parameterization of $n_i \times n_j \times n_k$, we obtain $(n_i - 1) \times (n_j + 1) \times (n_k + 1)$ design variables. In all the test cases in this work, we use $n_j = n_k = 1$; this leads to a linear interpolation of the root and tip airfoil sections over the span.

7.3 Aerodynamic fitness evaluation using CFD

Modeling This study is restricted to three-dimensional inviscid compressible flows governed by the Euler equations. Then, the state equations can be written in the conservative form :

$$\frac{\partial W}{\partial t} + \frac{\partial F_1(W)}{\partial x} + \frac{\partial F_2(W)}{\partial y} + \frac{\partial F_3(W)}{\partial z} = 0, \tag{19}$$

where W are the conservative flow variables $(\rho, \rho u, \rho v, \rho w, E)$, with ρ the density, $\overrightarrow{U} = (u, v, w)$ the velocity vector and E the total energy per unit of volume. $\overrightarrow{F} = (F_1(W), F_2(W), F_3(W))$ is the vector of the convective fluxes, whose components are given by :

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{pmatrix} \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{pmatrix} \quad F_3(W) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{pmatrix}. \tag{20}$$

The pressure p is obtained from the perfect gas state equation :

$$p = (\gamma - 1)(E - \frac{1}{2}\rho \|\overrightarrow{U}\|^2), \tag{21}$$

where $\gamma = 1.4$ is the ratio of the specific heat coefficients.

Spatial discretization Provided that the flow domain Ω is discretized by a tetrahedrization $\mathcal{T}_h$, a discretization of equation (19) at the mesh node s_i is obtained by integrating (19) over the volume C_i, that is built around the node s_i by joining barycenters of the tetrahedra and triangles containing s_i and midpoints of the edges adjacent to s_i :

$$Vol_i \frac{\partial W_i}{\partial t} + \sum_{j \in N(i)} \Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = 0, \tag{22}$$

where W_i represents the cell averaged state and Vol_i the volume of the cell C_i. $N(i)$ is the set of the neighboring nodes. $\Phi(W_i, W_j, \overrightarrow{\sigma}_{ij})$ is an approximation of the integral of the fluxes (20) over the boundary ∂C_{ij} between C_i and C_j, which depends on W_i, W_j and $\overrightarrow{\sigma}_{ij}$ the integral of a unit normal vector over ∂C_{ij}. These numerical fluxes are evaluated using upwinding, according to the approximate Riemann solver of Roe :

$$\Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = \frac{\overrightarrow{F}(W_i) + \overrightarrow{F}(W_j)}{2} \cdot \overrightarrow{\sigma}_{ij} - |A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})| \frac{W_j - W_i}{2}. \tag{23}$$

A_R is the jacobian matrix of the fluxes for the Roe average state and verifies:

$$A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})(W_j - W_i) = (\overrightarrow{F}(W_j) - \overrightarrow{F}(W_i)) \cdot \overrightarrow{\sigma}_{ij}. \tag{24}$$

A high order scheme is obtained by interpolating linearly the physical variables from s_i to the midpoint of $[s_i s_j]$, before equation (22) is employed to evaluate the fluxes. Nodal gradients are obtained from a weighting average of the P1 Galerkin gradients computed on each tetrahedron containing s_i. In order to avoid spurious oscillations of the solution in the vicinity of the shock, a slope limitation procedure using the Van-Albada limiter is introduced. The resulting discretization scheme exhibits a third order accuracy in the regions where the solution is regular.

Time integration A first order implicit backward scheme is employed for the time integration of (22), which yields :

$$\frac{Vol_i}{\Delta t}\delta W_i + \sum_{j\in N(i)} \Phi(W_i^{n+1}, W_j^{n+1}, \overrightarrow{\sigma}_{ij}) = 0, \tag{25}$$

with $\delta W_i = W_i^{n+1} - W_i^n$. Then, the linearization of the numerical fluxes provides the following integration scheme :

$$\left(\frac{Vol_i}{\Delta t} + J_i^n\right)\delta W_i = -\sum_{j\in N(i)} \Phi(W_i^n, W_j^n, \overrightarrow{\sigma}_{ij}). \tag{26}$$

Here, J_i^n is the jacobian matrix of the first order numerical fluxes, whereas the right hand side of (26) is evaluated using high order approximations. The resulting integration scheme provides a high order solution of the problem. More details can be found in [6].

8 Results of optimization

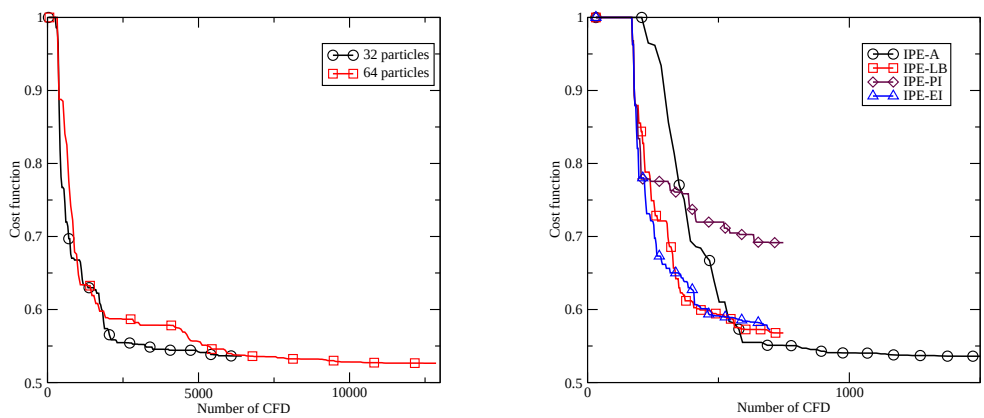
In this section we present results of aerodynamic shape optimization using different methods. In particular we compare the following methods: basic PSO, PSO with IPE and PSO using global metamodels. In order to study the efficiency and efficacy of the various methods, we apply them to solve the same optimization problem but with different number of design variables. All the methods used in this work and the FFD parameterization are implemented in our shape optimization platform FAMOSA, which utilizes MPI to perform parallel evaluation of the objective functions.

The first five iterations of IPE method are performed exactly to fill the database. The size of local database used for pre-evaluation is taken to be twice the number of design variables, and we select the closest points from the global database to the current evaluation point. Except for the adaptive criterion, only 10% of the particles are evaluated on the exact model. For PSO with and without IPE, we perform 200 iterations, and while using global metamodels, we perform 50 iterations. When reporting the reduced objective function in the tables, we consider three decimal places in the objective function and report the number of CFD evaluations required to achieve the final objective function value.

The following parameters are used for the merit functions. With the lower bound function, we use three merit functions corresponding to $\kappa = 0, 1, 2, 3$. For the probability of improvement function, we set the target T to be $(1-\alpha)\mathcal{J}_{\min}$ where $\alpha = 0.01, 0.05, 0.10, 0.20$ leading to four merit functions, and where $\mathcal{J}_{\min}$ current value of the cost function corresponding to the best solution. The smaller value of κ or α leads to a local search around the current best solution, i.e., exploitation, while the larger values leads to a more global exploration in those regions of the design space where the metamodel is inaccurate due to sparse data. For the expected improvement criterion, there is no such parameter to be specified.

Particles	Cost	#CFD
32	0.536	5664
64	0.526	11136

Table 1: Results of PSO for 8 design variables



(a) (b)

Figure 5: Results with 8 design variables: (a) Comparison of PSO with 32 and 64 particles, and (b) comparison of different pre-screening criteria in IPE method

8.1 8 design variables

This corresponds to a parameterization of $3 \times 1 \times 1$ and leads to eight design variables each of which is constrained to be in $[-200, +200]$. The results of the basic PSO are given in Table (1) using 32 and 64 particles, and convergence plots are shown in Fig. (5). We see from these results that 32 particles are sufficient, in the sense that the results do not change significantly when using 64 particles. From fig. (5) we notice that the convergence of the cost function has a fast rate initially and then becomes slow. This is a rather general character of PSO which makes it slow to converge precisely to an optimum point. The use of metamodels is hence a good strategy to accelerate the convergence of PSO and could yield significant benefits.

Next we perform optimization using IPE with 32 particles. The results are given in table (2) and figure (5b), for four different pre-screening criteria. The adaptive criterion gives the

Adaptive		LB		PI		EI	
Cost	#CFD	Cost	#CFD	Cost	#CFD	Cost	#CFD
0.535	1501	0.567	712	0.691	694	0.571	697

Table 2: IPE results using different pre-screening criteria with 8 design variables

DB size	LB		PI		EI	
	Cost	#CFD	Cost	#CFD	Cost	#CFD
24 (21)	0.523	181	0.546	113	1.000	71
48 (42)	0.524	182	0.547	178	0.615	53
96 (78)	0.523	230	0.539	226	0.626	105

Table 3: GMO results using different initial databases for 8 design variables

lowest cost function but LB and EI screening is almost as good as the adaptive one. The PI criterion fails to sufficiently reduce the cost function value.

To test the metamodel-based optimization, we make experiments with different size of the starting database. A small initial database may not give a good enough enough approximation and hence it is important to use a sufficiently large initial database. The results with different initial database sizes are shown in table (3) and in figs. (6) and we see the dramatic reduction in the number of CFD evaluations as compared to PSO and also IPE. The first column of the table gives the size of the initial sample obtained through Latin Hypercube Sampling; the numbers in the brackets indicate the actual number of points that were successfully evaluated, the remaining failing during grid deformation. In each case, we find that the lower bound criterion yields the lowest cost function value and is also very robust to change in the initial database size. The EI criterion fails to gain any reduction with the smallest database; this is most probably due to an innaccurate estimation of the variance which severely affects the performance of GMO-EI. With the two larger databases, the performance is GMO-EI is better but still not as good as with the LB and PI criteria.

Finally, in fig. (7) we compare PSO using 32 particles with the best results from IPE and GMO, both of which happen to use the lower bound criterion. We can visually see the dramatic reduction in the number of CFD evaluations when using metamodels. In particular, GMO leads to a reduction of almost 95% as compared to the 32-particle PSO and yields a better cost function value.

8.2 16 design variables

This corresponds to an FFD parameterization of $5 \times 1 \times 1$ and leads to 16 design variables each of which is constrained to be in $[-200, +200]$. The results of the basic PSO are given

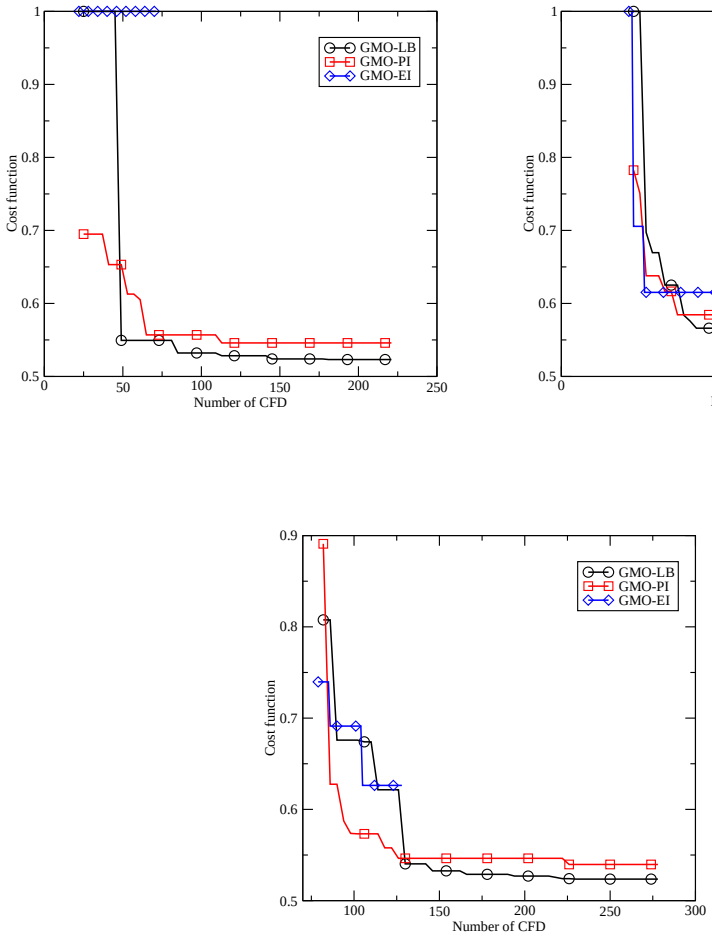


Figure 6: Results of GMO with different starting database sizes and 8 design variables:

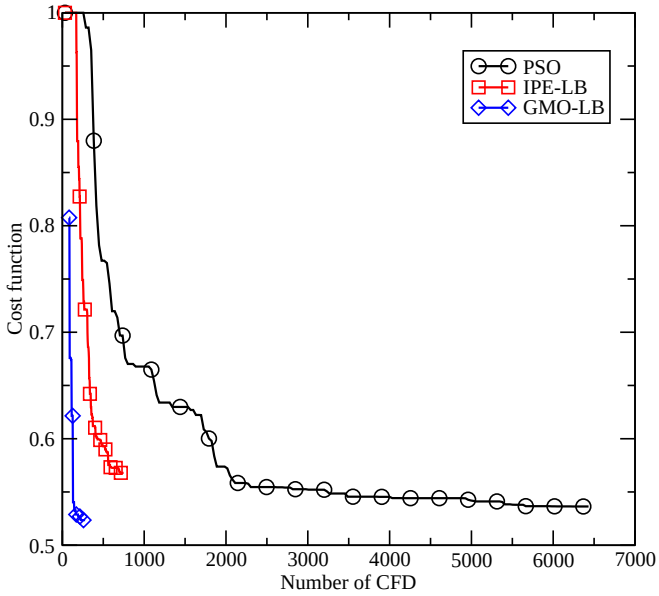


Figure 7: Comparison of different methods with 8 design variables

Particles	Cost	#CFD
32	0.531	6432
64	0.525	9920

Table 4: Results of PSO for 16 design variables

Method	Adaptive		LB		PI		EI	
	Cost	#CFD	Cost	#CFD	Cost	#CFD	Cost	#CFD
IPE	0.530	2193	0.527	1166	0.525	758	0.525	1460
GMO	-	-	0.503	218	0.523	250	0.624	115

Table 5: Results using IPE and GMO approach, and 16 design variables

in Table (4) using 32 and 64 particles, and convergence plots are shown in Fig. (8a). We see from these that 32 particles are sufficient to reduce the cost function. Next we perform optimization using the IPE approach and the results are given in Tab. (5) and Fig. (8b). In this case all the pre-screening criteria yield similar results. In Tab. (5) and Fig. (8c) we present results using the global metamodel approach with an initial database of 96 (66). The LB and PI merit functions give similar results while the EI function does not succeed in sufficiently reducing the cost function. Finally, in Fig. (8d), we compare the 32-particle PSO with the best results from IPE and GMO approach. Clearly both the metamodel approaches reduce the number of CFD evaluations, with the GMO approach again out-performing the others.

8.3 32 design variables

This test case corresponds to an FFD parameterization of $9 \times 1 \times 1$ and leads to 32 design variables. The range of each design variable is reduced to $[-100, +100]$ since a larger range leads to failure in grid deformation while generating initial database for the GMO approach. The reduction in the range does not degrade the quality of the optimum solution since it is compensated by an increase in the number of design variables. The basic PSO method gave an optimum cost function of 0.483, see table (6), which is less than the previous parameterizations. The better optimum solution is possible due to the increased degree of shape parameterization allowing the exploration of a larger set of shapes. Since the results do not change much from 64 particles to 128 particles, we take the 64 particles as the reference case for comparison with other methods.

In table (7) and figure (9b), the results of IPE approach are presented. The adaptive criterion is able to yield a solution similar to the PSO approach while the other critera do not succeed as well. The results of metamodel-based optimization are also presented in table (7) and in figure (9c) with an initial database of 192 (141) points; in this case the approach with

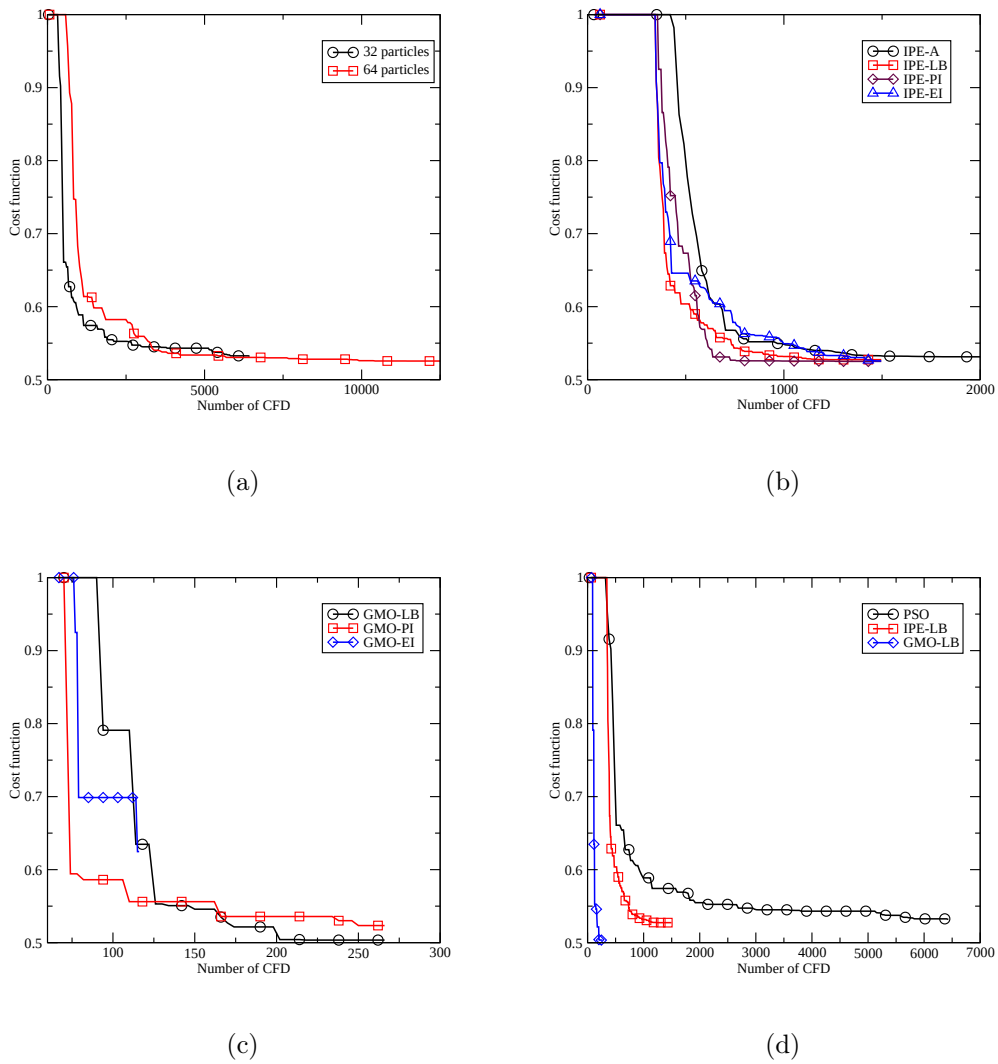


Figure 8: Results with 16 design variables: (a) Comparison of PSO with 32 and 64 particles, (b) Comparison of different pre-screening criteria in IPE, (c) Comparison of different merit functions in GMO, (d) Comparison of PSO, IPE-LB and GMO-LB.

Particles	Cost	#CFD
64	0.483	12736
128	0.478	19968

Table 6: Results of PSO for 32 design variables

Method	Adaptive		LB		PI		EI	
	Cost	#CFD	Cost	#CFD	Cost	#CFD	Cost	#CFD
IPE	0.489	5929	0.516	1424	0.554	1328	0.510	1472
GMO	-	-	0.485	305	0.531	217	0.652	155

Table 7: Results using IPE and GMO approach, and 32 design variables

lower bound merit function is able to yield a comparable solution. Finally, in figure (9d), we compare the basic PSO with the best results from the metamodel approaches; we again obtain a large reduction in the number of CFD evaluations with the use of metamodels, with the GMO approach being most efficient.

8.4 Discussion of results

The above results demonstrate the large gains in efficiency that are possible by the use of metamodels. While both local metamodels in the IPE approach and global metamodel-based optimization lead to a reduction in the number of exact evaluations (CFD), the GMO approach gives dramatically large reductions while yielding optimum solutions as good as with exact evaluations, or sometimes even better. This conforms to our expectation that a global metamodel is able to better represent the function landscape and allows the optimizer to quickly jump to the region of optimum solution in the design space. The IPE approach is still plagued by the slow convergence property of PSO and hence does not give as big gains in efficiency as the global model-based approach. However it is expected that there is a limit to the size of design space beyond which the efficacy of a global metamodel reduces due to the difficulty of constructing a good model. In the present work we are able to obtain good performance upto 32 design variables, which is a reasonably large design space.

Concerning the use of various merit functions, we have tested three of the standard functions used by the optimization community. In principle the lower bound function does not lead to dense iterates and might be expected to perform poorly. However, in our tests we found the best results with this merit function, with the probability of improvement function being a close second. The expected improvement function performed badly in most of the tests. We conjecture that this could be due to two reasons: firstly the estimate of cost function variance used in the merit functions does not account for the variance of the penalty terms, which are in effect assumed to be exact. Secondly, only one new point is added to the database in

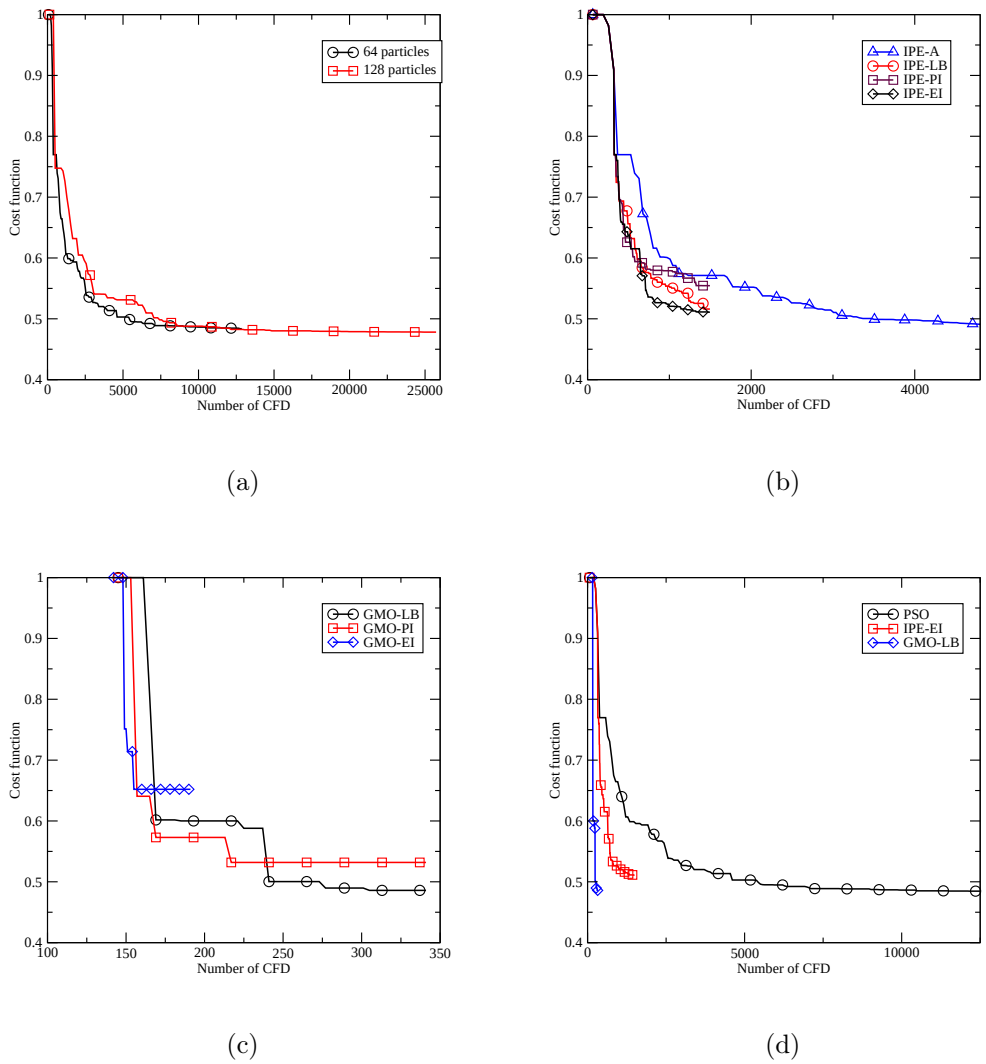


Figure 9: Results with 32 design variables: (a) Comparison of PSO with 64 and 128 particles, (b) Comparison of different pre-screening criteria in IPE, (c) Comparison of different merit functions in GMO, (d) Comparison of PSO, IPE-LB and GMO-LB.

each iteration and even this may not be added if the grid deformation procedure or the CFD fails for the particular design point. Improving these two aspects of the procedure might lead to better performance of the EI merit function.

In figure (10), we compare the pressure distribution on the wing surface and pressure contours in a span-wise cross section for the initial shape and optimized shapes. We clearly see that there is a strong shock in the initial shape which is considerably weakened by the shape optimization.

9 Summary and conclusions

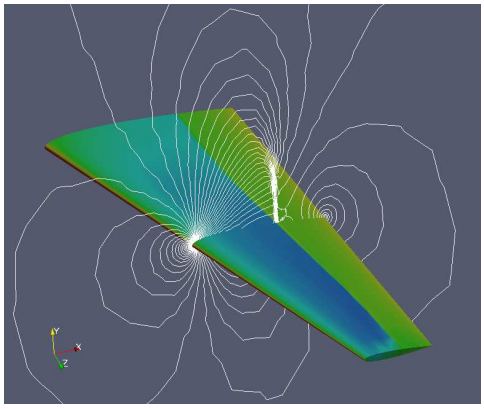
The problem of finding the minimum of an expensive cost function as in aerodynamic shape optimization is addressed via the use of metamodels. Approaches based on local and global kriging models are developed and tested. Standard optimizers like PSO are very inefficient, requiring large number of costly function evaluations and are slow to converge. Hence the use of metamodels to accelerate these methods is very important. In this work we have used local models in the framework of inexact pre-evaluation that was previously developed in the context of PSO. Global models are used as a complete replacement of the exact model and PSO is applied to the global models. Both local and global model approaches lead to considerable reduction in the computational expense. Global models gave the most dramatic reduction even for a high dimensional problem with 32 design variables.

Concerning the merit functions used in global metamodel-based optimization, the best results were obtained using lower bound function even though in principle this does not lead to dense iterates. The expected improvement criterion performed very poorly; part of the reason is attributable to the failure in grid deformation procedure, leading to a situation where the database does not get updated at the end of an iteration. In terms of good utilization of parallel computers, the lower bound and probability of improvement functions are preferable since more than one merit function can be used corresponding to both exploitation and exploration of the design space. Due to this reason they are also more robust to failure in the analysis code.

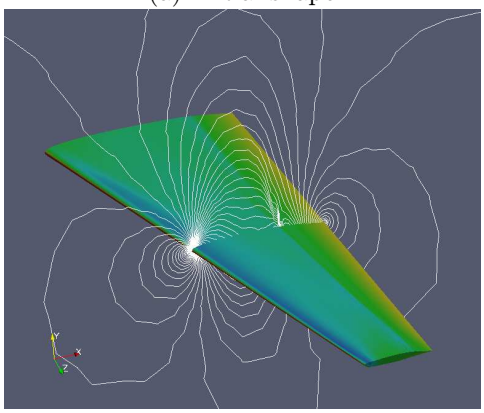
Global metamodel-based optimization is shown to be very efficient for aerodynamic shape optimization, requiring only a few hundred CFD evaluations even for a moderately high design space. The confidence in these methods would be further improved by application to more realistic problems like RANS-based shape optimization, which will form part of our future work.

Acknowledgments

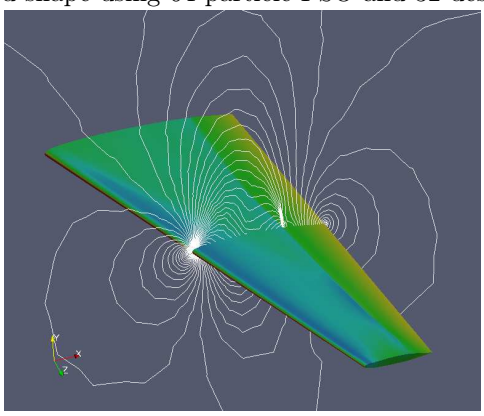
The first author would like to thank the *Indo-French Center for the Promotion of Advanced Research*, New Delhi, India, for providing financial aid to visit INRIA Sophia Antipolis under Project No. 3401-2.



(a) Initial shape



(b) Optimized shape using 64 particle PSO and 32 design variables



(c) Optimized shape using GMO-LB and 32 design variables

Figure 10: Comparison of pressure solution on initial and optimized shapes

References

[1] M. Andreoli, A. Janka, and J.-A. Desideri. Free-form deformation parameterization for multilevel 3d shape optimization in aerodynamics. Technical Report 5019, INRIA, November 2003.

[2] D. Büche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Tran. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 35(2), 2005.

[3] D. Cox and S. John. Sdo: A statistical method for global optimization. In N. M. Alexandrov and N. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the art*, pages 315–329. SIAM, Philadelphia, 1997.

[4] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

[5] J. E. Dennis and V. Torczon. Managing approximation models in optimization. In N. M. Alexandrov and N. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the art*, pages 330–347. SIAM, Philadelphia, 1997.

[6] A. Dervieux and J. A. Desideri. Compressible flow solvers using unstructured grids. Research Report 1732, INRIA, June 1992.

[7] R. Duvigneau, B. Chaigne, and J.-A. Desideri. Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using particle swarm optimization. Research Report RR-6003, INRIA, Sophia Antipolis, 2006.

[8] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Trans. Evol. Comput.*, 10(4):421–439, 2006.

[9] G. Farin. *Curves and surfaces for computer-aided geometric design*. Academic Press, 1989.

[10] P. Fourie and A. Groenwold. The particle swarm optimization in size and shape optimization. *Structural and Multidisciplinary Optimization*, 23(4), 2002.

[11] H. M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, 2001.

[12] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[13] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, Perth, Australia, 1995.

[14] H. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

[15] M. Locateli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10:57–76, 1997.

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

[17] Z. Michalewics. *Genetic algorithms + data structures = evolutionary programs.* AI Series. Springer-Verlag, New York, 1992.

[18] P. Miller. Swarm behaviour. *National Geographic*, July 2007. available online at `http://www7.nationalgeographic.com/ngm/0707/feature5/`.

[19] C. Praveen and R. Duvigneau. Low cost pso using inexact pre-evaluation: Application to aerodynamic shape design. *Comput. Methods Appl. Mech. Engrg.*, 198:1087–1096, February 2009.

[20] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

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

[22] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *International Conference on Evolutionary Computation*, 1998.

[23] V. Torczon and M. W. Trosset. Using approximations to accelerate engineering design optimization. In *ISSMO/NASA First Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization*, June 14-27 1998.

[24] A. Torn and A. Zilinskas. *Global Optimization.* Springer, Berlin, 1987.

[25] G. Venter and J. Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA Journal*, 41(8), 2003.

[26] D. N. Wilke. Analysis of the particle swarm optimization algorithm. Master's thesis, Department of Mechanical and Aeronautical Engineering, University of Pretoria, 2005.

[27] 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.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399