Metamodel-assisted particle swarm optimization and application to aerodynamic shape optimization

Praveen Chandrashekarappa, Regis Duvigneau

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Metamodel-assisted particle swarm optimization and application to aerodynamic shape optimization

Praveen. C — Regis Duvigneau

N° 6397

December 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6397--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Metamodel-assisted particle swarm optimization and application to aerodynamic shape optimization

Praveen. C , Regis Duvigneau

Thème NUM — Systèmes numériques
Projet OPALE

Rapport de recherche n° 6397 — December 2007 — 39 pages

Abstract: Modern optimization methods like Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) have been found to be very robust and general for solving engineering design problems. They require the use of large population size and may suffer from slow convergence. Both of these lead to large number of function evaluations which can significantly increase the cost of the optimization. This is especially so in view of the increasing use of costly high fidelity analysis tools like CFD. Metamodels also known as surrogate models, are a cheaper alternative to costly analysis tools. In this work we construct radial basis function approximations and use them in conjunction with particle swarm optimization in an inexact pre-evaluation procedure for aerodynamic design. We show that the use of mixed evaluations by metamodels/CFD can significantly reduce the computational cost of PSO while yielding optimal designs as good as those obtained with the costly evaluation tool.

Key-words: Particle swarm optimization, metamodels, radial basis functions, aerodynamic shape optimization

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

Optimisation par essaim de particules assistée par métamodèles et application à l'optimisation de forme aérodynamique

Résumé : Les méthodes d'optimisation modernes comme les algorithmes génétiques et l'optimisation par essaim de particules sont des méthodes robustes et générales pour résoudre des problèmes de conception en ingénierie. Elles nécessitent cependant d'utiliser une population de grande taille et peuvent souffrir de vitesse de convergence médiocre. Cela conduit à un nombre d'évaluations de la fonction objectif important et des coûts prohibitifs. C'est particulièrement le cas lors d'utilisation d'outils d'analyse sophistiqués et coûteux, comme les solveurs CFD. Les métamodèles sont une alternative moins coûteuse que ces outils d'analyse. Dans cette étude, on construit des approximations de type fonctions à base radiale et on les utilise en conjonction avec une méthode d'optimisation par essaim de particules dans une procédure de pré-évaluation inexacte pour la conception optimale en aérodynamique. On montre que l'utilisation d'évaluations mixtes métamodèles/CFD peut réduire significativement le coût de calcul pour une méthode d'optimisation par essaim de particules, tout en conduisant à une solution aussi performante.

Mots-clés : Optimisation par essaim de particules, Métamodèles, fonctions à base radiale, Conception optimale en aérodynamique

1 Introduction

Optimization methods like genetic algorithms, particle swarm optimization, etc. have been found to be ideal for solving large scale problems. Among their many advantages are their ability to handle non-smooth functions (since gradient information is not required) and the possibility of finding global optimal solutions. A distinguishing feature of these methods is that they operate with a *population/swarm*, i.e., they make use of multiple candidate solutions at each step of their iteration. This requires the computation of the *cost/fitness* function for each candidate in every optimization iteration. The ability to locate the global optimum depends on sufficient exploration of the design space which requires using a sufficiently large population size. This is especially true when the cost function is multi-modal and the dimension of the design variable space is high. With the increasing use of high fidelity models, e.g. Navier-Stokes equations for flow analysis, the computation of the cost function for a single design can be costly in terms of time and resource utilization. The combination of such high-fidelity analysis tools with population-based optimization techniques can render them impractical or severely limit the size of the population that can be used.

To overcome this barrier, several researchers have used surrogate models [2, 9, 6, 14, 20] in place of the costly evaluation tool. These surrogate models are inexpensive compared to the exact model. There are several ways in which a surrogate model can be developed.

- Data-fitting models: An approximation to the cost function is constructed using available data. This data may be either generated specifically for constructing the model or may be taken from the initial few iterations of the optimization method. Examples of data-fitting models are polynomials (usually quadratic, also known as response surface models), artificial neural networks (like multi-layer perceptron, radial basis function networks) and Gaussian process models (kriging). These models can be either global, which make use of all available data, or local, which make use of only a small set of data around the point where the function is to be approximated. Global models have been used as a complete replacement of the original cost function with optimization being carried out on the surrogate model. Local models have been typically used to pre-screen promising designs which are then evaluated using the exact cost function. This leads to a reduction in computational cost since the number of exact function evaluations is reduced.

- Variable convergence models: The cost function usually depends on the numerical solution of a PDE. Most numerical methods are iterative in nature and contain a stopping criterion which is measured in terms of a solution residual. To get an accurate solution a small value of the residual is usually used. Such an accurate solution maybe unnecessary when all we want is an estimate of a cost function which is usually some integral that converges much faster. In such a situation the stopping criterion can be relaxed thereby considerably reducing the time taken by a single computation.

- Variable resolution models: In these models, a hierarchy of grids is used and the surrogate model is just the costly evaluation tool but run on a coarse grid.

- Variable fidelity models: In these models, a hierarchy of physical models are used, for example Euler equations (surrogate model) and RANS equations (exact model). Even when a high fidelity model like RANS is used, one can use a wall function approximation as a surrogate model and a turbulence model applied upto to the wall as the exact model.

Data fitting models have been extensively used for optimization of costly functions. Quadratic models were frequently used in the past but their lack of accuracy has led to the development of more sophisticated approximation methods like neural networks, radial basis functions and kriging. There are several variations in the use of metamodels for optimization. In off-line trained methods, a metamodel is first constructed by generating a set of data points in the design space and evaluating the cost function at these points. This metamodel is then used to optimize the cost function without recourse to the exact function. The success of this method relies on the ability to construct an accurate metamodel which is doubtful for realistic problems which usually involve large number of design variables and complex function landscapes. On-line trained methods construct and update the metamodel as and when required and are closely integrated into the optimization loop. Whenever a new function value is available, the metamodel is updated and the optimization proceeds using the new metamodel. The metamodel becomes progressively more accurate as more and more data points are included in its construction.

Evolutionary algorithms have been used with metamodels to reduce the cost of exact function evaluations. Giannakoglou et al. [10] use local metamodels to pre-evaluate individuals in the population; then a small percentage of individuals is selected for exact evaluation and the results are stored in a database. The standard EA operators are applied to the individuals using a combination of exact and approximate function values. Buche at al. [2] construct a sequence of local kriging models and optimize them using evolutionary algorithms. The metamodel is constructed to model a local region around the best current individual. The data used to construct the local models is continuously adjusted based on the location of the best point discovered until then. At each iteration the optimal solution of the metamodel is evaluated on the exact function and the result added to the database. Zhou et al. [27] use a combination of global and local metamodels to accelerate EA; the global model is used to pre-evaluate all individuals in the population and a small percentage of promising individuals is optimized using a trust-region enabled gradient-based local search using a local metamodel. The exact function evaluations generated during the gradient search are added to a database. The modified individuals are replaced in the population and the standard EA operators are applied.

Emmerich et al. [6] apply kriging models in an IPE frame-work using evolutionary algorithms to solve multi-objective optimization problems. They also study the performance of different pre-screening criteria and extend them to the multi-objective case.

Function approximations that make use of both function and gradient values can also be constructed [15], the gradients being efficiently computed using adjoint methods. Giannakoglou et al. [12] construct neural network and RBF approximations using both function and gradient values and show that these models are more accurate than pure function based metamodels. Both local and global metamodels are used together with an IPE strategy. However such approximations are costlier to construct since they contain large number of parameters.

In this work we consider data-fitting models, particularly radial basis functions which have been found to be effective in interpolation of high dimensional data with small number of data points as compared to polynomial based methods. Such metamodels have already been used to improve the efficiency of genetic algorithms. Briefly, a genetic algorithm can be described as follows:

1. Evaluate the fitness of all individuals in the population
2. Apply the selection operator to eliminate non-promising individuals
3. Apply the crossover operator to generate offsprings from the selected individuals
4. Apply mutation operator to modify randomly the offsprings

Giannakoglou [9] has proposed a two-level evaluation strategy, called Inexact Pre-Evaluation (IPE), to reduce the computational time related to GAs. It relies on the observation that numerous cost function evaluations are useless, since numerous individuals do not survive to the selection operator. Hence, it is not necessary to determine their fitness accurately. The strategy proposed by Giannakoglou consists in using metamodels to pre-evaluate the fitness of the individuals in the population. Then only a small portion of the population which corresponds to the most promising individuals are accurately evaluated using the original and expensive model.

Inspired by the success of GAs combined with metamodels and IPE, we study the application of a similar strategy to particle swarm optimization. PSO also requires a large number of function evaluations since it requires a large number of particles to effectively locate the optimum. We propose a new pre-screening criterion which is specific to PSO. The proposed algorithm is applied to the aerodynamic shape optimization of a supersonic business jet and a transonic wing. In both cases substantial reduction in the number of CFD evaluations is achieved while finding optimal shapes that are as good as in the case of CFD evaluations alone.

2 Radial basis function models

Radial basis function approximations were introduced by Hardy [13] to represent topographical surfaces given sets of sparse scattered measurements. They have been found to

Name	$\phi(r)$	Parameters	Smoothness
Gaussian	$\exp(-r^2/a^2)$	–	C^∞
Inverse multiquadric	$(r^2+a^2)^{s/2}$	$s<0$	C^∞
Sobolev spline	$r^s K_s(r)$	$s>0$	$C^{\lfloor s \rfloor}$

Table 1: Unconditionally positive definite functions

be very accurate for interpolation of arbitrarily scattered data [23]. There are two types of radial basis functions, piecewise smooth and infinitely smooth. The piecewise smooth RBFs lead to an algebraic rate of convergence to the desired function as the number of points increase, whereas the infinitely smooth RBFs yield a spectral or even faster rate of convergence, assuming of course that the desired function itself is smooth. Radial basis function interpolation seeks an approximation $\hat{f}$ of the form

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \Phi(x - x_n)$$

where $\Phi(x) = \phi(\|x\|)$ is a radial function. Examples of radial basis functions are given in Table 1. In the RBF terminology, the positions $x_n, n = 1, \ldots, N$ are called the *RBF centers*.

The coefficients $w = [w_1, w_2, \ldots, w_N]^\top$ are determined from the interpolation conditions

$$\hat{f}(x_m) = f_m, \quad m = 1, 2, \ldots, N$$

which can be written in matrix form as[1]

$$Aw = F$$

with $F = [f_1, f_2, \ldots, f_N]^\top$. The matrix A has elements $A_{mn} = \Phi(x_m - x_n)$ and is symmetric since Φ is a radial function. For the functions in Table 1, the matrix A is also *positive-definite* for every set of N distinct points in $\mathbb{R}^d$; this is true for any value of N or d. Such functions are said to be *unconditionally positive definite*. There are radial basis functions which do not have this property; some examples are given in Table 2. In this case, we can make the RBF *conditionally positive definite* by adding a suitable family of polynomials.

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \Phi(x - x_n) + \sum_{l=1}^{M(q)} \alpha_l p_l(x)$$

[1]For brevity of notation, the subscript N will be dropped in this section.

Name	$\phi(r)$	Parameters	q
Spline	r^s	$s > 0, s \notin 2\mathbb{N}$	$q \geq \lceil s/2 \rceil$
Thin-plate spline	$r^s \log r$	$s > 0, s \in 2\mathbb{N}$	$q > s/2$
Multi-quadric	$(r^2 + a^2)^{s/2}$	$s > 0, s \notin 2\mathbb{N}$	$q \geq \lceil s/2 \rceil$

Table 2: Conditionally positive definite functions

where $p_l, l = 1, ...M(q)$ forms a basis for $\mathbb{P}_q$, the space of polynomials of degree $\leq q$. The equations to determine the coefficients w and α are

$$\begin{aligned} \sum_{n=1}^{N} w_n \Phi(x_m - x_n) + \sum_{l=1}^{M} \alpha_l p_l(x_m) &= f_m, \quad m = 1, \ldots, N \\ \sum_{n=1}^{N} w_n p_l(x_n) &= 0, \quad l = 1, \ldots, M \end{aligned}$$

The above set of equations is guaranteed to have a unique solution for any disjoint data set. If $N \geq M$ and the unknown function $f \in \mathbb{P}_q$ then the above interpolation will exactly reproduce the function.

Note: The basis functions can be either decreasing (for example the Gaussian) or increasing (for example the thin plate splines) functions, and lead to full matrices. There are also basis functions with compact support which lead to sparse coefficient matrices; however these give only algebraic convergence while the smooth basis functions give exponential convergence.

Note: In the present work we consider only interpolating RBFs which exactly reproduce the input data. One can also use fitted RBF models which may not exactly interpolate the data [11]. RBF models can also be considered where the centers do not coincide with the location of the data points [9].

2.1 Effect of attenuation factor

The attenuation factor in radial basis functions has a critical influence on the accuracy of the interpolation model. We illustrate this with a numerical example. The RBF interpolant is constructed for a test function $f(x) = x(1-x)\sin(2\pi x)$ using the data from 10 equally spaced points in $[0, 2]$. The error between the interpolant and the exact function is evaluated on a grid of 100 points. Table (3) shows the error and condition number for different attenuation factors. We notice that for both very small and very large values of a the error is high. The condition number of the coefficient matrix A is seen to increase with increasing values of the attenuation factor. The high error at large

a	0.01	0.1	0.5	1.0	2.0
Mean error	1.29	0.142	0.0114	0.0978	16.3
Max error	1.24	0.213	0.0181	0.156	19.6
Condition no.	1.0	3.4	1.1×10^9	3.4×10^{14}	2.8×10^{18}

Table 3: Effect of attenuation factor on the error of RBF interpolation for test function $f(x) = x(1-x)\sin(2\pi x)$

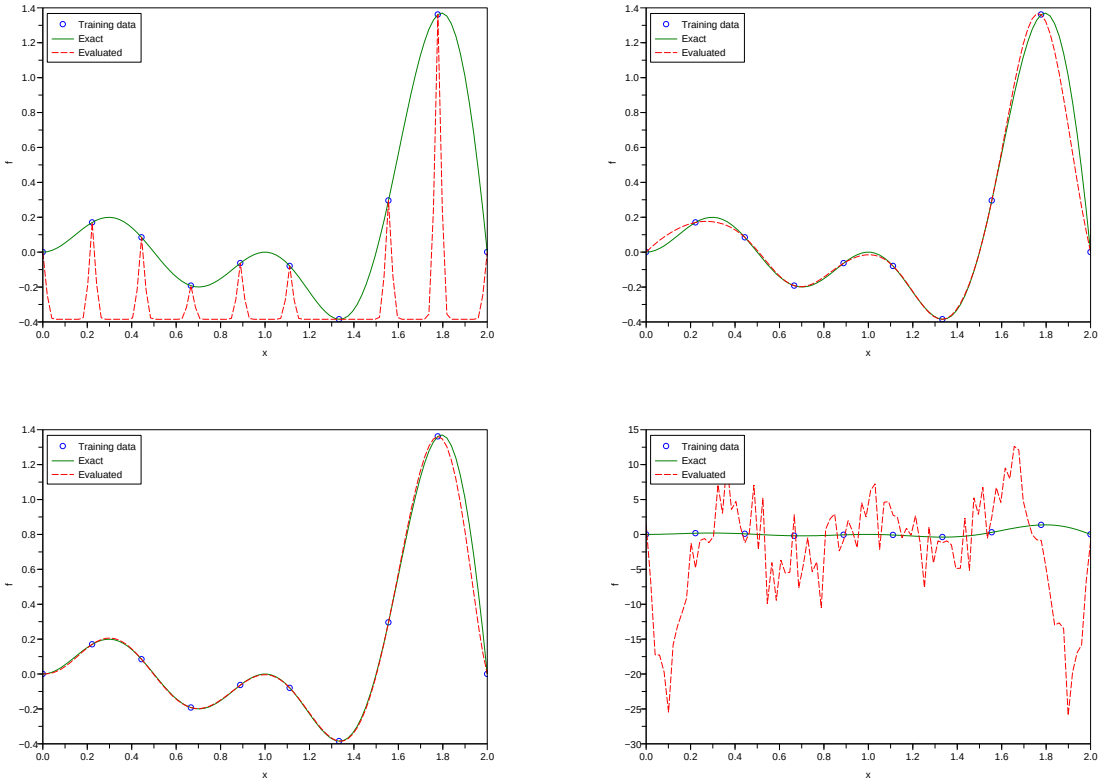


Figure 1: RBF interpolant for test function $f(x) = x(1-x)\sin(2\pi x)$: (a) $a = 0.01$, (b) $a = 0.1$, (c) $a = 0.5$, and (d) $a = 1.0$

values of a is due to the numerical instability resulting from ill-conditioning of the matrix A. Note that as long as the condition number is not very high, the interpolant exactly reproduces the training data. The RBF interpolant and the exact function are plotted in Figures 1. Figure 2 shows the variation of average error and condition number with attenuation factor. We notice that the error has a minimum for a particular value of attenuation factor $a_o \approx 0.45$ with average error of 2.87e-4. A theoretical justification for the existence of an optimal value for the attenuation factor has been recently given in [17].

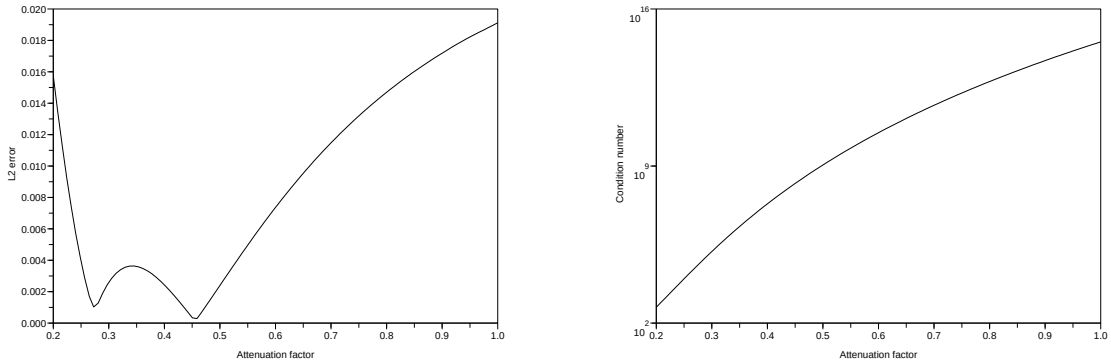


Figure 2: Variation of L_2 error and condition number with attenuation factor

2.2 Optimization of attenuation factor

In [21], several empirical methods for choosing the attenuation factor are discussed; see this paper for further references. Some researchers had expressed the hope that there may be a universally optimal value of the attenuation factor. Based on numerical experiments, Rippa [21] concludes that the best attenuation factor depends on the number and distribution of data points, on the function f and on the precision of the computation[2].

An obvious way to optimize the attenuation factor is to divide the available data into two subsets, a *training set* and a *testing set*; we can use the training set to construct the RBF model and use it to evaluate the function on the testing set. The attenuation factor can be optimized so that the error of interpolation on the testing set is minimized. However, in practical optimization problems, we may not have sufficient number of data points to perform the above sub-division. An alternative approach is the *leave-one-out* technique.

Let $\hat{f}^{(n)}(x;a)$ denote the RBF interpolant constructed using the data points

$$X^{(n)} = \{x_1, x_2, \ldots, x_{n-1}, x_{n+1}, \ldots, x_N\}$$

i.e., by ignoring the n'th data point in the full data set. This interpolant can be used to estimate the function value at the ignored point x_n and the corresponding error

$$E_n = f_n - \hat{f}^{(n)}(x_n;a)$$

can also be computed. By ignoring each data point successively and constructing an interpolant we obtain an error vector

[2] An interesting result in [3] states that there exists an attenuation factor and location of the RBF centers which leads to an interpolation formula uniformly accurate for all functions in a compact subset of $C(K)$, where K is a compact subset of $\mathbb{R}^d$.

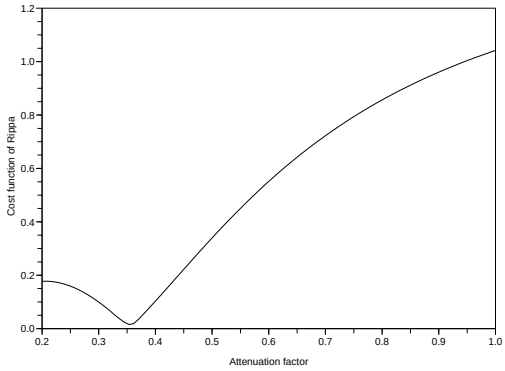


Figure 3: Variation of $C(a)$ with attenuation factor for test function $f(x) = x(1-x)\sin(2\pi x)$

$$E(a) = [E_1, E_2, \ldots, E_N]^\top$$

Rippa [21] suggests minimizing some norm of the above error vector with respect to the attenuation factor, i.e., find a_* such that

$$a_* = \text{argmin } \|E(a)\|$$

Rippa gives some numerical examples to show that the function $C(a) = \|E(a)\|$ behaves similar to the actual error. In particular, they achieve their minimum at similar values of attenuation factor. Figure (3) plots $C(a)$ for test function which indicates the existence of an optimum value $a_* \approx 0.353$.

2.3 Efficient implementation

The computation of $C(a)$ requires the solution of N linear equations each of order $(N-1) \times (N-1)$. If the linear system is solved using LU decomposition the total number of operations is of order N^4 which can be very expensive even for moderate size data sets. An efficient algorithm is given in [21] which requires only one LU decomposition at a cost of $O(N^3)$. Below, we essentially reproduce the algorithm as given in [21].

The RBF interpolant using the data points $X^{(n)}$ is given by

$$\hat{f}^{(n)}(x) = \sum_{m=1, m\neq n}^{N} w_m^{(n)} \Phi(x - x_m)$$

where the coefficients $w^{(n)}$ are determined by solving the interpolation problem $\hat{f}^{(n)}(x_r) = f(x_r), r = 1, \ldots, n-1, n+1, \ldots, N$. We denote this in matrix notation as

$$A^{(n)} w^{(n)} = F^{(n)}$$

where $A^{(n)}$ is obtained from A by removing the n'th row and n'th column, and $F^{(n)} = (f_1, \ldots, f_{n-1}, f_{n+1}, \ldots, f_N)^\top$. We note that if $y \in \mathbb{R}^N$ is such that $y_n = 0$ then

$$Ay = z \Longrightarrow A^{(n)}(y_1, \ldots, y_{n-1}, y_{n+1}, \ldots, y_N)^\top = (z_1, \ldots, z_{n-1}, z_{n+1}, \ldots, z_N)^\top \qquad (1)$$

Now consider the solution $u^{[n]}$ to the system

$$Au^{[n]} = e^{[n]} \qquad (2)$$

where $e^{[n]}$ is the n'th column of the $N \times N$ identity matrix. It is easy to verify that $u_n^{[n]} \neq 0$. Indeed, if $u_n^{[n]} = 0$ then by (1) and (2) we conclude that

$$A^{(n)}(u_1^{[n]}, \ldots, u_{n-1}^{[n]}, u_{k+1}^{[n]}, \ldots, u_N^{[n]})^\top = 0$$

which implies, by the non-singularity of $A^{(n)}$ that $u^{[n]} = 0$, which is impossible because $u^{[n]}$ is the solution to (1). Let us now consider the vector $v^{[n]} \in \mathbb{R}^N$ defined by

$$v^{[n]} = w - \frac{w_n}{u_n^{[n]}} u^{[n]}$$

Then we have that

$$Av^{[n]} = Aw - \frac{w_n}{u_n^{[n]}} Au^{[n]} = F - \frac{w_n}{u_n^{[n]}} e^{[n]} = \left(f_1, \ldots, f_{n-1}, f_n - \frac{w_n}{u_n^{[n]}}, f_{n+1}, \ldots, f_N \right)^\top$$

and since $v_n^{[n]} = 0$, we use (1) to conclude that

$$w^{(n)} = \left(v_1^{[n]}, \ldots, v_{n-1}^{[n]}, v_{n+1}^{[n]}, \ldots, v_N^{[n]} \right)^\top$$

This implies that

$$\begin{aligned}
\hat{f}^{(n)}(x_n) &= \sum_{m=1,m\neq n}^{N} w_m^{(n)} \Phi(x_n - x_m) \\
&= \sum_{m=1,m\neq n}^{N} v_m^{[n]} \Phi(x_n - x_m) \\
&= \sum_{m=1}^{N} v_m^{[n]} \Phi(x_n - x_m) \\
&= \left(A v^{[n]}\right)_n \\
&= f_n - \frac{w_n}{u_n^{[n]}}
\end{aligned}$$

which gives the following simple formula for the error of interpolation at the excluded point x_n

$$E_n = f_n - \hat{f}^{(n)}(x_n) = \frac{w_n}{u_n^{[n]}}$$

If we use LU decomposition to solve the linear equation systems, the cost of one LU decomposition of the matrix A is $O(N^3)$, while the cost of solving the N linear equations (2) is $O(N^2)$ so that the total cost is $O(N^3)$.

Rippa has used Brent's method which is a bracketing algorithm for locating the minimum. In our tests we found that it is not possible to predict in advance a suitable bracketing interval since this depends on the data set, the function and dimension of the problem. Hence we have used Particle Swarm Optimization (PSO) to locate the minimum of the cost function $C(a)$. Since this is a one dimensional minimization problem a small number of particles should be sufficient; we have used five particles in the swarm and tests indicate that the minimum point can be located with less than 100 iterations.

2.4 Some practical issues

The radial basis functions depend on the Euclidean distance between two data points. If the components of the independent variables $x \in \mathbb{R}^d$ have widely different scales then the Euclidean norm may not be appropriate. In [9] a weighted norm has been used in place of the Euclidean norm, where the weights depend on the gradient of the function. Here, the independent variables $\{x_n, n = 1, \ldots, N\}$ and function values $\{f_n, n = 1, \ldots, N\}$ are scaled before constructing the RBF model. The independent variables $x \in \mathbb{R}^d$ are scaled so that each component of x lies in the interval $[-1/2, +1/2]$ while function values are scaled to lie in the interval $[0, 1]$. If the function is constant, then in the scaled space all the function values will be zero and the coefficients w are also zero. A constant function

is thus recovered exactly for any value of attenuation factor. Note that this avoids the difficulty of reproducing constant functions with RBF which otherwise requires very flat ($a \to \infty$) basis functions.

The coefficient matrix A can become ill-conditioned for *large* values of attenuation factor, and also for very large and dense data sets. What is a large attenuation factor depends on the number of data points, their distribution and the dimension d. The best attenuation factor usually leads to a highly ill-conditioned coefficient matrix. An uncertainty principle established in [26] states that the attainable error and the condition number of the RBF interpolation matrix cannot both be small at the same time. When the matrix is highly ill-conditioned, it is not possible to compute the interpolant with finite precision arithmetic since the solution of linear algebraic equations becomes unstable. In [8] a method is proposed to compute the RBF interpolant for such ill-conditioned cases. However this is costly for our present purpose and we use a simple limiting approach. While minimizing the cost function $C(a)$ we compute the condition number of the coefficient matrix A; if it is larger than some specified value, then the cost function is not computed but is set to an arbitrarily large positive number. The particles in PSO are then naturally pulled towards regions of well conditioned attenuation factors. In the present computations, the upper limit on the condition number is set to $1/\epsilon$ where ϵ is the machine precision.

3 Particle swarm optimization

PSO is modeled on the behaviour of a swarm of animals when they hunt for food or avoid predators [19]. In nature a swarm of animals is found to exhibit very complex behaviour[3] 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. Each particle remembers the best position it has discovered and also knows the best position discovered by its neighbours and the whole swarm. 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 [5]. The algorithm is given for a function minimization problem

$$\min_{x_l \le x \le x_u} f(x)$$

[3]See a video of a flock of birds performing highly coordinated maneuver `http://youtube.com/watch?v=XH-groCeKbE`

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 $f(x_k^n), k = 1, \ldots, K$
4. Update the local and global memory

$$x_{*,k}^n = \underset{0 \leq s \leq n}{\operatorname{argmin}} f(x_k^s), \quad x_*^n = \underset{0 \leq s \leq n, 1 \leq k \leq K}{\operatorname{argmin}} f(x_k^s) \tag{3}$$

5. Update the particle velocities

$$v_k^{n+1} = \omega^n v_k^n + c_1 r_{1,k}^n (x_{*,k}^n - x_k^n) + c_2 r_{2,k}^n (x_*^n - x_k^n) \tag{4}$$

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

$$x_k^{n+1} = x_k^n + v_k^{n+1} \tag{5}$$

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 3, else STOP.

In the original algorithm proposed by Kennedy and Eberhart [16] the random numbers r_1, r_2 are scalars, i.e., one random number is used for each 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 [25] 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. We investigate the performance of these versions on a test case of aerodynamic optimization of a supersonic business jet.

4 Metamodel assisted PSO with IPE

Like genetic algorithms, PSO is also 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 the particle. Hence in the context of PSO also, an inexact pre-evaluation strategy seems to be advantageous in identifying promising particles i.e. particles whose local memory is expected to improve, which can then be evaluated on the exact function. 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 propose a metamodel-assisted PSO with inexact pre-evaluation 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. In the present work $N_e = 10$ is used.

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 $f(x_k^n), k = 1, \ldots, K$ using the exact model, else compute the cost function using metamodel $\tilde{f}(x_k^n), k = 1, \ldots, K$.
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*

$$x_{*,k}^n = \underset{0 \leq s \leq n}{\operatorname{argmin}} f(x_k^s), \quad x_*^n = \underset{0 \leq s \leq n, 1 \leq k \leq K}{\operatorname{argmin}} f(x_k^s) \tag{6}$$

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

$$v_k^{n+1} = \omega^n v_k^n + c_1 r_{1,k}^n (x_{*,k}^n - x_k^n) + c_2 r_{2,k}^n (x_*^n - x_k^n) \tag{7}$$

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

$$x_k^{n+1} = x_k^n + v_k^{n+1} \tag{8}$$

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 3, else STOP.

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. Giannakoglou [6] discusses several pre-screening criteria based on the estimated fitness function and variance of the estimation whenever available, as in the case of gaussian random process models. The pre-screening criteria are based on the notion of *improvement*. Let $f_{\min}$ be the current minimum function value and $\hat{f}(x)$ be the function value predicted by the metamodel for a new design point x. We can define an index of improvement for the design x as

$$I(x) = \begin{cases} 0 & \text{if } \hat{f}(x) > f_{\min} \\ f_{\min} - \hat{f}(x) & \text{otherwise} \end{cases} \tag{9}$$

Designs with larger value of this index are likely to lead to a reduction in the cost function and should be evaluated on the exact function. Some metamodels like kriging also give an estimate of the error in the approximation. This information can be useful to explore those regions of the design space which are not sufficiently probed. We do not consider these other criteria but refer to [6] for further details.

In the present work we use interpolating RBF metamodels which do not provide an estimate of the variance. Hence the pre-screening is based only on the estimated cost function value and we investigate two different criteria;

- After the IPE phase, the particles are sorted in the order of increasing cost function and a specified percentage of the *best* particles i.e. those with small cost function values, are selected for exact evaluation.
- We also propose a new pre-screening criterion for PSO as follows: the set S^n consists of all particles whose cost function is predicted to reduce in the IPE phase, i.e.,

$$S^n = \{k : \tilde{f}(x_k^n) < f(x_{*,k}^{n-1})\} \tag{10}$$

The second criterion is similar to the index of improvement but the minimum function value is that of the individual particles memory. All particles whose index is positive (non-zero) are evaluated on the exact function. Note that we do not specify any percentage as

in the case of GA with IPE. The number of exact function evaluations is automatically determined and we expect this number to adapt itself as the cost function is progressively reduced. Note that in this PSO+IPE approach, both the local and global memories always consist of exactly evaluated particles.

5 Parameterization using the Free-Form Deformation approach

A critical issue in parametric shape optimization is the choice of the shape parameterization. The objective of the parameterization is to describe the shape, or the shape modification, by a set of parameters which are considered as design variables during the optimization procedure. Parameterization techniques in shape optimization have to fulfill several practical criteria:

- the parameterization should be able to take into account complex geometries, possibly including constraints and singularities
- the number of parameters should be as small as possible, since the stiffness of the shape optimization numerical formulation increases abruptly with the number of parameters
- the parameterization should allow to control the smoothness of the resulting shapes

A survey of shape parameterization techniques for multi-disciplinary optimization, which are analyzed according to the previous criteria, is proposed by Samareh [22]. In accordance with his conclusions, the Free-Form Deformation (FFD) technique [24] is adopted in the present study, since it provides an easy and powerful framework for the deformation of complex shapes, as those encountered in aerodynamics or electromagnetics.

The FFD technique originates from the Computer Graphics field [24]. 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.

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:

$$\Delta q = \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} B_i^{n_i}(\xi_q) B_j^{n_j}(\eta_q) B_k^{n_k}(\zeta_q) \Delta P_{ijk}. \tag{11}$$

$B_i^{n_i}$, $B_j^{n_j}$ and $B_k^{n_k}$ are the Bernstein polynomials of order n_i, n_j and n_k (see for instance [7]):

$$B_p^n(t) = C_n^p \, t^p \, (1-t)^{n-p}. \tag{12}$$

$(\Delta P_{ijk})_{0 \leq i \leq n_i, 0 \leq j \leq n_j, 0 \leq k \leq n_k}$ 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.

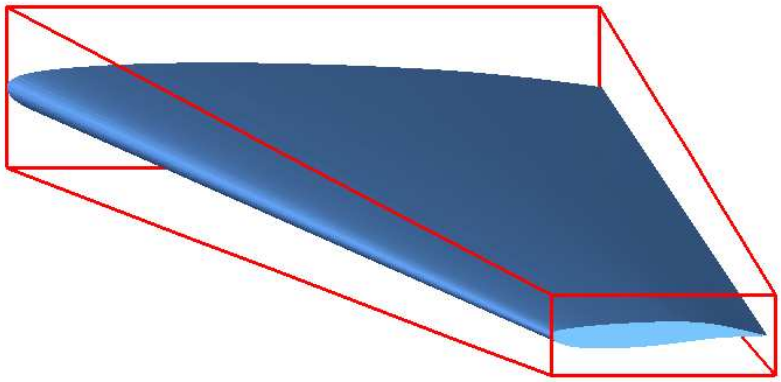


Figure 4: Example of FFD lattice (red) around a wing.

The FFD technique described above is well suited to complex shape optimization, thanks to the following properties:

- the initial shape can be exactly represented (no deformation occurs when all weighting coefficients are zero) ;
- the deformation is performed whatever the complexity of the shape (this is a free-form technique) ;
- geometric singularities can be taken into account (the initial shape including its singularities is deformed) ;
- the smoothness of the deformation is controlled (the deformation is ruled by Bernstein polynomials) ;
- the number of design variables depends on the user's choice (the deformation is independent of the shape itself) ;

- it nicely deals with multi-level representation (thanks to the Bézier degree elevation property).

The FFD technique is implemented in the shape optimization procedure and is used to control the shape deformation for applications in both aerodynamics and electromagnetics.

6 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 :

$$\frac{\partial W}{\partial t} + \frac{\partial F_1(W)}{\partial x} + \frac{\partial F_2(W)}{\partial y} + \frac{\partial F_3(W)}{\partial z} = 0, \tag{13}$$

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 :

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{pmatrix} \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{pmatrix} \quad F_3(W) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{pmatrix}. \tag{14}$$

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

$$p = (\gamma - 1)(E - \frac{1}{2}\rho \| \overrightarrow{U} \|^2), \tag{15}$$

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 (13) at the mesh node s_i is obtained by integrating (13) 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 :

$$Vol_i \frac{\partial W_i}{\partial t} + \sum_{j \in N(i)} \Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = 0, \tag{16}$$

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 (14) 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 :

$$\Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = \frac{\overrightarrow{F}(W_i) + \overrightarrow{F}(W_j)}{2} \cdot \overrightarrow{\sigma}_{ij} - |A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})| \frac{W_j - W_i}{2}. \tag{17}$$

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

$$A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})(W_j - W_i) = (\overrightarrow{F}(W_j) - \overrightarrow{F}(W_i)) \cdot \overrightarrow{\sigma}_{ij}. \tag{18}$$

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 (16) 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 (16), which yields :

$$\frac{Vol_i}{\Delta t} \delta W_i + \sum_{j \in N(i)} \Phi(W_i^{n+1}, W_j^{n+1}, \overrightarrow{\sigma}_{ij}) = 0, \tag{19}$$

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

$$\left(\frac{Vol_i}{\Delta t} + J_i^n \right) \delta W_i = - \sum_{j \in N(i)} \Phi(W_i^n, W_j^n, \overrightarrow{\sigma}_{ij}). \tag{20}$$

Here, J_i^n is the jacobian matrix of the first order numerical fluxes, whereas the right hand side of (20) is evaluated using high order approximations. The resulting integration scheme provides a high order solution of the problem. More details can be found in [4].

7 Test case 1: Supersonic Business Jet Optimization

We consider the drag minimization of a supersonic business jet at a Mach number of $M_\infty = 1.7$ and angle of attack $\alpha = 1^o$ subject to a constraint on the lift, volume and thickness. The constraints are implemented by adding penalty terms to the cost function. The governing equations are the Euler equations of inviscid compressible flow; hence the drag is only composed of lift-induced drag and wave drag. The wave drag has contributions due to lift and volume; a reduction in drag can be obtained just by reducing the volume.

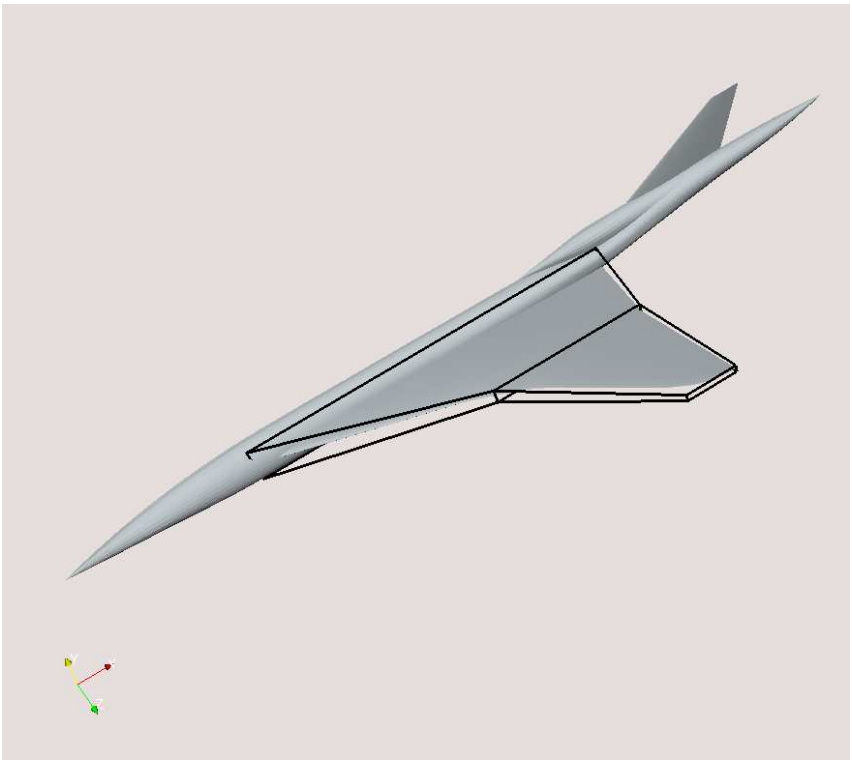


Figure 5: FFD box for supersonic business jet

Since in practice the volume of the wing has to be maintained for structural and other reasons, we impose a constraint on the volume in the cost function through a volume penalty term. The wings of supersonic aircrafts are very thin in order to reduce the wave drag; the optimization must not reduce the thickness of the wing since this affects its structural strength. Hence a penalty term which controls the thickness is added to the cost function. Finally the cost function that is used is given below

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right) + 10^3 \max(0, V_o - V) + I_p \tag{21}$$

where

- C_d = drag coefficient
- C_l = lift coefficient
- V = volume of the wing
- I_p = a penalty term to control the thickness

The quantities with subscript "o" indicate the values corresponding to the reference or starting shape. The penalty term I_p is computed as follows. A box is inserted inside the reference wing. When the wing grid is deformed, some points of the grid lying on the wing may go inside this box. The term I_p is computed as

$$I_p = 1000 \frac{\text{Number of grid points on wing surface lying inside the box}}{\text{Total number of grid points on the wing surface}} \tag{22}$$

This term approximately models the fraction of the wing surface that penetrates the inner box and thus penalizes the cost function if the wing thickness becomes too small. The CFD computations are performed on an unstructured grid with 37375 nodes and 184 249 tetrahedra using a finite volume scheme described in section (6).

7.1 FFD parameterization

The FFD parameterization is built only around the wing as shown in figure (5) 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 as closely as possible. The leading and trailing edges are kept fixed during the optimization by freezing the control points that correspond to $i = 0$ and $i = n_i$. The control points corresponding to $k = n_k$ which control the displacement of the wing tip are held fixed. Moreover, control points are only moved vertically. The parameterization corresponds to $n_i = 6$, $n_j = 1$ and $n_k = 2$ and leads to $(7-2) \times 2 \times 2 = 20$ degrees of freedom. The range of the control points is restricted to $[-500, +500]$ during the optimization.

7.2 Global metamodel

In order to study the effect of various parameters in the use of metamodels, we first construct global metamodels for lift and drag coefficients using RBF. The global metamodel will be used as the *exact model* for performing some of the tests. A set of 1000 points in $[-550, +550]^{20}$ is obtained using a Latin-Hypercube sampling [18]; however only 684 shapes have a valid grid since the remaining shapes lead to negative volumes during grid deformation. The metamodel is constructed using these 684 data points and the attenuation factor for the drag and lift coefficients are shown in table (4). We apply PSO to minimize the global metamodel using 60, 120 and 180 particles. The exact CFD solution is also evaluated on the predicted minimum point and the results are shown in table (5). The good agreement between the cost function predicted by the metamodel and actual cost function as given by CFD indicates that the global metamodel is itself satisfactory for the present optimization problem. This is not true in general since for a function of many variables that has a complex landscape, it is not easy to construct an accurate global metamodel. The difference in the minimum cost function using 120 and 180 particles is small indicating that 120 particles are sufficient in PSO for this problem.

Function	Range	Attenuation factor	Condition number
C_d	0.003939 - 0.011338	21203.64	5.64×10^{10}
C_l	-0.008715 - 0.044044	16727.44	1.78×10^{10}

Table 4: Global metamodel for drag and lift coefficients

Reference values: $C_l = 0.19542 \times 10^{-1}$ and $C_d = 0.410716 \times 10^{-2}$

Particles	Cost (MM)	$10C_l$ (CFD)	$100C_d$ (CFD)	Cost (CFD)
60	0.9350	0.195825	0.382842	0.9321
120	0.9227	0.195515	0.377299	0.9186
180	0.9214	0.199762	0.377383	0.9188

Table 5: PSO applied to global metamodel: The second column gives the optimum cost function obtained by applying PSO to the global metamodel and the remaining columns give the CFD solution for the optimum shape.

$\omega^0 = 1.2$	initial inertia
$h = 3$	inertia reduction criterion
$\alpha = 0.95$	inertia reduction rate
$c_1 = c_2 = 2$	trust coefficients
$p_c = 0.05$	craziness probability
$v^{\max} = (x_c^{\max} - x_c^{\min})/4$	maximum velocity

Table 6: Parameters used in PSO

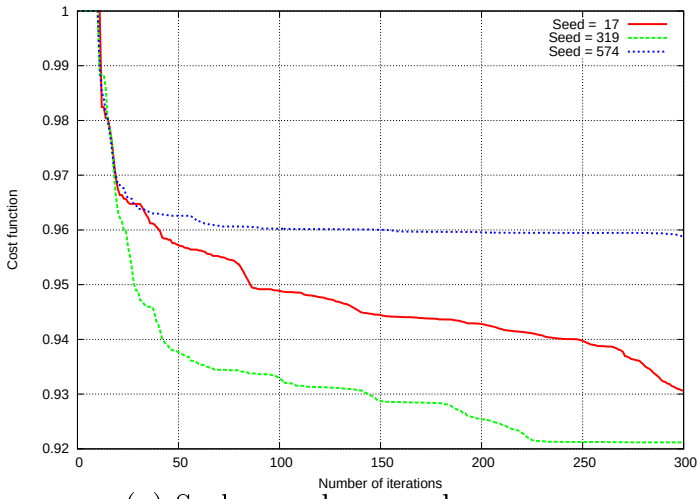
7.3 Optimization using global metamodel and PSO

PSO is applied to minimize the global metamodel; the parameters used in PSO are listed in table (6); more details on these parameters are available in [5].

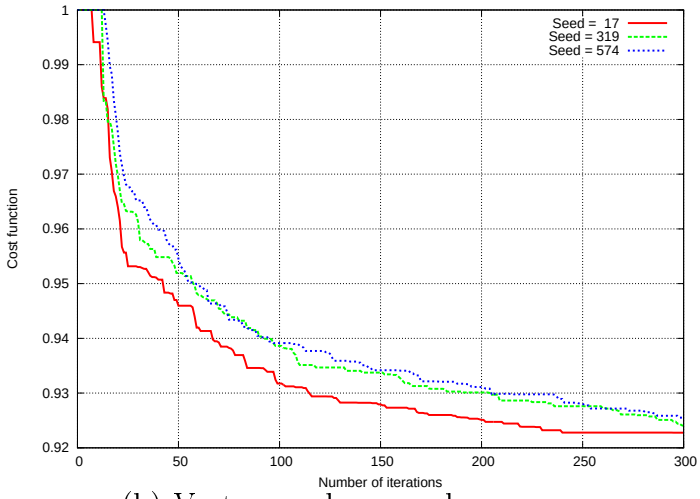
Effect of random numbers: Figures (6) show the convergence of cost function using the scalar and vector random numbers in the velocity update scheme for three different starting seeds. We see that the scalar scheme does not give consistent results with a wide scatter in the best achieved cost functions while the vector scheme is more consistent. In order to test the difference between the two schemes more rigorously, we perform a set of 50 optimization runs with different starting seeds and compute the statistics of the results. Table (7) gives the minimum and maximum of the achieved fitness, the average fitness and the standard deviation for the two schemes. The vector scheme achieves a smaller fitness and the spread of fitness values is also small, as indicated by the standard deviation. We clearly see that the vector scheme is more robust and consistent than the scalar scheme. In all subsequent tests we use the vector scheme in the velocity update of PSO.

Velocity scheme	Minimum cost	Maximum cost	Average cost	Standard deviation
Scalar	0.9137	0.9620	0.9368	0.00963
Vector	0.9191	0.9351	0.9269	0.00369

Table 7: Statistics of optimization using scalar and vector random number in the velocity update rule. 50 optimization runs are performed using 120 particles in PSO applied to the global metamodel



(a) Scalar random numbers r_1, r_2



(b) Vector random numbers r_1, r_2

Figure 6: Different velocity schemes: scalar and vector version of random numbers

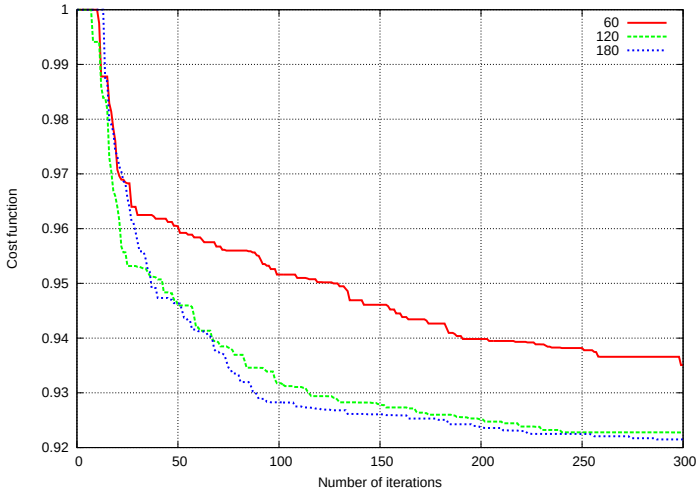


Figure 7: Effect of swarm size: PSO with 60, 120 and 180 particles used to minimize the global metamodel

Effect of number of particles: The ability of PSO in locating the global minimum depends on sufficient exploration of the design space especially for multi-modal functions as is common in engineering. This requires using a sufficiently large number of particles in the swarm. However this number should not be so large as to increase the computational cost. We apply PSO to minimize the global metamodel using 60, 120 and 180 particles and the results are shown in figure (7) and table (5), indicating that 120 particles are sufficient to locate the minimum for this problem. In all subsequent tests we use 120 particles in swarm.

7.4 Optimization using global metamodel, PSO and IPE

In order to test the effect of various parameters in IPE, we use the global metamodel as the *exact model*. The parameters in an IPE approach are:

- Type of metamodel, RBF, kriging, etc.
- Method of selecting the exact evaluations (pre-screening)
- Method of selecting the local database for constructing a local metamodel

In this work we use radial basis functions for constructing the metamodel. We first study the effect of the pre-screening criterion. As discussed in previous section, we study two different pre-screening methods, based on best particles and expected improvement in cost function. The data for constructing the local metamodel is selected based on proximity; the 40 closest points in the database are used. This is based on previous

	Iter	Seed = 17	Seed = 319	Seed = 574
100%	300	0.9227/36000	0.9240/36000	0.9255/36000
80%	300	0.9227/29040	0.9240/29040	0.9255/29040
50%	300	0.9257/18600	0.9242/18600	0.9283/18600
30%	400	0.9215/15240	0.9179/15240	0.9294/15240
20%	500	0.9184/12960	0.9246/12960	0.9165/12960
10%	500	0.9179/7080	0.9188/7080	0.9192/7080
Adap	500	0.9188/5717	0.9217/6385	0.9203/6428

Table 8: Effect of pre-screening criterion: The entries show the best cost function achieved and the number of exact function evaluations required.

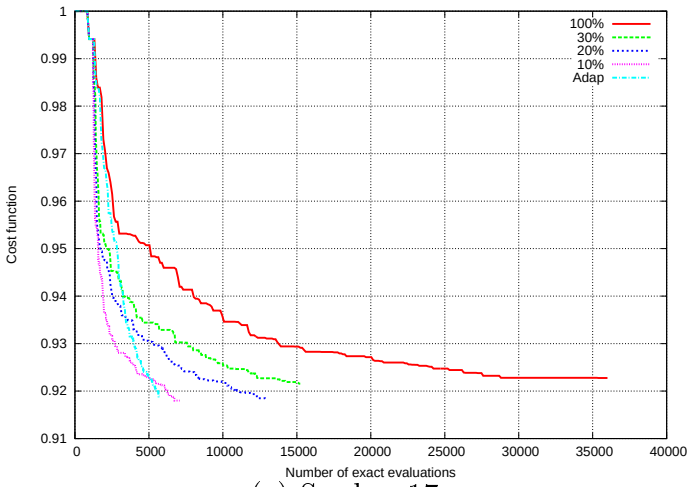
Exact	20	30	40	50	60
0.9227	0.9211	0.9234	0.9188	0.9229	0.9183

Table 9: Effect of size of local database: nearest neighbour

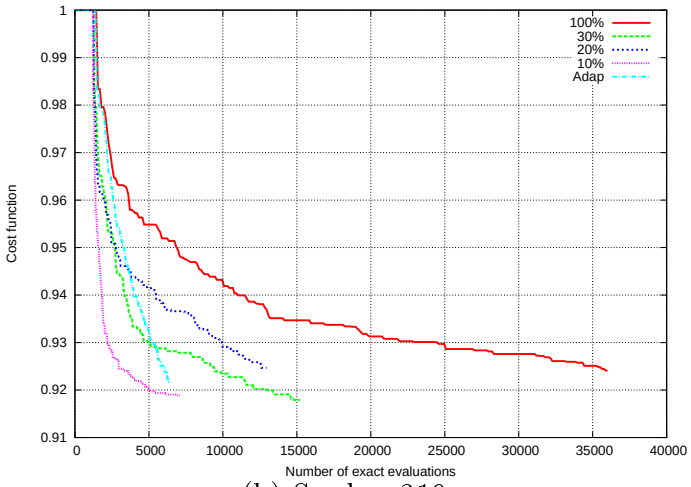
studies by Emmerich et al. [6] and our own studies discussed in the succeeding paragraphs. Figures (8) shows the evolution of the cost function for three different realizations, as a function of the number of exact evaluations and table (8) shows the best cost function achieved and the number of function evaluations required. We notice that metamodel-assisted PSO also leads to same level of cost functions as the case of exact evaluations. As the number of exact evaluations increases, we see that cost function achieved is equal to the 100% case. The case of 10% best particle and adaptive evaluations give good cost functions at a very low computational cost.

We next study the effect of the size of local database used for constructing the local models in the IPE phase. Since the dimension of the search space is 20, we test the optimization with 20, 30, 40, 50 and 60 points in the local database and table (9) shows the best cost function achieved. It is seen that the dependence on the size of local database is not very strong. Figure (9) shows the error of metamodel for different sizes of the database; it indicates that with 20 points the error is sometimes higher. A local database of 40 points gives good results both in terms of the error and the achieved cost function. This corresponds to twice the size of the search space dimension which is 20 in this problem.

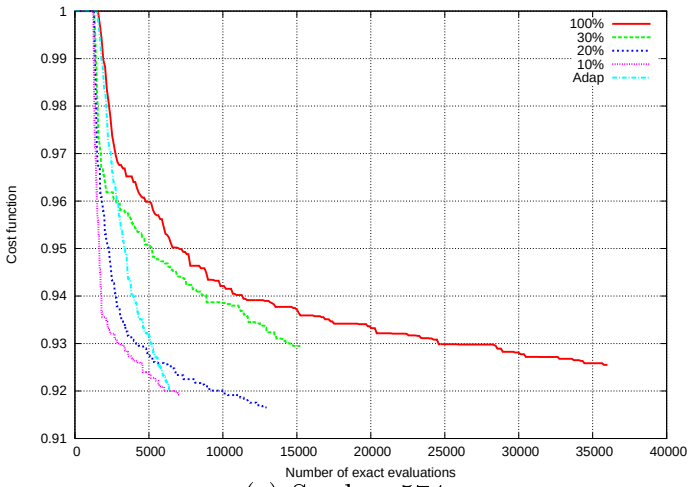
We next consider a variation in the selection of the local database. When the data points are chosen using only proximity criterion, it may not lead to a good stencil for constructing the metamodel. It also does not guarantee that all the components of design variables will have non-zero variations. If the points in the local database form a convex hull around the current evaluation point, it may lead to a better metamodel. However we do not try to select the points to satisfy the convex hull criterion but use a more simpler criterion which leads to similar result. If $x \in \mathbb{R}^d$ is the current evaluation point, we select atleast one point from the database so that the conditions $y_i^k < x_i$ and $y_i^s > x_i$ are satisfied for



(a) Seed = 17



(b) Seed = 319



(c) Seed = 574

Figure 8: Effect of pre-screening criterion. Three PSO runs are performed with different starting random seeds.

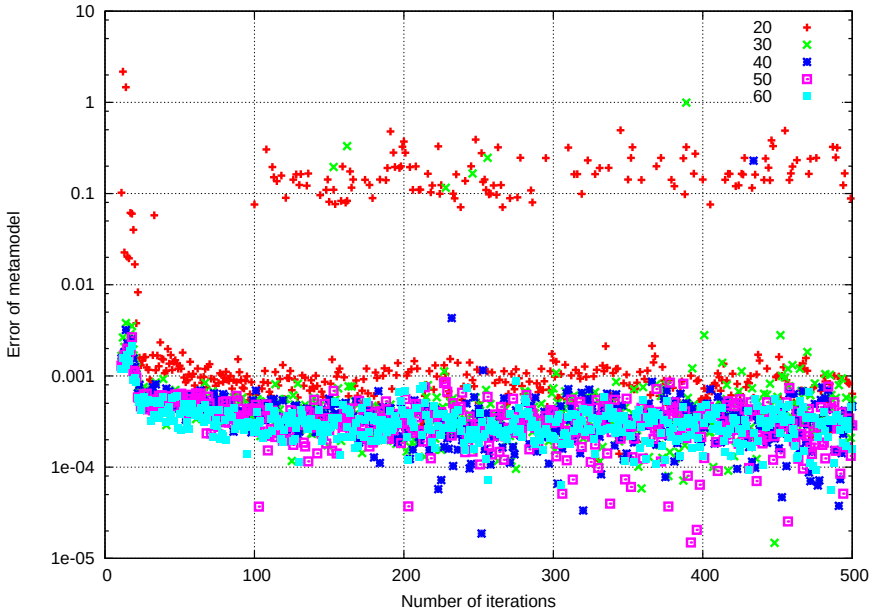


Figure 9: Relative error of metamodel for different size of local database: nearest neighbour

Exact	20	30	40	50	60
0.9227	-	-	0.9238	0.9218	0.9206

Table 10: Effect of size of local database: convex neighbour

every dimension i of the search space. Note that this requires atleast $2d$ points in the local database. Table (10) and figure (10) show the results of optimization using this type of database. The cost function achieved is comparable to the previous case as shown in table (9) and the error of the metamodel is also of the same magnitude as before. Atleast for this problem, the two methods of selecting the local database do not seem to have any significant effect on the results.

7.5 Optimization using CFD, PSO and IPE

The tests in the previous sections used a global metamodel as the *exact model*. We next perform the shape optimization using CFD as the exact model. The metamodel is used with 10%, 20%, 30% CFD evaluations and the adaptive pre-screening criteria. The local database is constructed with 40 nearest points from the database. When metamodels are used, more iterations are performed in PSO since the total number of exact evaluations is small. The results are given in table (11) and figure (11). First of all we notice that the cost function obtained after optimization are of the same order as those found with the global metamodel. This shows that in this case, the global metamodel itself was of very

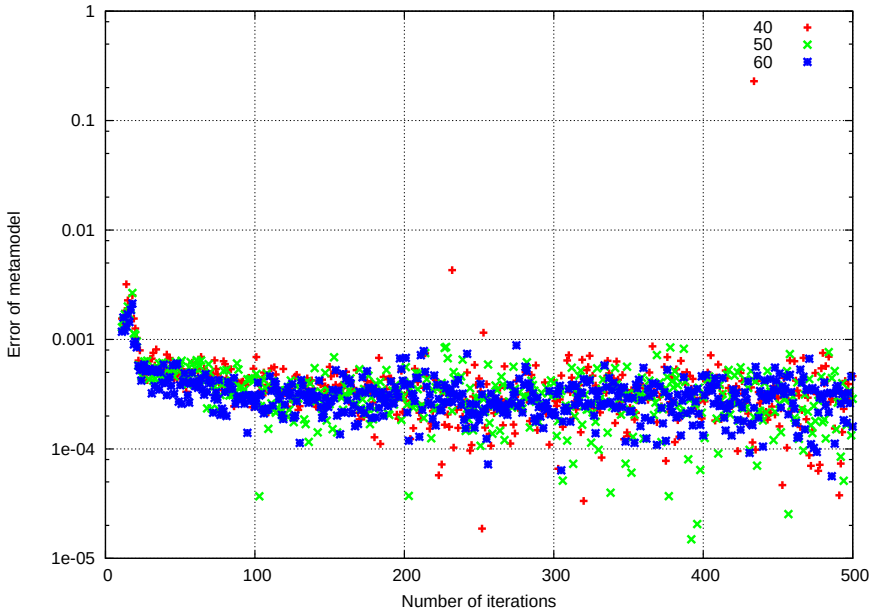
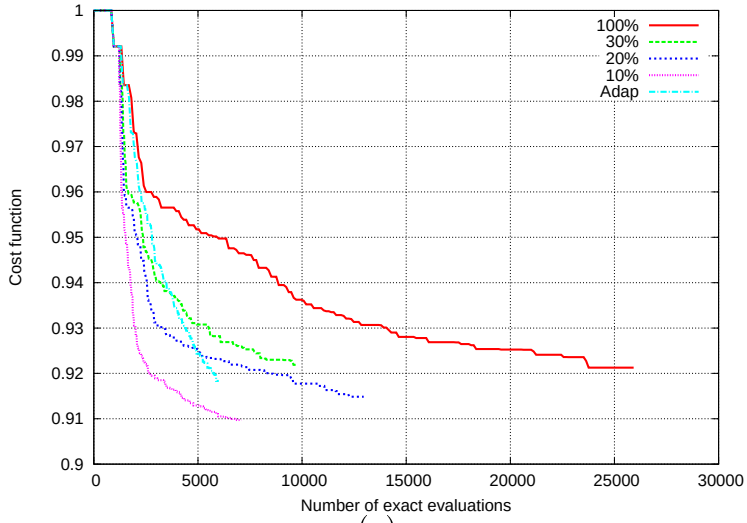


Figure 10: Relative error of metamodel for different size of local database: convex neighbour

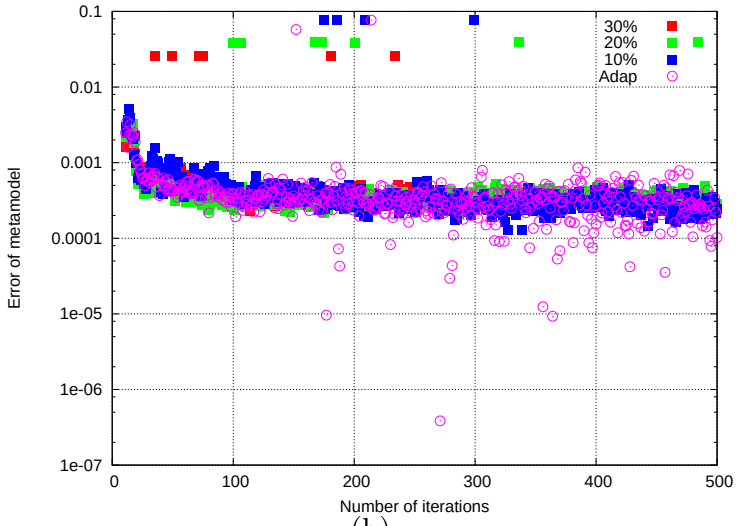
	Cost	CFD evaluations	$100C_d$	$10C_l$	Iter
Initial	1.0	-	0.410716	0.195429	-
100% CFD	0.9212	25920	0.378380	0.196545	216
30% CFD	0.9227	9480	0.378998	0.195732	248
20% CFD	0.9148	12960	0.375746	0.197580	500
10% CFD	0.9097	7080	0.373638	0.196345	500
Adaptive	0.9183	6002	0.377173	0.195799	500

Table 11: Results of PSO for supersonic business jet

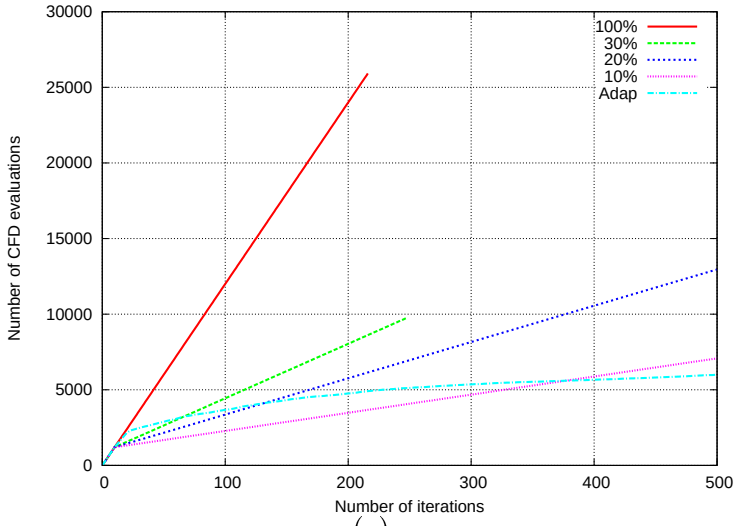
good accuracy. With the use of metamodel and IPE the same level of cost function as with full CFD evaluations, is obtained. Both the pre-screening criteria give similar level of cost functions but the 10% evaluations and adaptive criterion are most efficient. Figure (11) shows the evolution of the cost function as a function of the number of CFD evaluations. Figure (11-b) shows the average error of the metamodel while figure (11-b) shows the number of CFD evaluations as a function of the PSO iterations. These results are again comparable to those obtained with the global metamodel. Finally, figure (12) shows the shapes for initial configuration, CFD-optimized configuration and CFD+metamodel optimized configuration. It can be seen that the optimized shapes obtained using CFD alone and with metamodels are similar indicating that the use of metamodel leads to similar results.



(a)



(b)



(c)

Figure 11: Optimization of supersonic business jet: (a) Evolution of cost function, (b) Relative error of metamodel, and (c) Number of CFD evaluations

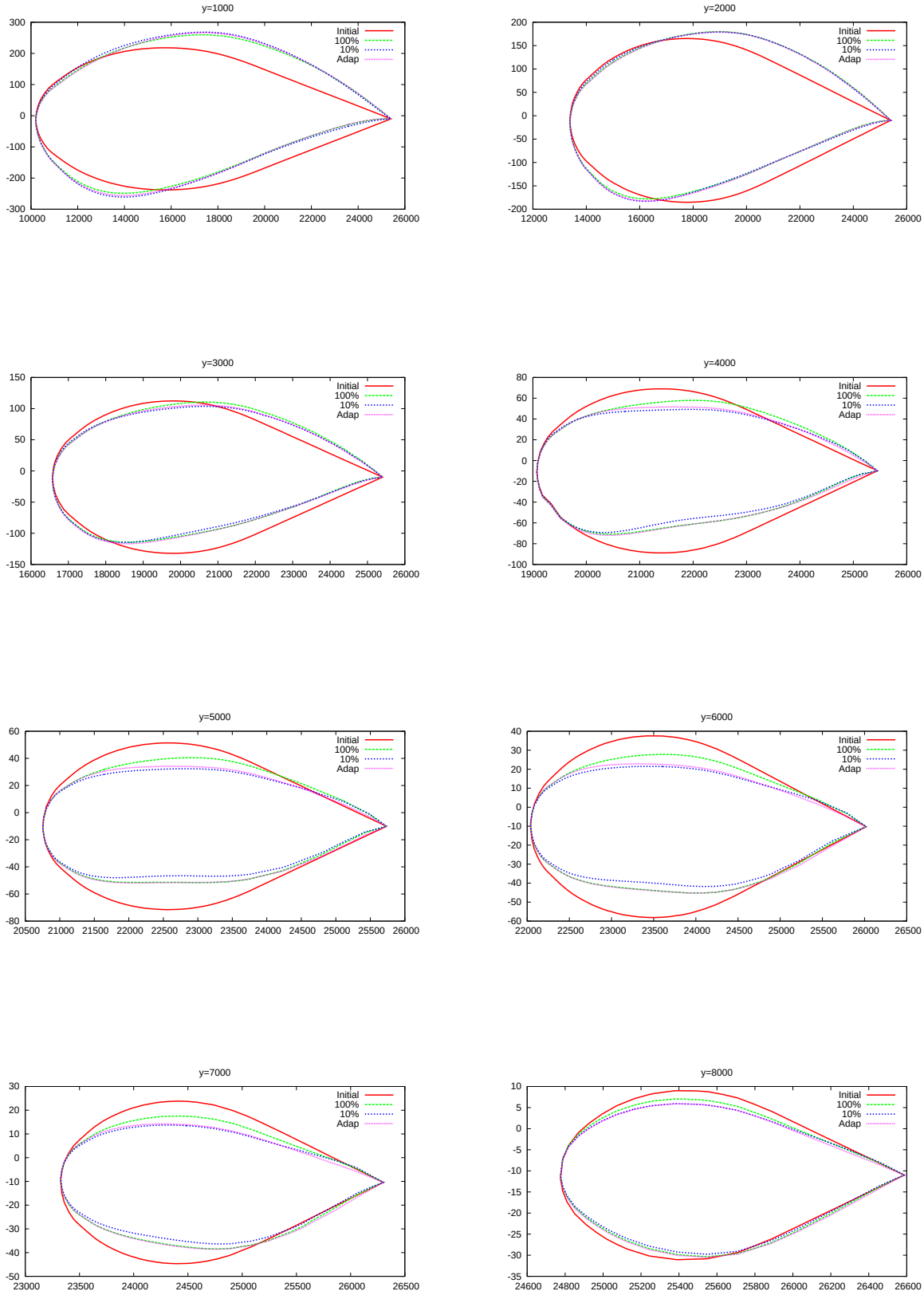


Figure 12: Wing shapes for supersonic business jet at different spanwise stations

8 Test case 2: 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 (13). 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.

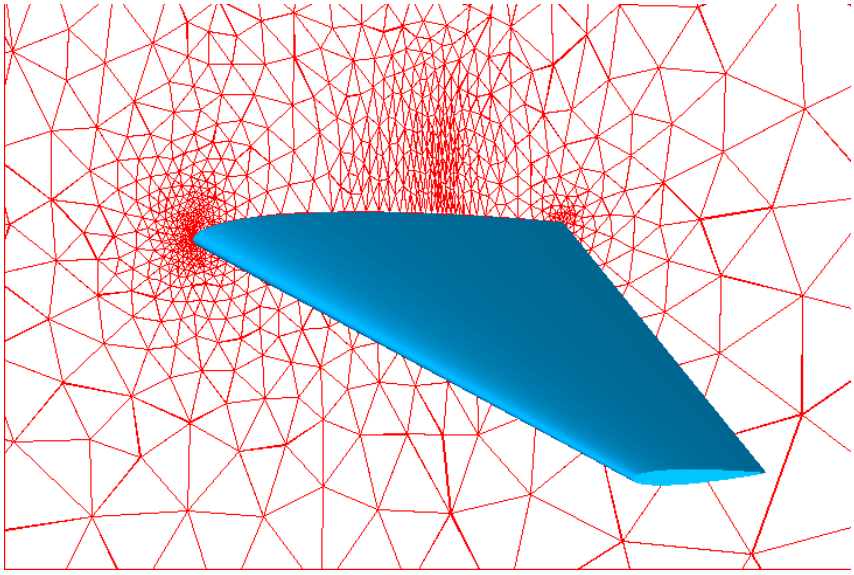


Figure 13: Initial wing shape (blue) and mesh in the symmetry plane (red).

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 decrease more than 0.1%. The constraint is taken into account using a penalization approach. Then, the resulting cost function is :

$$\mathcal{J} = \frac{C_d}{C_{do}} + 10^4 \max(0, 0.999 - \frac{C_l}{C_{lo}}) + 10^3 \max(0, V_o - V) \tag{23}$$

C_{do} and C_{lo} are respectively the drag and lift coefficients corresponding to the initial shape (NACA 0012 section) and V_o 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 (13)).

8.1 FFD parameterization

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,

	Cost	CFD evaluations	$10C_d$	C_l	Iter
Initial	1.0	-	0.263386	0.319024	-
100% CFD	0.4987	25800	0.131355	0.319350	215
10% CFD	0.4730	7080	0.124604	0.319020	500
Adaptive	0.5018	2511	0.132184	0.319987	500

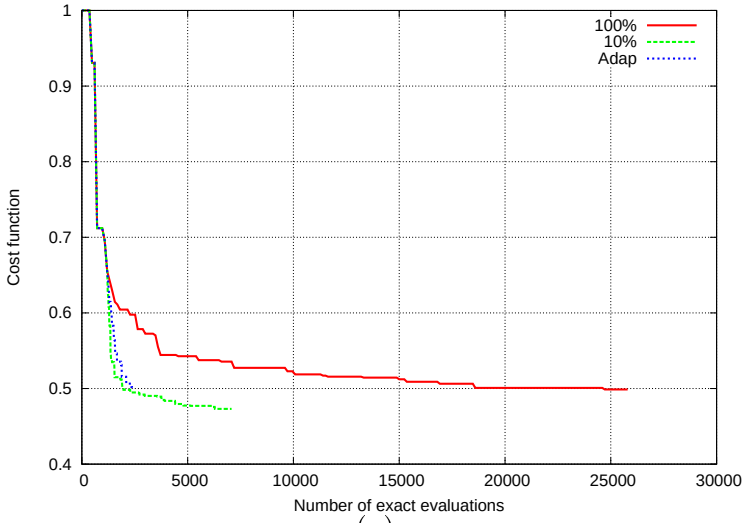
Table 12: Optimization of a transonic wing

control points are only moved vertically. The parameterization corresponds to $n_i = 6$, $n_j = 1$ and $n_k = 1$ and counts $(7-2) \times 2 \times 2 = 20$ degrees of freedom. The range of all the control points is restricted to $[-200, +200]$ during the optimization.

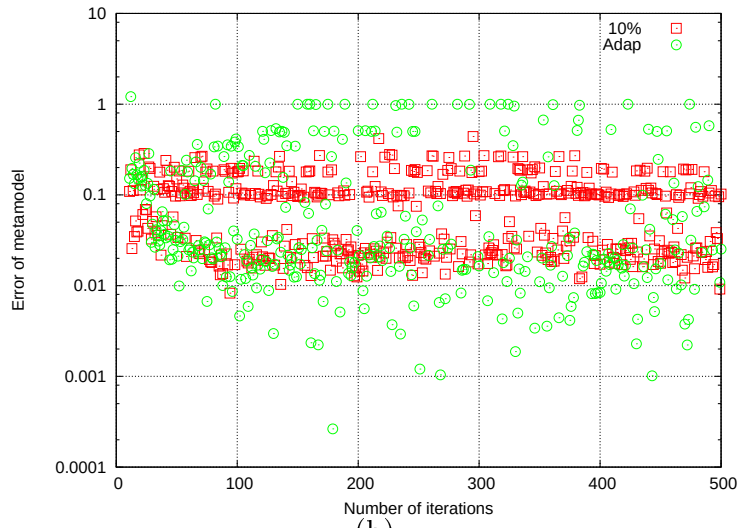
8.2 Optimization results

The optimization is performed using PSO with 120 particles and the same set of parameters as in section 7.3. The local metamodels are constructed using 40 nearest neighbours from the database. In the case of metamodel assisted PSO 500 iterations are performed. Table (12) shows the results of optimization. The metamodel assisted PSO is found to yield a cost function similar to the full CFD case while the number of CFD evaluations is significantly small. Figure (14-a) shows the evolution of the cost function with number of CFD evaluations while Figure (14-b) shows the growth of the average error of metamodel with the PSO iterations. Unlike the previous test case, we notice large errors in the metamodel in this test case. In most cases when the error is high, it is found that the maximum error is close to one. This is probably because of violation in lift constraint; the metamodel predicts that the lift constraint is satisfied but the CFD evaluation reveals that it is not. Since the lift penalty term is discontinuous there is a large error in the cost function. However the error in the metamodel does not degrade its ability to locate a good optimum solution.

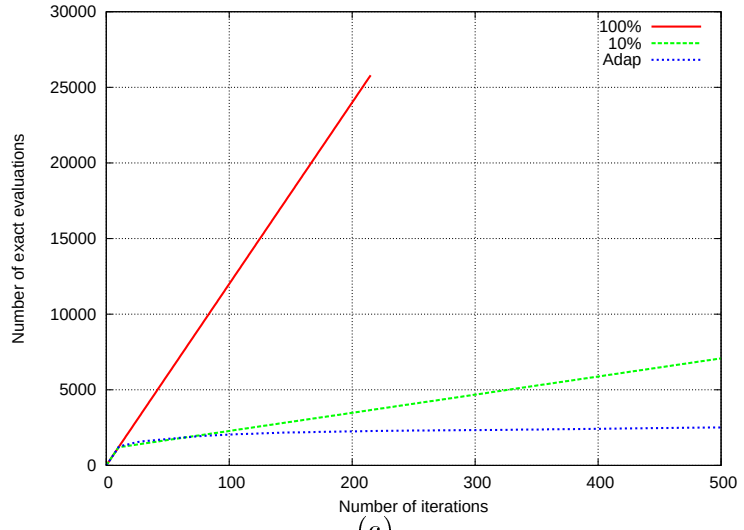
Figure (14-c) shows the variation of number of CFD evaluations with the iteration number. As in the case of SSBJ, the CFD evaluation count for the adaptive case grows very slowly and asymptotes to a nearly constant value indicating that the number of CFD evaluations goes to zero as the PSO iterations increase. Finally, figure (15) shows a comparison of the airfoil shapes at different spanwise locations. The shapes obtained with metamodel assisted PSO are quite close to those obtained with 100% CFD evaluations. Particularly, the shape of the upper surface is more critical since the shock is found on this side of the airfoil. We notice that metamodel-assisted optimization leads to very similar shapes on the upper surface.



(a)



(b)



(c)

Figure 14: Optimization of transonic wing

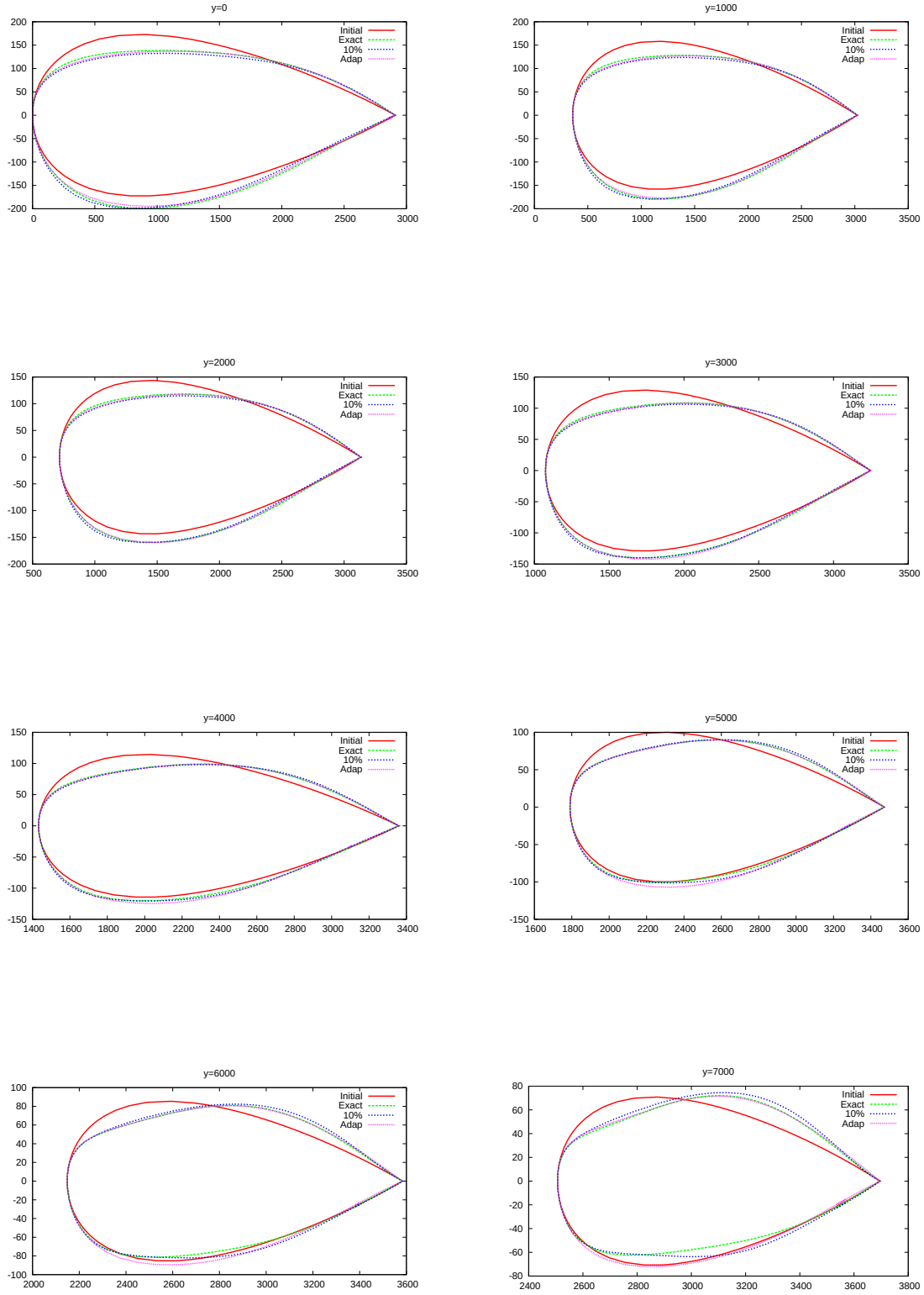


Figure 15: Wing shapes for transonic wing at different spanwise stations

9 Summary and conclusions

A particle swarm optimization algorithm combined with inexact pre-evaluation strategy is proposed. The novel idea is a pre-screening criterion specific to PSO; it is based on the predicted improvement in the local memory of individual particles after the IPE phase. No upper or lower limit is specified for the number of exact evaluations, which is allowed to be automatically determined by the screening criterion. The local metamodels are constructed using radial basis functions in which the shape parameter is optimized to yield accurate approximations. The new strategy is applied to solve two aerodynamic shape optimization problems. The proposed screening strategy is found to considerably reduce the number of required CFD computations while yielding optimal shapes comparable to the full exact CFD case.

Between the two pre-screening criteria tested in this work, no definite conclusion as to the superiority of either one can be made, though both of them yield acceptable solutions at highly reduced computational cost. The best particles criterion (using 10% exact evaluations) seems to be capable of yielding slightly better solutions due to greater exploration of the search space. The proposed strategy is very promising and must be applied to more test cases to demonstrate its robustness. Other metamodels like kriging which provide an estimate of variance can be utilized in the present strategy, allowing the use of other pre-screening criteria.

10 Acknowledgments

The authors gratefully acknowledge the scientific committee of IDRIS (Project 72906) and CINES (Project SOP2703) for the attribution of CPU time. This study has been supported by the "OMD" project (Multi-Disciplinary Optimization) granted by ANR-RNTL.

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] T. Chen and R. Chen. Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. *IEEE Transactions on Neural Networks*, 6(5), 1995.

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

[5] 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.

[6] 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.

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

[8] B. Fornberg and G. Wright. Stable computation of multi-quadric interpolants for all values of the shape parameter. *Computers and Mathematics with Applications*, 48:853–867, 2004.

[9] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Prog. Aero. Sci.*, 38:43–76, 2002.

[10] K. C. Giannakoglou, A. P. Giotis, and M. K. Karakasis. Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters. *J. of Inverse Prob. in Engg.*, 9(4):389–412, 2001.

[11] K. C. Giannakoglou and M. K. Karakasis. Hierarchical and distributed metamodel-assisted evolutionary algorithms. In *VKI Lecture Series 2006-03.* March 2006.

[12] K. C. Giannakoglou, D. I. Papadimitriou, and I. C. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-enhanced metamodels. *Comput. Methods Appl. Mech. Engrg.*, 195:6312–6329, 2006.

[13] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.*, 76:1905–1915, 1971.

[14] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1), 2005.

[15] A. J. Keane and P. B. Nair. *Computational Approaches for Aerospace Design.* Wiley, 2005.

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

[17] E. Larsson and B. Fornberg. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Computers and Mathematics with Applications*, 49:103–130, 2005.

[18] M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

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

[20] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing. Springer Verlag, 2004.

[21] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Adv. Comp. Math.*, 11, 1999.

[22] J. Samareh. A survey of shape parameterization techniques for high fidelity multidisciplinary shape optimization. *AIAA Journal*, 39(5):877–884, 2001.

[23] R. Schaback. Reconstruction of multivariate functions from scattered data. available on the internet at `http://www.num.math.uni-goettingen.de/schaback/teaching/texte/rbfbook.ps`.

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

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

[26] Z. M. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Num. Anal.*, 13(1):13–27, 1993.

[27] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Tran. on Systems, Man, and Cybernetics - Part C: Applications and Reviews.*

Contents

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