Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design

Praveen Chandrashekarappa, Regis Duvigneau

Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design

Praveen. C and R. Duvigneau *
INRIA Project-Team OPALE, 2004 Rte des Lucioles - B.P. 93
06902 Sophia Antipolis, France

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 computational cost. This is especially so in view of the increasing use of costly high fidelity analysis tools like Computational Fluid Dynamics (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

1. Introduction

Modern optimization methods like genetic algorithms [17] and particle swarm optimization [22] have been found to be capable of solving practical optimization problems. Among their many advantages are their ability to handle non-smooth functions (since

* Corresponding author
Email addresses: praveen@cfdlab.net (Praveen. C), Regis.Duvigneau@inria.fr (R. Duvigneau).

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* or *metamodels* [2,9,6,13] in place of the costly evaluation tool. These surrogate models are inexpensive compared to the exact model. The most commonly used type of metamodels are data-fitting models which construct an approximation to the cost function using a set of 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) [13], artificial neural networks (like multi-layer perceptron, radial basis function networks) [13,6] and Gaussian process models (kriging) [2]. 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.

Data fitting models have been extensively used for optimization of costly functions [13]. 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.

In this work we consider the use of metamodels to reduce the computational cost and hence accelerate global optimization methods like GA and PSO. 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. [25] 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 *inexact pre-evaluation* (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 [14], the gradients being efficiently computed using adjoint methods. Giannakoglou et al. [11] 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, and the development of adjoint solvers is a difficult task.

In this work we use 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 [5]. Briefly, a genetic algorithm can be described as follows:

(i) Evaluate the fitness of all individuals in the population
(ii) Apply the selection operator to eliminate non-promising individuals
(iii) Apply the crossover operator to generate offsprings from the selected individuals
(iv) 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 was introduced by Kennedy and Eberhart [15] as a simplified social model. It mimics the behaviour of bird flocking and is based on rules that enable sudden direction changes, scattering, regrouping, etc. These moves are motivated in nature by food seeking or predator avoiding, and can be implemented in a simple algorithm for global optimization. PSO also requires

a large number of function evaluations since it requires a large number of particles to effectively locate the optimum. Hence we propose a metamodel-assisted PSO in which *local RBF approximations are used to pre-evaluate the particles*. Then a small percentage of particles are selected (pre-screening) for exact evaluations. We also propose a new pre-screening criterion which is specific to PSO and determines automatically the number of exact evaluations. 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.

The rest of the paper is organised as follows. Section (2) contains a brief description of RBF approximations and some practical issues in its implementation. Section (3) describes the basic PSO algorithm, and its modification to incorporate metamodels and inexact pre-evaluation is explained in section (4). The developed algorithms are applied to 3-D aerodynamic shape optimization governed by Euler equations. Section (5.1) describes the parameterization of the shape deformations using free-form deformation (FFD). A brief description of the numerical scheme for solving the fluid equations is then given in section (5.2). Sections (6) and (7) describe the application of the algorithm to two aerodynamic shape design problems. Finally the paper ends with a summary and conclusions.

2. Radial basis function models

Radial basis function approximations were introduced by Hardy [12] to represent topographical surfaces given sets of sparse scattered measurements. They have been found to be very accurate for interpolation of arbitrarily scattered data [19]. 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. 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 $\hat{f}(x_m) = f_m$, $m = 1, 2, \ldots, N$, where $\{x_m, f_m\}$, $i = 1, ..., N$ is the given dataset. The above equations can be written in matrix form as $Aw = F$ 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. Examples of radial basis functions are the gaussian $\phi(r) = \exp(-r^2/a^2)$ and the inverse multi-quadric $\phi(r) = (r^2 + a^2)^{s/2}$, $s < 0$. For these functions, 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, and such functions are said to be *unconditionally positive definite*. When the RBF and the approximated function are infinitely smooth, the error in the approximation decreases exponentially as the number of data points is increased [24].

2.1. *Selection of attenuation factor*

Most radial basis functions have a parameter that has be selected by the user. In the case of the gaussian RBF, $\phi(r) = \exp(-r^2/a^2)$, the *attenuation factor* "a" determines

how fast the RBF decays away from its center and has to be chosen by the user. The attenuation factor in radial basis functions has a critical influence on the accuracy of the interpolation model.

In [18], several empirical methods for choosing the attenuation factor are discussed and it is concluded 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. 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 selected 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 $E(a) = [E_1, E_2, \ldots, E_N]^\top$. Rippa [18] 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.

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.

2.2. *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. 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 [24] 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. While minimizing the cost function $C(a)$ we compute the condition number of the coefficient matrix A; if it is larger than a 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. 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 [4]. The algorithm is given for a function minimization problem

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

Algorithm: *Particle swarm optimization*

(i) Set $n = 0$

(ii) Randomly initialize the position of the particles and their velocities $\{x_k^n, v_k^n\}, k = 1, \ldots, K$.

(iii) Compute cost function associated with the particle positions $\mathcal{J}(x_k^n), k = 1, \ldots, K$

(iv) Update the local and global memory

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

(v) Update the particle velocities

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

(vi) Apply craziness operator to the velocities

(vii) Update the position of the particles (we take the time step to be one)

$$x_k^{n+1} = x_k^n + v_k^{n+1} \tag{3}$$

(viii) Limit new particle positions to lie within $[x_l, x_u]$ using reflection at the boundaries

(ix) 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 [8]. 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 [8] 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 [21] in order to improve the stability and convergence rate of PSO is also used.

In the original algorithm proposed by Kennedy and Eberhart [15] 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 [23] 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 present work $N_e = 10$ is used. 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*

(i) Set $n = 0$

(ii) Randomly initialize the position of the particles and their velocities $\{x_k^n, v_k^n\}, k = 1, \ldots, K$.

(iii) 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 $\hat{\mathcal{J}}(x_k^n), k = 1, \ldots, K$.

(iv) 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.

(v) Update the local and global memory *using only the exactly evaluated cost functions*

$$x_{*,k}^n = \underset{0 \leq s \leq n}{\operatorname{argmin}} \mathcal{J}(x_k^s), \quad x_*^n = \underset{0 \leq s \leq n, 1 \leq k \leq K}{\operatorname{argmin}} \mathcal{J}(x_k^s) \tag{4}$$

(vi) Store exactly evaluated function values into a database

(vii) Update the particle velocities

$$v_k^{n+1} = \omega^n v_k^n + c_1 r_{1,k}^n (x_{*,k}^n - x_k^n) + c_2 r_{2,k}^n (x_*^n - x_k^n) \tag{5}$$

(viii) Apply craziness operator to the velocities

(ix) Update the position of the particles

$$x_k^{n+1} = x_k^n + v_k^{n+1} \tag{6}$$

(x) Limit new particle positions to lie within $[x_l, x_u]$ using reflection at the boundaries

(xi) If $n < N_{\max}$, then $n = n + 1$ and go to step (iii), 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 $\mathcal{J}_{\min}$ be the current minimum function value and $\hat{\mathcal{J}}(x)$ 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

$$I(x) = \begin{cases} 0 & \text{if } \hat{\mathcal{J}}(x) > \mathcal{J}_{\min} \\ \mathcal{J}_{\min} - \hat{\mathcal{J}}(x) & \text{otherwise} \end{cases} \tag{7}$$

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 local memory value is predicted to reduce in the IPE phase, i.e.,

$$S^n = \{k : \hat{\mathcal{J}}(x_k^n) < \mathcal{J}(x_{*,k}^{n-1})\} \tag{8}$$

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 the percentage of particles that are exactly evaluated; *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. Shape optimization framework

5.1. *Parameterization using the Free-Form Deformation approach*

The FFD technique originates from the Computer Graphics field [20]. 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 (1) 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:

$$\Delta q = \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} B_i^{n_i}(\xi_q) B_j^{n_j}(\eta_q) B_k^{n_k}(\zeta_q) \Delta P_{ijk}. \tag{9}$$

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

$$B_p^n(t) = C_n^p \, t^p \, (1-t)^{n-p}. \tag{10}$$

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

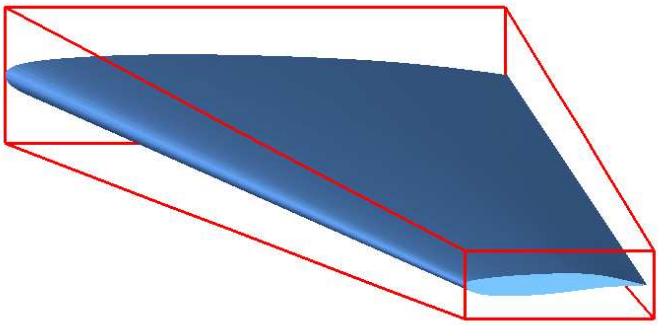


Fig. 1. Example of FFD lattice (red) around a wing.

5.2. *Aerodynamic fitness evaluation using CFD*

The evaluation of aerodynamic performance is made using inviscid flow model. The Euler equations are solved in three dimensions on an unstructured tetrahedral grid using an upwind, finite volume method. The finite volumes are constructed around each vertex of the tetrahedral grid by joining the centroid of the tetrahedra with the midpoint of the edges. The approximate Riemann solver of Roe is used for computing the numerical flux function. Second order accuracy is achieved by linearly interpolating the physical variables from the vertices to the mid-point of the edges in the grid. In order to avoid spurious oscillations of the the solution in the vicinity of the shock, a slope limiter is used. The resulting system of equations is solved using an implicit time integration scheme with local time-stepping. More details about the numerical scheme can be found in [3].

6. 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. 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

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right) + 10^3 \max(0, V_o - V) + I_p \qquad (11)$$

where C_d = drag coefficient, C_l = lift coefficient, V = volume of the wing and I_p = a penalty term to control the thickness. The quantities with subscript "o" indicate the

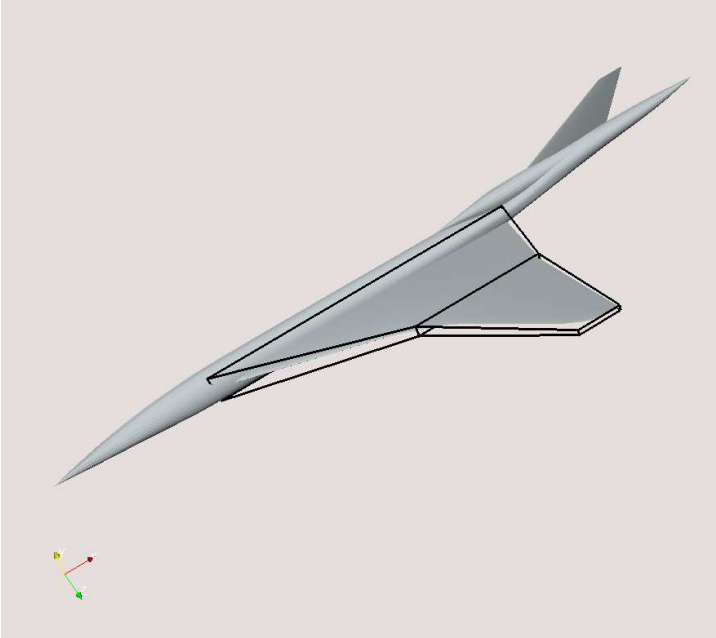


Fig. 2. FFD box for supersonic business jet

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

$$I_p = 1000 \frac{\text{Number of grid points on wing surface lying inside the box}}{\text{Total number of grid points on the wing surface}} \tag{12}$$

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 (5.2).

6.1. *FFD parameterization*

The FFD parameterization is built only around the wing as shown in figure (2) 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.

Particles	60	120	180
Cost (MM)	0.9350	0.9227	0.9214
Cost (CFD)	0.9321	0.9186	0.9188

Table 1
PSO applied to global metamodel; the exact cost function from CFD for optimal shape determined from minimizing the metamodel is also given.

$\omega^0 = 1.2$	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 2
Parameters used in PSO

6.2. *Global metamodel validation*

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 is obtained using a Latin-Hypercube sampling [16]; however only 684 shapes have a valid grid since the remaining shapes lead to negative volumes during grid deformation and the metamodel is constructed using these data points. 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 (1). 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.

6.3. *Optimization using global metamodel and PSO*

PSO is applied to minimize the global metamodel in order to study the effect of various parameters in the use of metamodels. The parameters used in PSO are listed in table (2); more details on these parameters are available in [4].
Effect of random numbers: Figures (3) show the convergence of cost function using the scalar and vector random numbers in the velocity update scheme for three different starting random number 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 (3) gives the minimum and maximum of the achieved fitness, the average fitness and the standard deviation for the two schemes. The vector

Velocity scheme	Min. cost	Max. cost	Avg. cost	Std. dev.
Scalar	0.9137	0.9620	0.9368	0.00963
Vector	0.9191	0.9351	0.9269	0.00369

Table 3
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

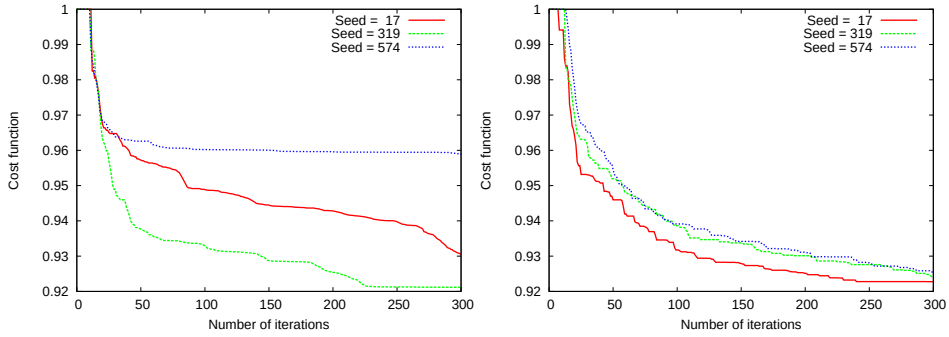


Fig. 3. Different velocity schemes: scalar and vector version of random numbers r_1, r_2 in PSO

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.

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 (4) and table (1), indicating that 120 particles are sufficient to locate the minimum for this problem. In all subsequent tests we use 120 particles in the swarm.

6.4. *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*. 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 section (4), 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 studies by Emmerich et al. [6] and our own studies discussed in the succeeding paragraphs. Table (4) shows the best cost function achieved and the number of function evaluations required for different pre-screening criteria. We notice that metamodel-assisted PSO also leads to same level of cost functions as the case of exact evaluations.

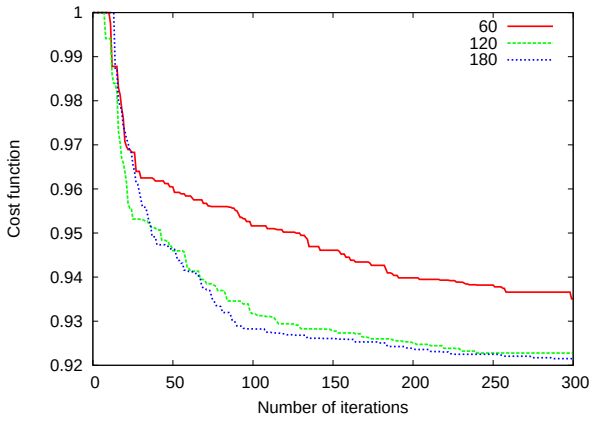


Fig. 4. Effect of swarm size: PSO with 60, 120 and 180 particles used to minimize the global metamodel

	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 4
Effect of pre-screening criterion: The entries show the best cost function achieved and the number of exact function evaluations required.

Local DB type	Exact	20	30	40	50	60
Nearest neighbour	0.9227	0.9211	0.9234	0.9188	0.9229	0.9183
Convex neighbour	0.9227	-	-	0.9238	0.9218	0.9206

Table 5
Effect of size of local database

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 (5) shows the best cost function achieved. It is seen that the dependence on the size of local database is not very strong. Figure (5-a) 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

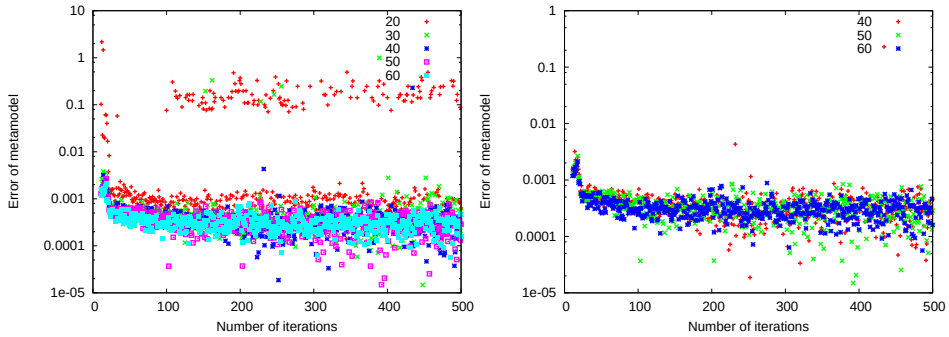


Fig. 5. Relative error of metamodel for different size of local database: (a) nearest neighbour (b) convex neighbour

Random no.	Seed = 17		Seed = 319		Seed = 574	
IPE	10%	Adap	10%	Adap	10%	Adap
Scalar	0.9418	0.9439	0.9375	0.9232	0.9406	0.9502
Vector	0.9179	0.9188	0.9188	0.9217	0.9192	0.9203

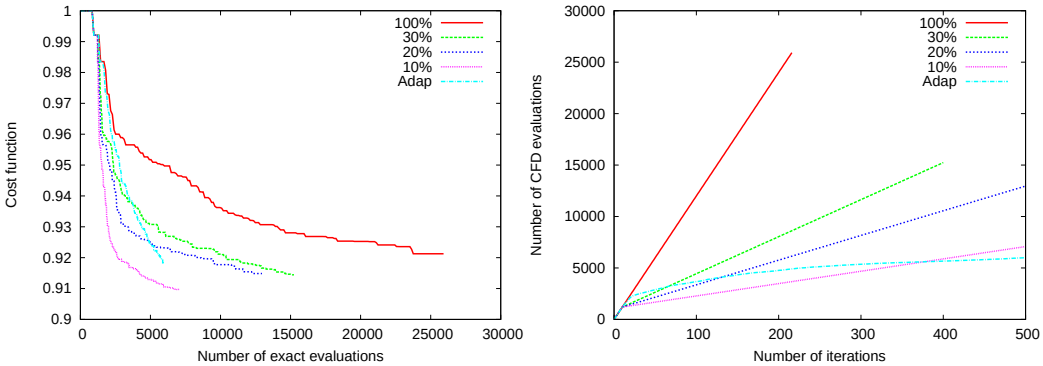
Table 6
Effect of random numbers on metamodel-assisted PSO

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 in the local dataset. 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 every dimension i of the search space. Note that this requires atleast $2d$ points in the local database. Table (5) and figure (5-b) show the results of optimization using this type of database. The cost function achieved is comparable to the previous case as shown in table (5) 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.

In all the above tests, the vector version of random numbers was used in the velocity update since this was found to be more robust. We next give some examples to show that this is important for the metamodel-assisted PSO also. We perform metamodel-assisted optimization using 10% and adaptive exact evaluations and the results are shown in table (6). It is clear that the scalar version does not perform as well as the vector version. As discussed before the robustness of the vector scheme arises from its ability to avoid degenerate line searches and space filling characteristics, the last of which is also important for metamodel-assisted optimization since it leads to better local metamodels.

Method	100% CFD	30% CFD	20% CFD	10% CFD	Adaptive
Cost	0.9212	0.9144	0.9148	0.9097	0.9183
Iterations	216	400	500	500	500
CFD eval.	25920	15240	12960	7080	6002

Table 7
Results of PSO for supersonic business jet



(a) (b)

Fig. 6. Optimization of supersonic business jet: (a) Evolution of cost function, (b) Number of CFD evaluations

6.5. *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 (7) and figure (6). 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 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 (6-a) shows the evolution of the cost function as a function of the number of CFD evaluations while figure (6-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 (7) 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.

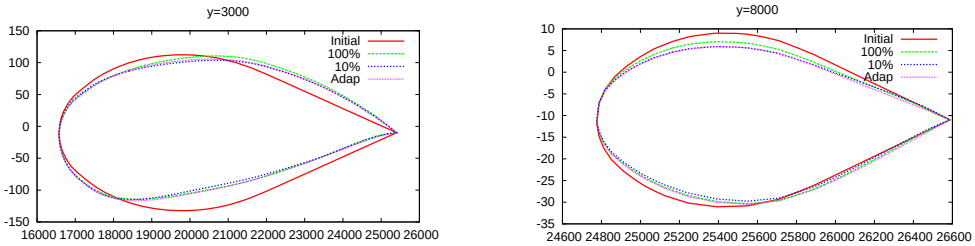


Fig. 7. Wing shapes for supersonic business jet at different spanwise stations

7. 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 (8). The free-stream Mach number is $M_\infty = 0.83$ and the incidence $\alpha = 2°$. Initially, the wing section corresponds to the NACA 0012 airfoil.

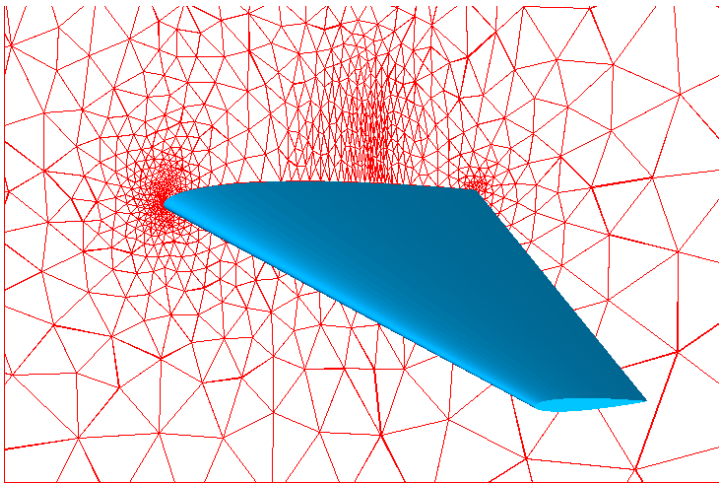


Fig. 8. Transonic wing: Initial wing shape (blue) and mesh in the symmetry plane (red).

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 :

$$\mathcal{J} = \frac{C_d}{C_{do}} + 10^4 \max(0, 0.999 - \frac{C_l}{C_{lo}}) + 10^3 \max(0, V_o - V) \tag{13}$$

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 (8)).

Method	100% CFD	10% CFD	Adaptive
Cost	0.4987	0.4730	0.5018
Iterations	215	500	500
CFD eval.	25800	7080	2511

Table 8
Optimization of a transonic wing

7.1. *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 1). 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, 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.

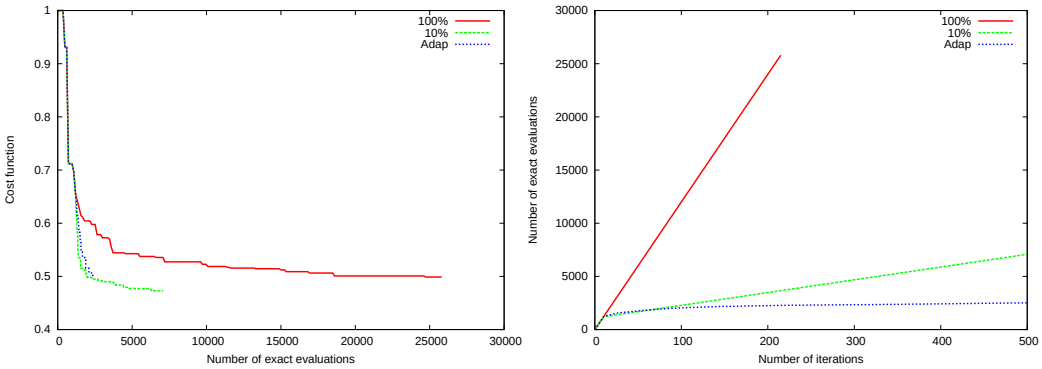
7.2. *Optimization results*

The optimization is performed using PSO with 120 particles and the same set of parameters as in section 6.3. The local metamodels are constructed using 40 nearest neighbours from the database. In the case of metamodel assisted PSO, 500 iterations are performed. Table (8) 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 (9-a) shows the evolution of the cost function with number of CFD evaluations. Both the pre-screening criteria are able to yield reductions in cost function comparable to the exact evaluations case. The 10% exact evaluations case finds a lower cost function than the adaptive case or even the 100% exact case because we are able to perform more PSO iterations.

Figure (9-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 (10) 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.

8. 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



(a) (b)

Fig. 9. Optimization of transonic wing: (a) Evolution of cost function, (b) Number of CFD evaluations

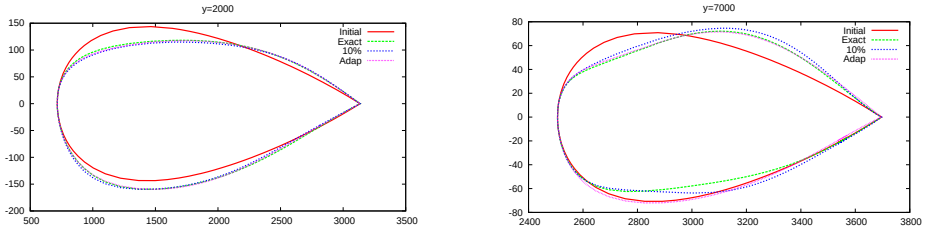


Fig. 10. Wing shapes for transonic wing at different spanwise stations

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. The present work clearly demonstrates the potential of improving the efficiency of PSO using metamodels.

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 main advantage of the proposed adaptive screening criterion for PSO is that the user does not have to make any choice regarding the number of exact evaluations.

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

[4] 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.

[5] R. Duvigneau and C. Praveen. Meta-modeling for robust design and multi-level optimization. 42nd AAAF Congress on Applied Aerodynamics, Sophia Antipolis, March 2007.

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

[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, 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.

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

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

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

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

[16] 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.

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

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

[19] 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`.

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

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

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

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

[24] 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.

[25] 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*.