Multilevel Modeling

Praveen Chandrashekarappa, Régis Duvigneau

Multilevel Modeling

0.1. Principles

Modern semi-stochastic optimization methods like genetic algorithms [MIC 92] and particle swarm optimization [VEN 03] have been found to be capable of solving practical optimization 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* or *meta-models* [B¨ 05, GIA 02, EMM 06, JIN 05, ONG 04] 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

---

Chapter written by Praveen CHANDRASHEKARAPPA and Régis DUVIGNEAU.

method. Examples of data-fitting models are polynomials (usually quadratic, also known as response surface models) [JIN 05], artificial neural networks (like multi-layer perceptron, radial basis function networks) [JIN 05, EMM 06] and Gaussian process models (kriging) [B¨ 05]. 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 as preconditioners to accelerate the exploration of the search space.

– 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.

In the following sections, we consider data-fitting models, particularly radial basis functions and gaussian random process models, also known as kriging. Both these methods have been found to be effective in interpolation of high dimensional data with small number of data points as compared to polynomial based methods. Data fitting models have been extensively used for optimization of costly functions [JIN 05]. 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.

Giannakoglou [GIA 02] 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 [KEN 95] 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.

0.2. Particle swarm optimization

PSO is modeled on the behaviour of a swarm of animals when they hunt for food or avoid predators [MIL 07]. 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 [DUV 06]. The algorithm is given for a function minimization problem

$$\min_{x_l \leq x \leq x_u} J(x)$$

Algorithm: *Particle swarm optimization*

1. Set $n = 0$
2. Randomly initialize the positions and velocities $\{x_k^n, v_k^n\}, k = 1, \ldots, K$.
3. Compute cost function values $J(x_k^n), k = 1, \ldots, K$
4. Update the local and global memory

$$x_{*,k}^n = argmin_{0 \leq s \leq n} J(x_k^s), \quad x_*^n = argmin_{0 \leq s \leq n, 1 \leq k \leq K} J(x_k^s) \tag{1}$$

5. Update the particle velocities

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

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

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

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

Apart from the above basic algorithm, we use several additional strategies to enhance the efficiency of PSO. The inertia parameter ω is decreased during the iterations as proposed by Fourie and Groenwold [FOU 02]. 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:

$$\text{If } J(x_*^n) = J(x_*^{n-h}) \text{ then } \omega^n = \alpha \omega^{n-1}$$

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 [FOU 02] 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 [MIC 92]. Finally, an upper limit on velocity as recommended by Shi and Eberhart [SHI 98] in order to improve the stability and convergence rate of PSO is also used.

In the original algorithm proposed by Kennedy and Eberhart [KEN 95] 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 [WIL 05] 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.

0.3. 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*

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

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

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

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

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

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

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

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 [EMM 06] 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 $J_{\min}$ be the current minimum function value and $\hat{J}(x)$ 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

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

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 [EMM 06] 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.,

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

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*.

0.4. Test-case 1: Supersonic Business Jet Optimization

0.4.1. *Test-case description*

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

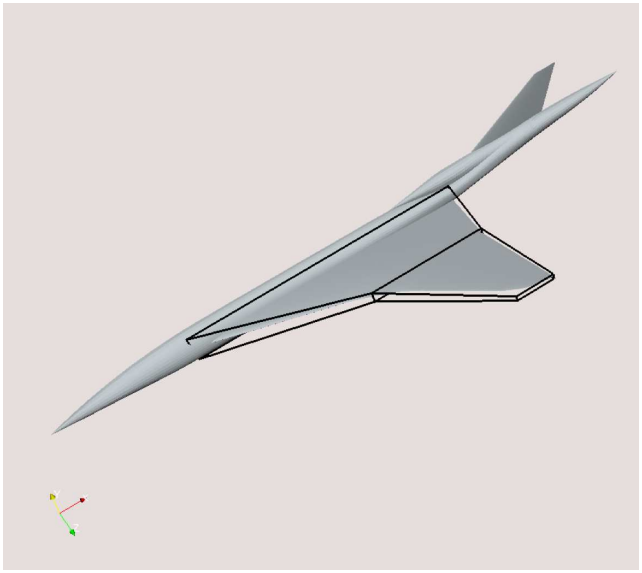


Figure 1. *FFD box for supersonic business jet*

$$J = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right) + 10^3 \max(0, V_o - V) + I_p \qquad (9)$$

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 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

$$I_p = 1000 \frac{\text{Number of grid points on wing surface lying inside the box}}{\text{Total number of grid points on the wing surface}} \qquad (10)$$

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 solver developed at INRIA and described in [DER 92].

0.4.2. *FFD parameterization*

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. The Free-Form Deformation (FFD) technique [SED 86] 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. 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:

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

$B_i^{n_i}$, $B_j^{n_j}$ and $B_k^{n_k}$ are the Bernstein polynomials of order n_i, n_j and n_k (see for instance [FAR 89]):

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

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

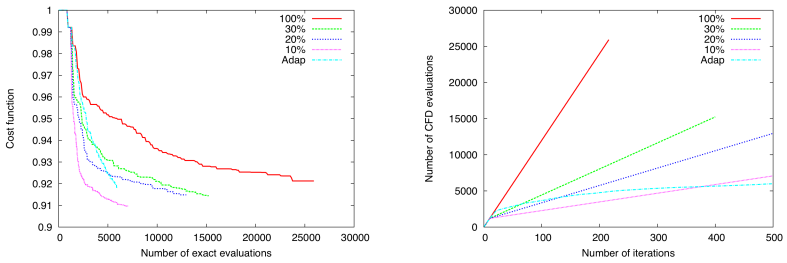
The FFD parameterization is built only around the wing as shown in figure (1) with ξ, η and ζ in the chordwise, spanwise and thickness directions respectively. The lattice is chosen in order to fit the planform of the wing as closely as possible. The leading and trailing edges are kept fixed during the 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.

0.4.3. *Metamodel-assisted PSO optimization*

We 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.

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 1. *Results of PSO for supersonic business jet*



(a) (b)

Figure 2. *Optimization of supersonic business jet: (a) Evolution of cost function, (b) Number of CFD evaluations*

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 (1) and figure (2). 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 (2-a) shows the evolution of the cost function as a function of the number of CFD evaluations while figure (2-b) shows the number of CFD evaluations as a function of the PSO iterations. Finally, figure (3) 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.

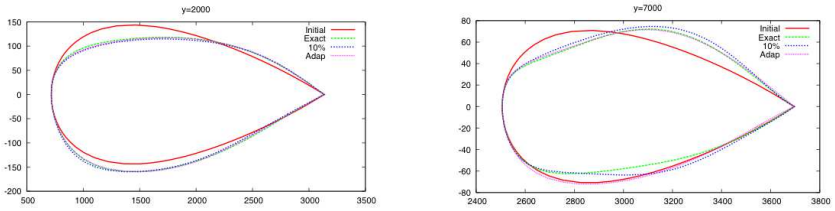


Figure 3. *Wing shapes for supersonic business jet at different spanwise stations*

0.5. Test-case 2: Transonic wing optimization

0.5.1. *Test-case description*

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 [AND 03]. The overall wing shape can be seen in figure (4). 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.

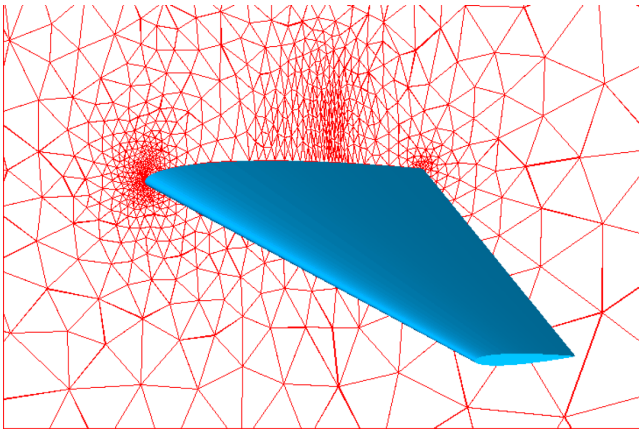


Figure 4. *Transonic wing: Initial wing shape (blue) and mesh in the symmetry plane (red).*

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 :

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

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 (4)).

0.5.2. *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. 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.

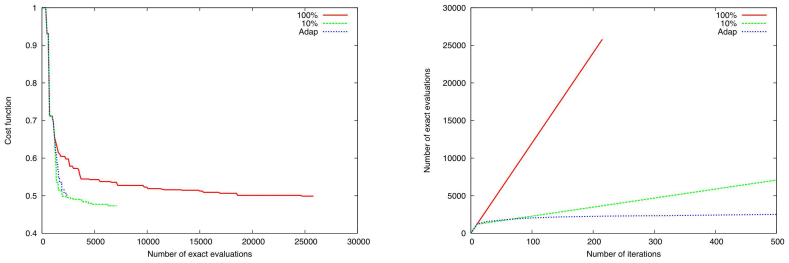
0.5.3. *Metamodel-assisted PSO optimization*

The optimization is performed using PSO with 120 particles and the same set of parameters as in previous section. The local metamodels are constructed using 40 nearest neighbours from the database. In the case of metamodel assisted PSO, 500 iterations are performed. Table (2) 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 (5-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 (5-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 (6) 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.

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

Table 2. *Optimization of a transonic wing*



(a) (b)

Figure 5. *Optimization of transonic wing: (a) Evolution of cost function, (b) Number of CFD evaluations*

0.6. Conclusion

The use of global optimization methods, such as Evolutionary Strategies or Particle Swarm Optimization, on the basis of high-fidelity solvers is still an issue, although the growth of computational facilities. Therefore, the development of hierarchical

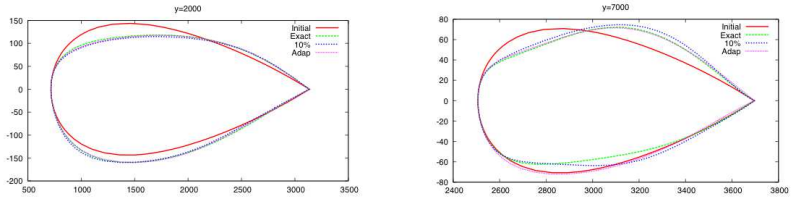


Figure 6. *Wing shapes for transonic wing at different spanwise stations*

methods, based on different modeling levels, is mandatory to solve realistic design problems in an industrial framework. Several strategies can be considered, ranging from the use of metamodels to the use of solvers relying on a simplified physics. However, the main issue today is the development of smart algorithms, which would be able to automatically adjust the modeling level required to solve a given design problem.

0.7. Bibliography

[AND 03] ANDREOLI M., JANKA A.DESIDERI J.-A., Free-form deformation parameterization for multilevel 3D shape optimization in aerodynamics, 5019, INRIA, November 2003.

[B¨ 05] BÜCHE D., SCHRAUDOLPH N. N.KOUMOUTSAKOS P., Accelerating evolutionary algorithms with Gaussian process fitness function models, *IEEE Tran. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 35, 2, 2005.

[DER 92] DERVIEUX A.DESIDERI J. A., Compressible flow solvers using unstructured grids, Research Report 1732, INRIA, June 1992.

[DUV 06] DUVIGNEAU R., CHAIGNE B.DESIDERI J.-A., Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using particle swarm optimization, Research Report RR-6003, INRIA, Sophia Antipolis, 2006.

[EMM 06] EMMERICH M., GIANNAKOGLOU K.NAUJOKS B., Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels, *IEEE Trans. Evol. Comput.*, vol. 10, 4, p. 421-439, 2006.

[FAR 89] FARIN G., *Curves and surfaces for computer-aided geometric design*, Academic Press, 1989.

[FOU 02] FOURIE P.GROENWOLD A., The particle swarm optimization in size and shape optimization, *Structural and Multidisciplinary Optimization*, vol. 23, 4, 2002.

[GIA 02] GIANNAKOGLOU K. C., Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence, *Prog. Aero. Sci.*, vol. 38, p. 43-76, 2002.

[JIN 05] JIN Y., A comprehensive survey of fitness approximation in evolutionary computation, *Soft Computing*, vol. 9, 1, 2005.

[KEN 95] KENNEDY J.EBERHART R., Particle swarm optimization, *IEEE International Conference on Neural Networks*, Perth, Australia, 1995.

[MIC 92] MICHALEWICS Z., *Genetic algorithms + data structures = evolutionary programs*, AI Series, Springer-Verlag, New York, 1992.

[MIL 07] MILLER P., Swarm behaviour, *National Geographic*, July 2007, available online at `http://www7.nationalgeographic.com/ngm/0707/feature5/`.

[ONG 04] ONG Y. S., NAIR P. B., KEANE A. J.WONG K. W., Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, JIN Y., Ed., *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing, Springer Verlag, 2004.

[SED 86] SEDERBERG T.PARRY S., Free-form deformation of solid geometric models, *Computer Graphics*, vol. 20, 4, p. 151-160, 1986.

[SHI 98] SHI Y.EBERHART R., A modified particle swarm optimizer, *International Conference on Evolutionary Computation*, 1998.

[VEN 03] VENTER G.SOBIESZCZANSKI-SOBIESKI J., Particle swarm optimization, *AIAA Journal*, vol. 41, 8, 2003.

[WIL 05] WILKE D. N., Analysis of the particle swarm optimization algorithm, Master's thesis, Department of Mechanical and Aeronautical Engineering, University of Pretoria, 2005.