Efficient Hierarchical Optimization using an Algebraic Multilevel Approach

Abderrahmane Benzaoui, Régis Duvigneau

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficient Hierarchical Optimization using an Algebraic Multilevel Approach

Abderrahmane BENZAOUI — Regis DUVIGNEAU

N° 6974

June 2009

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6974--FR+ENG

Efficient Hierarchical Optimization using an Algebraic Multilevel Approach

Abderrahmane BENZAOUI , Regis DUVIGNEAU

Thème NUM — Systèmes numériques
Projet OPALE

Rapport de recherche n° 6974 — June 2009 — 28 pages

Abstract: This paper presents an efficient method to reduce the optimization cost. In this method, the eigenvectors of the Hessian of the objective function are determined first. Then, the search for the optimum is carried out successively in subspaces defined by these vectors. For this purpose, the Multi-directional-Search Algorithm is used in this study, but any other optimization algorithm can be employed. The method is validated in two test cases: analytical function and shape reconstruction problem. In both cases, this method shows very promising results.

Key-words: Multilevel optimization, Spectral Decomposition of the Hessian, Multi-directional-Search Algorithm.

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 Hiérarchique Efficace basée sur une Approche Algébrique Multiniveau

Résumé : Ce rapport présente une méthode efficace permettant de réduire le coût des algorithmes d'optimisation. Dans cette méthode, les vecteurs propres du Hessien de la fonctionnelle sont déterminés dans un premier temps. Ensuite, la recherche de l'optimum est effectuée dans des sous-espaces définis par ces vecteurs. Pour cela, l'Algorithme de Recherche Multidirectionnelle est utilisé dans cette étude. Cependant, tout autre algorithme d'optimisation peut être employé. La méthode est validée sur deux cas tests : Une fonction analytique et un problème de reconstruction de forme. Dans les deux cas, cette méthode montre des résultats très prometteurs.

Mots-clés : Optimisation multiniveau, Décomposition Spectrale du Hessien, Algorithme de Recherche Multi-directionnelle.

1 Introduction

Optimization tools in engineering design problems very often require a high computational cost. This cost originates from two main sources: First, the evaluation of the optimized function involved in such problems is in general very expensive. Then, depending on the method employed and on the dimension of the design vector, the optimization procedure requires a high number of evaluations of the objective function to reach the final solution. Many studies aimed at reducing the computational cost of these tools by reducing the cost of evaluations or by using more efficient algorithms that require a lower number of call to the objective function.

To improve the efficiency of the algorithm, some authors propose to use a multilevel approach instead of a direct one. This means that the optimization is carried out, at some steps, on a coarse level where not all the design parameters are considered. Their idea is inspired from the multigrid theory used to solve problems with differential equations. Among this studies, Jameson and Martinelli [11] used the multigrid concept to solve simultanously the flow equations and the optimization problems. Lewis and Nash [14] used the same concept to solve optimization problems of systems governed by differential equation. Désidéri et al [6, 5, 8] generalized this approach to elaborate a multilevel shape optimization algorithm where the shape is parameterized using the Bézier curves or a Free-Form Deformation (FFD) technique [17]. In these studies, the coarse optimization is carried out on subspaces that depend on the parameterization of the geometry of interest. In our study, we propose a more general method that can accelerate the convergence of the optimization algorithm and can be employed for any kind of problem. This method uses the eigenvectors of the Hessian of the cost function to define the subspaces of optimization. This is a more efficient and abstract approach for a multilevel optimization. In this article, we introduce our multilevel method and present the corresponding algorithm. This method is then validated in two test functions in order to be employed in further studies.

This article is organised as follow: we start first by giving a short survey of the optimization methods. In particular, we pay more attention to the Multi-directional-Search Algorithm method (MSA) [4] which is used in this study. Then, we describe in details our multilevel approach. After that, we show the validation of this approach on two test cases: an analytical function and a shape reconstruction problem. Finally, the article is terminated by a general conclusion.

2 Survey of optimization methods

Optimizing a function f (generally called a cost function) is finding its minimum. In many problems, this minimization is subject to some constraints. To make the problem easier, the constraints are integrated in the cost function by means of a penality term or any other adequate way depending on the nature of these constraints. There are several methods of optimization which can be classified into two principal categories: deterministic methods and stochastic methods. Deterministic methods do not make use of random parameters when

looking for the optimum. So, the same routine, ran two times under the same conditions leads to the same results. Howevere, a stochatic method uses some random techniques to locate the optimum. Thus, the same routine, ran two times under the same conditions does not necessary lead to the same results. Even though, when converged, the result of this routine must be unique.

The deterministic methods are classified into gradient-based method and gradient-free ones. Gradient-based methods are the most classical and the most intuitive optimization techniques. Finding a minimum of a given convex function is looking for the zero of its gradient. One of the most popular methods is the steepest descent one. It uses the gradient vector to define a path of optimization. But this can have a low rate of convergence. The Newton method is an enhancement of the descend methods by means of the inverse of the Hessian matrix. Since the computation of the inverse of the Hessian is cumbersome, Quasi-Newton methods suggest an approximation of this matrix, which is updated at each iteration.

Many difficulties appear when using gradient-based methods. First, they need to compute the first and sometimes the second derivatives which can be costly and sensitive to numerical noises [10] specially when the derivatives are evaluated using techniques such as finite-difference method. These errors can generate false local minima. Then, Gradient-based methods can easily be trapped into local minima and so do not locate the global optimum. Many design problems are multimodal (i.e they have many local minima). So, these methods are not suited for such problems.

Simplex methods are an alternative of the gradient-based ones. These methods does not make use of the derivatives of the cost function. their main idea is the displacement of a simplex (i.e a set of $n+1$ design vectors, where n is the dimension of these vectors) in the design space, so to get a decrease of the cost function. The success of these methods is motivated by the development of parallel machines. The $n+1$ evaluations of the cost function can then be made simultaneously. Among the Simplex methods, The Nelder-Mead Algorithm, and the Multi-Directional-Search Algorithm (MSA) are the most popular [4] [16]. Nevertheless, as the gradient-based methods, the Simplex methods are also trapped into local minima and so not very suited to multimodal problems.

Stochastic methods are known to be more robust and able to avoid local minima. These methods generally mimic natural phenomena to obtain the optimum. Among the stochastic methods, the Evolutionary strategies and Particle Swarm Optimisation method (PSO) are the most commonly used. These techniques use a set of candiate solutions called population. In Evolutionary Algoritms, the candidate solutions are treated as if they were biological spieces that evolve to a best fitness. Using some operators that mimic biological behaviours (such as reproduction, selection, mutation), the optimum solution can be obtained after some generations (iterations). In Particle Swarm Optimization, The candidate solutions are treated as if they were animals that move to a best location (to find food or to avoid a predator for exemple). At each iteration, the new position of each particle is influenced by its own memory (local memory) and by the memory of its neighbours (global memory). The operators of the stochastic methods use some random parameters in order to search

the solution into the entire design space. For this reason, they are less likely to be trapped by local minima and they are more able to locate global optimum. The main drawback of these methods is the number of the cost function evaluations which depends on the size of the population. Hence the stochastic methods are generally expensive if they do not use a coarse approximation of the cost function.

A survey of the optimization techniques can be found in [7] for instance. The aim of our study is not to compare these techniques. We just need to use one of them to validate our Multilevel optimization algorithm. Because of its simplicity, the MSA algorithm was employed. Bellow, we present in details this algorithm.

3 Multi-directional-Search Algorithm

This algorithm, developped by Dennis and Torczon [4], uses a simplex to find the optimum. This simplex is composed of $n+1$ design vectors X_0, X_1, ..., X_n. After being initialised, the simplex in each new iteration is obtained as follows:

Suppose that $X_0^{(k)}$ is the vertex that gives the smallest cost function among all the simplex vertices, where k is the iteration number. Then the simplex is reflected with respect to $X_0^{(k)}$ according to the following equation:

$$\widetilde{X_i^{(k)}} = (1+\alpha)X_0^{(k)} - \alpha X_i^{(k)} \; i = 0, ..., n \tag{1}$$

where $\widetilde{X_i^{(k)}}$ designate the reflected vertices and α is a positif real usually equal to 1. If the reflexion is successful, i.e if one of the new vertices has a smallest cost function than that of $X_0^{(k)}$, this means that perhaps the solution is in this direction. So the new simplex is expanded to pursue the search in this direction. The simplex of the new iteration is then obtained by:

$$X_i^{(k+1)} = (1-\gamma)\widetilde{X_0^{(k)}} + \gamma\widetilde{X_i^{(k)}} \; i = 0, ..., n \tag{2}$$

where $\gamma > 1$ is the expansion coefficient. Usually γ is set to 2. In the other case, if the reflextion fails, the simplex is contracted so that the vertices come closer to the best one. The simplex of the new iteration is then obtained by:

$$X_i^{(k+1)} = (1-\beta)\widetilde{X_0^{(k)}} + \beta\widetilde{X_i^{(k)}} \; i = 0, ..., n \tag{3}$$

where β is the contraction coefficient. Usually β is set to $-\frac{1}{2}$.

The algorithm is carried out until satisfying a given stopping criterion. Several choices for this criterion are possible. In our study, we compute the distance (using Euclidean norm) from the best vertex $X_0^{(k)}$. Then, the stopping criterion is given by:

$$\frac{max_i \left(\parallel X_i^{(k)} - X_0^{(k)} \parallel \right)}{\parallel X_{ref} \parallel} < \epsilon \tag{4}$$

where X_{ref} is a given reference vector.

While this condition is not satisfied and the number of iteration is below the maximum value for the selected level, the MSA algorithm pursues the iterations.

4 Multilevel optimization

When studying an optimization problem in fluid dynamics for instance, many difficulties occur. First, the physical problem is modeled by complex equations (such as Navier-Stokes equations), so that the evaluation of the cost function is cumbersome. Then, the numerical computation of the physical problem is expensive. And finally, the fine parameterization of the optimized object leads to a high dimension design vector which results in a stiff optimization search. Many authors proposed hierarchical techniques to make the optimization algorithm cheaper. Among these techniques, we can cite the use of a simplified model of the physical problem (for exemple, the use of Euler equations instead of the Navier-Stokes ones), the use of a metamodel instead of the exact model, or the use of hierarchical parameterization instead of a single level one. Here we do not describe nor give an exhaustive list of these hierarchical techniques. The interested reader can refer for instance to [9]. In our study, we are interested only by techniques concerning the parameterization level of the optimised object (Multilevel algorithms).

The idea of the multilevel algorithms is to accelerate the fine level optimization by looking for the solution on a coarse level when necessary. This is inspired by the multigrid method used to solve problems with partial differential equations. Indeed, it is well known in such problems, that the computation of a fine solution is expensive not only because of the increased iterations cost (resolution of a high dimension linear system), but also because of the low rate of convergence of the iterative algorithm. This is why the multigrid techniques use a coarse level to accelerate the fine level resolution. Several strategies can be considered, ranging from simple level increase to V-cycle or Full Multi-Grid approaches [6].

The idea is similar in optimization. Earlier studies reproduced successfully the multigrid concept and strategies to optimization problems. In particular, Jameson and Martinelli [11] generalised a multigrid code to optimization in aerodynamic design. Lewis and Nash [14, 15] used the multigrid concept for optimization of systems governed by differential equations. A survey of these methods can be found in [2]. Furthermore, Désidéri et al used the multigrid approach to elaborate a multilevel shape optimization with a Bézier or a FFD parameterization [1, 6, 8]. In these studies, when the Bézier or FFD parameterization is employed, the transfer from a coarse level to a finer one is done using the Bézier degree-elevation process, which is a straightforward technique that permits to add some control points without anay modification on the optimized shape. Even though, this is not the best way to use a multilevel strategy.

In the present study, a more efficient method is proposed by combining the multigrid concept with the spectral decomposition of the Hessian matrix. Indeed, for a descent method for instance, the smallest eigenvalues of the Hessian matrix correspond to directions where the convergence of the optimization algorithm is very slow, while the highest eigenvalues

correspond to directions where the convergence is fast. Thus, instead of iterating on the entire design space, our algorithm looks for the solution in a selected subspace in order to get a faster solution in the directions of low convergence rate. Then it pursues the search on the entire design space. This can be down by several strategies analogous to those of the multigrid method.

Let E be the entire design space and $X_0 \in E$ is a starting design vector. The multilevel algorithm starts first by computing the Hessian matrix $H(X_0)$ and its eigenvectors $V = (v_1, ..., v_n)$. These vectors are ranked from the smallest to the highest corresponding eigenvalue. Suppose that $X^{(l)} \in E$ is the design vector obtained at a level l. Then, the multilevel algorithm looks for the new design vector $X^{(l+1)} \in E$ at the new level $l+1$ by adding a correction term which minimizes the cost function in the current optimization subspace. This is done as follows:

Suppose that the new level is characterized by $m = m_{l+1}$ parameters ($m \leq n$) and consider the basis V_m and the subspace E_m defined by:

$$V_m = (v_1, ..., v_m) \tag{5}$$

$$E_m = \{z = V_m y \mid y \in \Re^m\} \tag{6}$$

The correction term must be a vector from E_m. Hence, to find the best correction, the optimization algorithm looks for a vector $y \in \Re^m$ so that the cost function:

$$g(y) = f(X^{(l)} + V_m y) \tag{7}$$

is minimized. In this way, we can either go from a coarse to a finer level or from a fine to a coarser level without loosing any information obtained from the former level. In our case, using the MSA algorithm, at each new level we start by the initialization of a simplex of $m+1$ vertices in $\Re^m$ (one of the vertices is zero). Then, the MSA algorithm finds the optimum value y^* that minimizes (7). The design vector of the new level is then:

$$X^{(l+1)} = X^{(l)} + V_m y^* \tag{8}$$

The resolution at any level can be full or incomplete. The algorithm can be carried out using several levels, ordered in some way called a cycle, by analogy to the multigrid terminology. So a cycle is defined by a sequence of levels of dimensions $(m_1, m_2, ..., m_{nl})$ and by the corresponding numbers of iterations $(it_1, it_2, ..., it_{nl})$ where nl is the number of levels in the cycle. Then, the cycle can be repeated many times if necessary.

The multilevel algorithm is summarized below:

1. Read the input data and initialize the design variable X_0;
2. initialize the number of cycles at $k = 0$;
3. while $k < nc$ and the stopping criterion not satisfied, perform iterations on a multilevel cycle:

(a) compute the Hessian matrix $H(X_k)$ (using an exact or a surrogate model) and evaluate its eigenvectors;

(b) initialize the level number at $l = 1$;

(c) let $X^l = X_k$

(d) while $l \leq nl$ do:

i. select a basis V_m with $m = m_l$ eigenvectors,

ii. use the MSA optimiser to perform a correction vector $y^* \in \Re^m$ that minimizes the cost function $g(y) = f\left(X^{(l)} + V_m y\right)$;

iii. $X^{(l+1)} \longleftarrow X^{(l)} + V_m y^*$;

iv. $l \longleftarrow l + 1$ and goto (3d);

(e) $X_{k+1} \longleftarrow X^{(l)}$;

(f) $k \longleftarrow k + 1$ and goto (3);

4. $X^* \longleftarrow X_{(k)}$.

The stopping criterion in the step (3) of the above algorithm is the relative decrease in the cost function. While this decrease is above a given value, the algorithm pursues the computation. Furthermore, in step (3(d)i), the m_l eigenvectors correspond to the smallest eigenvalues of the Hessian. In this way, on a coarse level, the algorithm focuses only on directions with low convergence rate.

5 Applications

5.1 First test case: analytical function

This test case is a straightforword problem that allows us to validate our optimization algorithm and to find the best multilevel strategy. The function to be optimized is a quadratic function given by the following expression:

$$f(X) = a + B^T X + X^T C X \tag{9}$$

where $X \in \Re^n$ is the optimization parameter. In this function, $a \in \Re$, $B \in \Re^n$ and $C \in \Re^{n \times n}$ are chosen arbitrarily. But to guarantee the existence of a minimum, C is taken as a symmetric positive definite matrix. In this case, the Hessian of f is simply equal to $2C$ and is also positive definite. Since we would like to use this function to test the multilevel optimization with a Hessian decomposition approach, C is chosen in a manner that allows us to control its eigenvalues and eigenvectors. More precisely, for $n = 12$, we have chosen: $a = 2$ and $B = (1, 3, 2, 4, -1, 5, 10, 8, 9, 0, -4, 12)^T$. The matrix C is given in appendix (A).

This function has a unique minimum that can be obtained by considering that the gradient is equal to zero. This is equivalent to solve the system

$$CX = -\frac{1}{2}B \tag{10}$$

In order to examine the effect of the condition number of the Hessian matrix on the optimization procedure, we have tested three functions of the form (9) where only the matrix C is modified. Table (1) gives the condition number and the optimal value for the three tested functions.

Function	Condition number	Optimal value
$f1$	10	-39.55215
$f2$	18000	-32.517335
$f3$	10^{11}	-21.79586

Table 1: Condition numbers and optimal values for the three tested functions.

In the following section, we try to find these values using the optimization algorithm. The efficiency of the multilevel strategy is measured by the number of evaluations of the function required to reach the optimum.

5.1.1 Single level optimization

Before testing the multilevel algorithm we try to find the optimum value of the above function using the MSA method without any multilevel nor Hessian decomposition strategy. For instance, for the function $f2$, we carry out 35000 iterations of the MSA optimizer in order to find a good approximation of the value given in table (1). The optimum value obtained is $f_{min} = -32.516920$. We have so a relative error of 0.001%. The corresponding number of evaluations is 910014, which is very high. This means that using the same algorithm in a problem with the same parameterization size, and where the evaluation of the cost function is expensive can be very stiff, if possible. So a limitation in the accuracy is then necessary. If we limitate the iteration number to 5000, which is still a high number, the optimum value obtained is -29.2075 and the relative error is 10%.

This exemple shows the necessity to improve the optimization procedure to become faster. In order to validate our multilevel algorithm and to prove its efficiency, we start by optimizing the above function using a simple level strategy with a spectral decomposition of the Hessian. This means that the optimized function is equivalent to that of equation (7) where $m = n$. Therefore, it consists in working in the eigenvectors basis. This is done by setting $nl = 1$ and $m_1 = n$ in the algorithm described in §4. The evolution of the cost function with respect to the number of evaluations is presented in figure (1). The result of this optimization is spectacular. The simple use of the Hessian eigenvectors basis permits the MSA optimizer to converge very fast. Indeed, the optimum value $f_{min} = -32.517335$ was reached after 130 iterations only with 3394 evaluations of the cost function. This method is thus 268 times faster then the first one! As we will see, this performance depends on the condition number of the Hessian matrix and can be better using an adequate multilevel strategy.

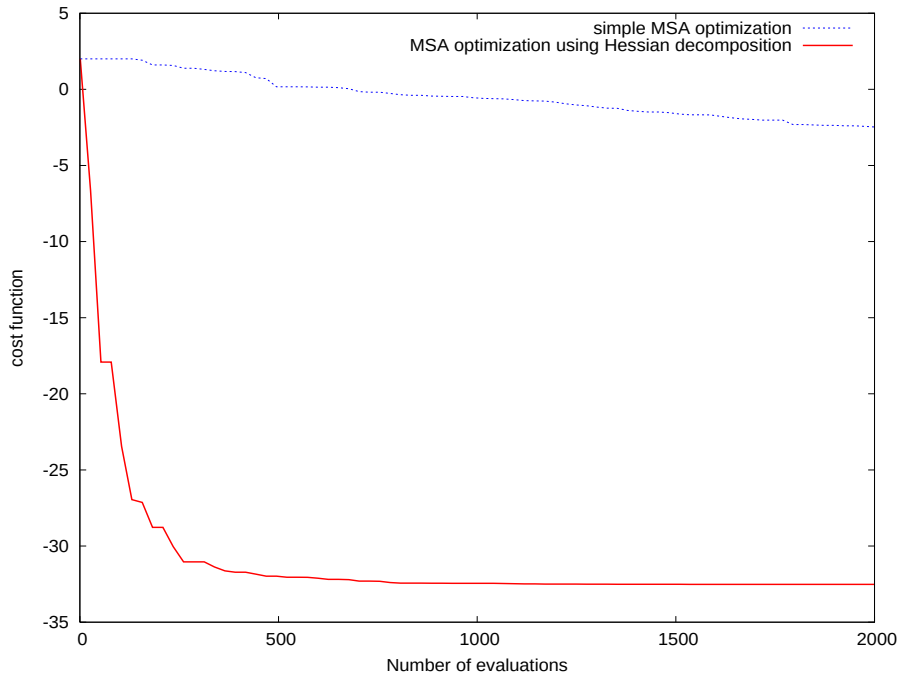

Figure 1: Evolution of the cost function $f2$ for a single level optimization.

5.1.2 Multilevel optimization

As mentioned above, the multilevel strategy can improve the performance of the optimization algorithm when it is used in an adequate manner. For the same function $f2$, we carry out the algorithm described in §4 with two and three levels using the strategy shown in table (2). To be efficient, it is recomended to use a few number of iterations at each level and to repeat the multilevel cycle until convergence. However, a slight modification on this algorithm is introduced. It consists on dividing the initial simplex by two at each cycle. Without this modification, when the design variable X is near the final solution, the first MSA iterations at each level become inefficient because the simplex vertices are relatively far from the solution. In this case, no decrease in the cost function can be obtained if we use a very few number of iterations. So, a small simplex is then more suitable.

Table (3) presents the multilevel tests for the function $f2$ using the strategy described in table (2), whereas figure (2) shows the evolution of cost function with respect to the number of evaluations for the best cases. In table (3), the case 1 corresponds to the single level optimization using the spectral decomposition of the Hessian.

Cycles	cycle 1	cycle 2	cycle 3
Fine level resolution	• ↘	• ↘	• ↘
Coarse level resolution	•	•	•

Table 2: schematic descreption of the two-level cycles.

Case	levels	number of iterations by cycle	number of cycle	number of evaluations	optimal value
1	12	130	1	3394	−32.517335
2	12 6	3 3	19	2661	−32.517335
3	12 6	3 2	22	2773	−32.517307
4	12 6	3 4	18	2773	−32.517335
5	12 6	2 4	22	2817	−32.517302
6	12 6	4 2	18	2737	−32.517335
7	12 6 3	3 2 2	17	2483	−32.517335

Table 3: Description of the Multilevel optimization of the analytical function $f2$.

As we can see in table (3) and in figure (2), the multilevel strategy allows a faster optimization than the single level one. Indeed, if we compare cases 1 and 7 for instance, we can see that the multilevel strategy allows a reduction in the number of evaluations of about 27%. The best multilevel tests we obtained for this function are case 2 for two-level optimization and case 7 for three-level optimization. In these cases, the number of iterations by level is small but enough to get good performances. It is not necessary to use a higher number of iterations by level because this will result in an increasing number of

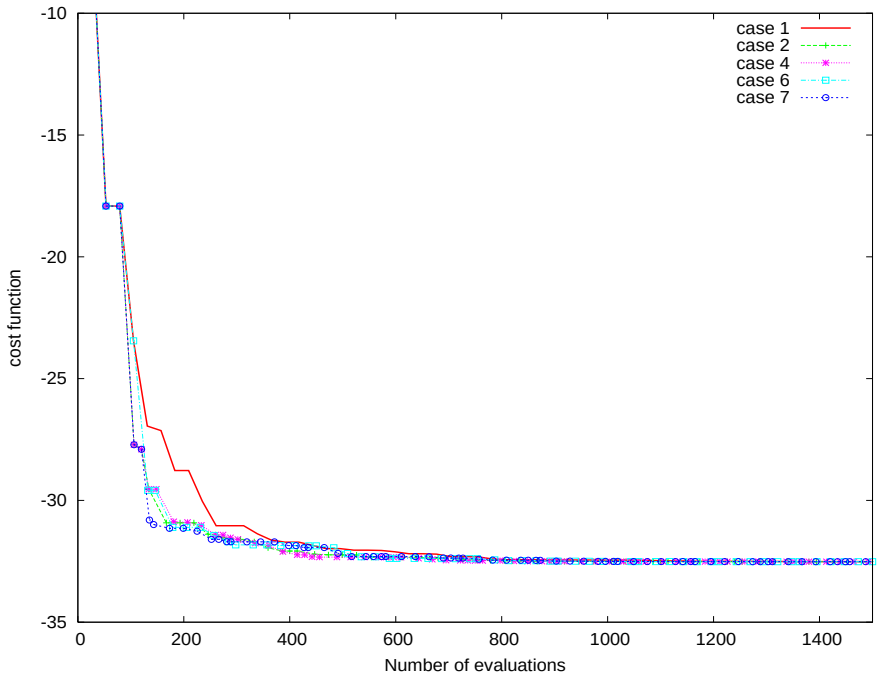

Figure 2: Evolution of the cost function $f2$ for a multilevel optimization.

evaluations without any acceleration of the convergence. However, if we reduce the number of iterations, the convergence becomes slow resulting in an increasing number of cycles and so an increasing number of evaluations, while the accuracy on the optimum value is worst.

5.1.3 The effect of the condition number

Tables (4) and (5) present the multilevel tests for the functions $f1$ and $f3$, whereas figures (3) and (4) show the evolution of cost function with respect to the number of evaluations for the best cases. In these tables, cases 0 refers to the single level optimization using a simple MSA method, while cases 1 refers to the single level optimization using the spectral decomposition of the Hessian.

We can see on these tables and figures that, when the condition number of the Hessian is small (function $f1$), the Hessian decomposition is not interesting since the total number of evaluations at the convergence is higher slightly in case 1 than that in case 0 of table (4). Even though, the multilevel approach is still interesting and a very good reduction on the total number of evaluations can be obtained (about 46% of reduction between cases 2 and 0). Note that, using the MSA method, the optimization at the coarse level needs a

Case	levels	number of iterations by cycle	number of cycle	number of evaluations	optimal value
0	12	180	1	4694	−39.55215
1	12	210	1	5474	−39.55215
2	12 6	3 ↘ 3	18	2521	−39.55215
3	12 6	4 ↘ 3	19	3155	−39.55215
4	12 6 3	3 ↘ 2 ↘ 2	18	2629	−39.55215

Table 4: Multilevel optimization of the analytical function $f1$ with a V-cycle strategy.

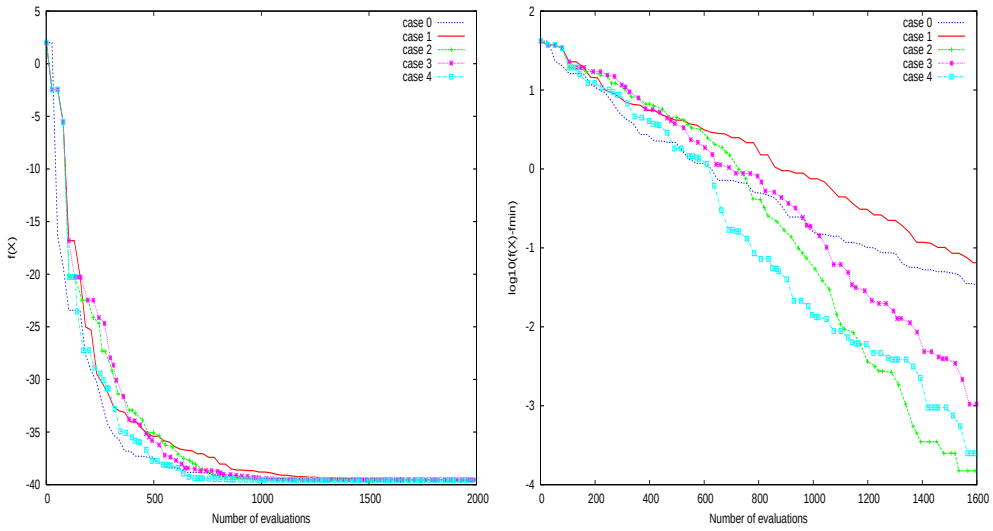

Figure 3: Evolution of the cost function $f1$ for a single level and a multilevel optimization.

smaller number of evaluations than the optimization at the fine level, because the number of vertices is smaller (equal to $m+1$ instead of $n+1$). But this is not the unique reason for which the multilevel optimization is interesting. Indeed, if we compare cases 2 and 0 in

Case	levels	number of iterations by cycle	number of cycle	number of evaluations	optimal value
0	12	1000	1	26014	1.997
1	12	82	1	2146	−21.735863
2	12 6	3 ↘ 3	12	1681	−21.735863
3	12 6	4 ↘ 3	12	1993	−21.735863
4	12 6	3 ↘ 4	12	1849	−21.735863
5	12 6 3	3 ↘ 2 ↘ 2	15	2191	−21.735862
6	12 6 3	3 ↘ 2 ↘ 3	12	1849	−21.735863

Table 5: Multilevel optimization of the analytical function $f3$ with a V-cycle strategy.

table (4) we can see that not only the number of evaluations is reduced, but also the total number of iterations. This means that, even though its effect is not very strong, looking for the solution on subspaces generated by the eigenvectors permits to accelerate the global convergence of the optimization algorithm.

Nevertheless, when the condition number of the Hessian is high (function $f3$), the Hessian decomposition seems to be necessary and very efficient. Indeed, After 1000 iterations, the single level MSA optimizer without decomposition of the Hessian is unable to obtain any sensitive decrease in the cost function. However, when the Hessian decomposition is employed, only 82 iterations are needed to reach the optimal value (see cases 0 and 1 of table (5)). The multilevel approach is of mild interest for this function. Indeed, in case 2 we gain about 22% in the number of evaluations if we compare it to case 1.

From these tests we can conclude that the multilevel strategy permits to accelerate the convergence of the optimization algorithm and to reduce its cost. This is true whatever is the condition number of the Hessian matrix. If this number is high, it becomes necessary

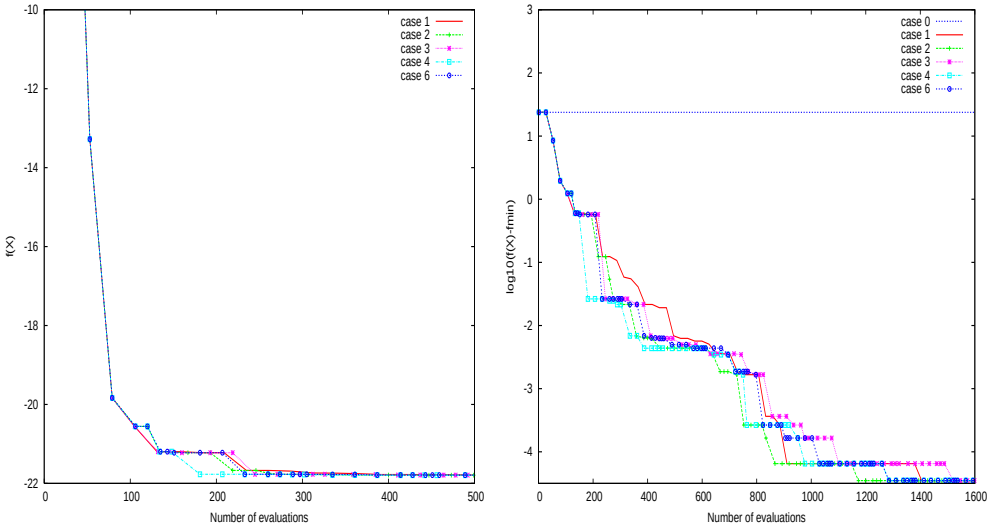

Figure 4: Evolution of the cost function $f3$ for a single level and a multilevel optimization.

to look for the solution in subspaces generated by the eigenvectors of the Hessian. In this case, the gain in the optimization cost is significant.

5.2 Second test case: Shape reconstruction problem

5.2.1 Test-case description

In this test case, we would like to approach a given target function $F_0(t)$ by a Bézier curve $F(t, X)$, where $X = (x_1, ...x_n)^T$ is the design vector whose components are the control points of the Bézier curve. So, the approximated function is given by:

$$F(t, X) = \sum_{k=0}^{n-1} B_{n-1}^k(t) x_{k+1} \tag{11}$$

for $t \in [0, 1]$. In equation (11), n is the parameterization level and $B_{n-1}^k(t)$ are the Bernstein polynomials given by:

$$B_n^k(t) = C_n^k t^k (1-t)^{n-k} \tag{12}$$

where

$$C_n^k = \frac{n!}{k!(n-k)!} \tag{13}$$

The target function we select in this study is also a Bézier curve[1] of degree $n_0 > n$. It is given by:

$$F_0(t) = \sum_{k=0}^{n_0-1} B_{n_0-1}^k(t) x_{0k+1} \tag{14}$$

where x_{01},...x_{0n_0} are the control points of the target function whose value are given in appendix (B).

The squared error of the approximation of the target function $F_0(t)$ by the Bézier curve $F(t, X)$ is given by:

$$e(X) = \int_0^1 (F(t, X) - F_0(t))^2 dt \tag{15}$$

For a given parameterization level n, the approximation problem is equivalent to the search of a design vector X that minimizes the squared error. This is hence an unconstrained optimization problem where the cost function is $f(X) \equiv e(X)$. A theoretical solution of this problem can be found easily by considering that the gradient of the cost function vanishes at the optimum. This leads to the following linear system:

$$A \cdot X = \mathcal{B} \tag{16}$$

where $A = (a_{kj})_{1 \le k \le n,\ 1 \le j \le n}$, $\mathcal{B} = (b_k)_{1 \le k \le n}$ and:

$$a_{kj} = 2 \int_0^1 B_{n-1}^{k-1}(t) B_{n-1}^{j-1}(t) dt \tag{17}$$

$$b_k = 2 \int_0^1 B_{n-1}^{k-1}(t) F_0(t) dt \tag{18}$$

Note that the matrix A is equal to the Hessian of the cost function and is symmetric positive definite. This quadratic problem has a unique solution which can be obtained by solving the system (16).

Note also that the above problem is not constrained since we have no condition on the control points. However, it is usual to impose conditions such as $x_1 = F_0(0)$ and $x_n = F_0(1)$. In this case, the resolution of the problem does not change, but its dimension is reduced to $n - 2$ since the unkown control points become $x_2, ..., x_{n-1}$.

To evaluate the cost function, the interval $[0, 1]$ is subdivided into n_p points. The squared error (15) is then given by:

$$e(X) \approx \frac{1}{n_p} \sum_{p=1}^{n_p} \left(F\left(\frac{p-1}{n_p - 1}, X \right) - F_0\left(\frac{p-1}{n_p - 1} \right) \right)^2 \tag{19}$$

[1] This is just an exemple and we can select any kind of functions. The resolution method and the behaviour of the optimization algorithm do not depend on the nature of the target shape.

A similar discretization is done to compute the Hessian matrix A, given by (17), and the right hand side term of equation (16) which is given by (18).

In order to test our optimization algorithm, we approximate two target functions F_{01} and F_{02} of parameterization levels 14 and 18 by respectively functions F_1 and F_2 of levels 8 and 16. The corresponding cost functions are so f_1 and f_2 respectively. This allows us to test at the same time the ability to approximate different shapes and the influence of the condition number of the Hessian matrix on the algorithm. Table (6) gives the optimum value and the condition numbers corresponding to these cost functions for $n_p = 20$, whereas figure (5) gives the shape of the target and the approximated functions.

Cost Function	n_0	n	Condition number	Optimal value
f_1	14	8	4617.6	1.751264×10^{-5}
f_2	18	16	3.6×10^{9}	1.037×10^{-9}

Table 6: Condition numbers and optimal values for the two tested functions of the shape reconstruction problem.

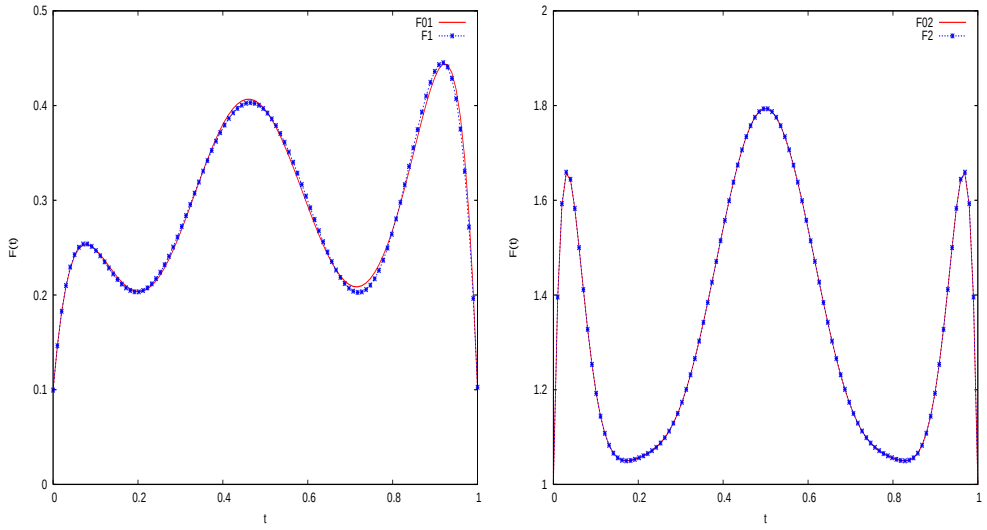

Figure 5: Target and approximated curves for functions F_{01} (left) and F_{02} (right) of the shape reconstruction problem.

5.2.2 Experimentation and results

In this section, we try to obtain the approximated function using the optimization algorithm. As for the first test case, we test many cases in order to find the best optimization strategy. Tables (7) and (8) describe some cases for cost functions f_1 and f_2 respectively (corresponding to the target functions F_{01} and F_{02}) wheras figure (6) shows the evolution of these functions with respect to the number of evaluations for the studied cases.

Case	levels	number of iterations by cycle	number of cycle	number of evaluations	optimal value $\times 10^5$
0	8	200	1	3610	2.303
1	8	135	1	2440	1.751264
2	8 4	3 ↘ 5	18	2125	1.751264
3	8 4	3 ↘ 4	19	2053	1.751264
4	8 4	3 ↘ 3	15	1471	1.751264
5	8 4 2	3 ↘ 2 ↘ 2	19	1958	1.751264

Table 7: The tested cases for the multilvel optimization of the cost functions f_1 of the shape reconstruction problem.

From tables (7) and (8) and figure (6) we can see that the behaviour of the optimization algorithm is the same as that for the first test case. For medium Hessian condition, the simple use of the Hessian decomposition allows the algorithm to be faster. The multilevel strategy permits to accelerate the convergence. The best acceleration is obtained with few iterations by level and by cycle (cases 4 and 5 for function f_1 and 2 and 4 for function f_2). This acceleration is more important for high condition number.

6 Conclusion

In this paper we present an efficient approach of optimization based on the use of an algebraic multilevel method. This method uses the eigenvectors of the Hessian of the objective function to define subspaces of optimization. This approach is described in details and validated

Case	levels	number of iterations by cycle	number of cycle	number of evaluations	optimal value
0	16	200	1	6818	0.00041
1	16	140	1	4778	1.037×10^{-9}
2	16 8	3 ↘ 3	20	3641	1.037×10^{-9}
3	16 8	4 ↘ 3	20	4321	1.037×10^{-9}
4	16 8 4	3 ↘ 2 ↘ 2	20	3781	1.037×10^{-9}

Table 8: The tested cases for the multilvel optimization of the cost functions f_2 of the shape reconstruction problem.

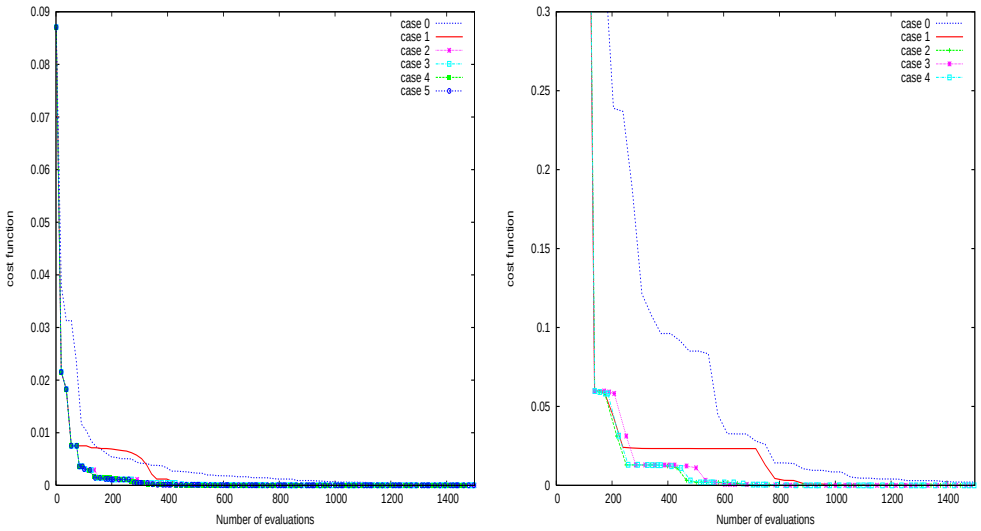

Figure 6: Evolution of the cost functions f_1 (left) and f_2 (right) of the shape reconstruction problem for several optimization strategies.

on two test cases. It is shown with this test cases, that the use of the eigenvectors of the Hessian as basis of the design space accelerates the convergence of the optimization algorithm. This acceleration is significant for medium and high condition number of the Hessian. A slight additional acceleration is obtained when the Hessian decomposition is combined to an adequate multilevel strategy similar to those of the multigrid method.

The evaluation of the Hessian matrix of the objective function in the test cases is straightforward. This is why it is computed exactly. Nevertheless, for much complicated problems, where no analytical expression for the Hessian is available, it can be very useful to approximate it by a surrogate model. Indeed, if the Hessian is evaluated by the Finite-Difference method for exemple, the total number of call to the objective function increases by an order of n^2 call for each Hessian evaluation, where n is the dimension of the design vector. So the surrogate model can reduce the cost of the Hessian evaluation sensitively.

The test problems studied in this article are quadratic and convex, that means that they have a unique minimum and a constant positive difinite Hessian. For these problems, the optimization using the Multi-directional-Search method is easy. However, this is seldom the case in practice. Most of the engineering problems involve multimodal functions and a method such MSA is not suitable because it is not able to avoid local minima. Then, it would be interesting to test our multilevel approach using a stochastic algorithm such as Particle Swarm Optimization, more suitable for multimodal problems.

7 Acknowledgments

This study hase been supported by the "OMD" project (Multi-Disciplinary Optimization) granted by ANR-RNTL.

Appendices

A Analytical function for testing the multilevel algorithm

The analytical functions used in this study to test the multilevel algorithm are given by the general experssion:

$$f(X) = a + B^T X + X^T C X$$

where $a = 2$ and $B = (1, 3, 2, 4, -1, 5, 10, 8, 9, 0, -4, 12)^T$. The matrix C is constructed in such a way that permits us to control its eigenvalues and eigenvectors. Indeed, since we want to test a method based on the eigenvectors subspaces, it is preferable that the eigenvectors of the matrix C should be different from the unit vectors of the canonical basis. In practice, to get such a matrix, we start from a diagonal one which has the desired eigenvalues. Then, by means of successive rotations arround all the unit vectors, we obtaine a matrix that conserves the same eigenvalues but for which the eigenvectors are different from the canonical basis.

In order to test the effect of the condition number of the Hessian, three test functions, f_1, f_2 and f_3 were considered in this study. These functions have the same values for a and B but differente matrices C_1, C_2 and C_3 respectively. In the following, we give the details about these functions:

- eigenvalues of the matrix C_1: $1, 2, 2.5, 3, 4, 4.5, 5, 6, 7.2, 8, 9.5, 10$
- eigenvalues of the matrix C_2: $1, 1000, 14000, 50, 70, 80, 90, 15, 2, 3, 40, 18000$
- eigenvalues of the matrix C_3: $1, 10, 100, 1000, 10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}, 10^{11}$
- columns 1 to 4 of the matrix C_1:

$9.7629155 \times 10^{+00}$	$5.6208746 \times 10^{-01}$	$-1.0626881 \times 10^{-01}$	$-9.2818830 \times 10^{-02}$
$5.6208746 \times 10^{-01}$	$8.1451627 \times 10^{+00}$	$1.3770233 \times 10^{+00}$	$1.3850436 \times 10^{-01}$
$-1.0626881 \times 10^{-01}$	$1.3770233 \times 10^{+00}$	$6.6849147 \times 10^{+00}$	$1.8063792 \times 10^{+00}$
$-9.2818830 \times 10^{-02}$	$1.3850436 \times 10^{-01}$	$1.8063792 \times 10^{+00}$	$5.4974392 \times 10^{+00}$
$-5.9758123 \times 10^{-02}$	$2.0864099 \times 10^{-02}$	$3.0582552 \times 10^{-01}$	$1.9782906 \times 10^{+00}$
$-4.0956354 \times 10^{-02}$	$2.9328205 \times 10^{-02}$	$6.3251268 \times 10^{-02}$	$3.8459803 \times 10^{-01}$
$-3.0950247 \times 10^{-02}$	$4.9356755 \times 10^{-02}$	$9.2080432 \times 10^{-03}$	$-4.7742210 \times 10^{-02}$
$-2.3490984 \times 10^{-02}$	$5.3687066 \times 10^{-02}$	$-1.5039154 \times 10^{-02}$	$-4.6279645 \times 10^{-02}$
$-1.8537947 \times 10^{-02}$	$6.2408534 \times 10^{-02}$	$-8.7547586 \times 10^{-02}$	$9.1268011 \times 10^{-02}$
$-1.3495635 \times 10^{-02}$	$4.8514758 \times 10^{-02}$	$-5.2914361 \times 10^{-02}$	$2.7131573 \times 10^{-03}$
$-7.6144911 \times 10^{-03}$	$1.3452063 \times 10^{-02}$	$3.1096754 \times 10^{-02}$	$-4.4863302 \times 10^{-02}$
$2.7329545 \times 10^{-03}$	$8.5515493 \times 10^{-04}$	$-5.1350838 \times 10^{-02}$	$1.4255185 \times 10^{-01}$

- columns 5 to 8 of the matrix C_1:

$-5.9758123 \times 10^{-02}$	$-4.0956354 \times 10^{-02}$	$-3.0950247 \times 10^{-02}$	$-2.3490984 \times 10^{-02}$
$2.0864099 \times 10^{-02}$	$2.9328205 \times 10^{-02}$	$4.9356755 \times 10^{-02}$	$5.3687066 \times 10^{-02}$
$3.0582552 \times 10^{-01}$	$6.3251268 \times 10^{-02}$	$9.2080432 \times 10^{-03}$	$-1.5039154 \times 10^{-02}$
$1.9782906 \times 10^{+00}$	$3.8459803 \times 10^{-01}$	$-4.7742210 \times 10^{-02}$	$-4.6279645 \times 10^{-02}$
$4.5304527 \times 10^{+00}$	$1.8118005 \times 10^{+00}$	$6.9393908 \times 10^{-01}$	$3.3417883 \times 10^{-02}$
$1.8118005 \times 10^{+00}$	$4.2043160 \times 10^{+00}$	$1.4764748 \times 10^{+00}$	$8.8013774 \times 10^{-01}$
$6.9393908 \times 10^{-01}$	$1.4764748 \times 10^{+00}$	$4.0756228 \times 10^{+00}$	$1.1484860 \times 10^{+00}$
$3.3417883 \times 10^{-02}$	$8.8013774 \times 10^{-01}$	$1.1484860 \times 10^{+00}$	$4.0376518 \times 10^{+00}$
$-9.0394638 \times 10^{-02}$	$1.0894871 \times 10^{-01}$	$8.9514996 \times 10^{-01}$	$8.0315921 \times 10^{-01}$
$1.3949754 \times 10^{-02}$	$-3.1079111 \times 10^{-02}$	$2.4563596 \times 10^{-01}$	$8.5497151 \times 10^{-01}$
$-6.9448221 \times 10^{-02}$	$3.1241715 \times 10^{-02}$	$3.6622936 \times 10^{-02}$	$3.0216088 \times 10^{-01}$
$-1.7933843 \times 10^{-01}$	$1.0483966 \times 10^{-01}$	$-1.8180862 \times 10^{-02}$	$-2.3542808 \times 10^{-01}$

- columns 9 to 12 of the matrix C_1:

$-1.8537947 \times 10^{-02}$	$-1.3495635 \times 10^{-02}$	$-7.6144911 \times 10^{-03}$	$2.7329545 \times 10^{-03}$
$6.2408534 \times 10^{-02}$	$4.8514758 \times 10^{-02}$	$1.3452063 \times 10^{-02}$	$8.5515493 \times 10^{-04}$
$-8.7547586 \times 10^{-02}$	$-5.2914361 \times 10^{-02}$	$3.1096754 \times 10^{-02}$	$-5.1350838 \times 10^{-02}$
$9.1268011 \times 10^{-02}$	$2.7131573 \times 10^{-03}$	$-4.4863302 \times 10^{-02}$	$1.4255185 \times 10^{-01}$
$-9.0394638 \times 10^{-02}$	$1.3949754 \times 10^{-02}$	$-6.9448221 \times 10^{-02}$	$-1.7933843 \times 10^{-01}$
$1.0894871 \times 10^{-01}$	$-3.1079111 \times 10^{-02}$	$3.1241715 \times 10^{-02}$	$1.0483966 \times 10^{-01}$
$8.9514996 \times 10^{-01}$	$2.4563596 \times 10^{-01}$	$3.6622936 \times 10^{-02}$	$-1.8180862 \times 10^{-02}$
$8.0315921 \times 10^{-01}$	$8.5497151 \times 10^{-01}$	$3.0216088 \times 10^{-01}$	$-2.3542808 \times 10^{-01}$
$4.4144362 \times 10^{+00}$	$4.1567018 \times 10^{-01}$	$6.5128412 \times 10^{-01}$	$-2.9517762 \times 10^{-01}$
$4.1567018 \times 10^{-01}$	$4.6760789 \times 10^{+00}$	$3.0214106 \times 10^{-01}$	$-3.3530113 \times 10^{-01}$
$6.5128412 \times 10^{-01}$	$3.0214106 \times 10^{-01}$	$5.2609029 \times 10^{+00}$	$-1.1481623 \times 10^{+00}$
$-2.9517762 \times 10^{-01}$	$-3.3530113 \times 10^{-01}$	$-1.1481623 \times 10^{+00}$	$1.4101066 \times 10^{+00}$

- columns 1 to 4 of the matrix C_2:

$1.3507833 \times 10^{+04}$	$6.7406566 \times 10^{+03}$	$3.3702384 \times 10^{+03}$	$1.6725857 \times 10^{+03}$
$6.7406566 \times 10^{+03}$	$3.3967105 \times 10^{+03}$	$1.6471350 \times 10^{+03}$	$9.4732708 \times 10^{+02}$
$3.3702384 \times 10^{+03}$	$1.6471350 \times 10^{+03}$	$1.0770710 \times 10^{+03}$	$-5.9885304 \times 10^{+01}$
$1.6725857 \times 10^{+03}$	$9.4732708 \times 10^{+02}$	$-5.9885304 \times 10^{+01}$	$1.4180254 \times 10^{+03}$
$8.4377266 \times 10^{+02}$	$3.5444149 \times 10^{+02}$	$6.2354442 \times 10^{+02}$	$-9.7398703 \times 10^{+02}$
$4.2575342 \times 10^{+02}$	$1.2333644 \times 10^{+02}$	$5.6811820 \times 10^{+02}$	$-1.0036749 \times 10^{+03}$
$1.9190201 \times 10^{+02}$	$3.3374780 \times 10^{+02}$	$-1.1273555 \times 10^{+03}$	$2.9278273 \times 10^{+03}$
$1.1160305 \times 10^{+02}$	$-3.5811160 \times 10^{+01}$	$4.7875127 \times 10^{+02}$	$-1.0848726 \times 10^{+03}$
$5.9442740 \times 10^{+01}$	$-6.5347562 \times 10^{+01}$	$4.8219628 \times 10^{+02}$	$-1.1242647 \times 10^{+03}$
$2.4762670 \times 10^{+01}$	$3.2737633 \times 10^{+01}$	$-1.0064212 \times 10^{+02}$	$2.7931117 \times 10^{+02}$
$9.6358696 \times 10^{+00}$	$4.5785273 \times 10^{+01}$	$-1.8179770 \times 10^{+02}$	$4.3287249 \times 10^{+02}$
$-8.6156343 \times 10^{+00}$	$1.1937757 \times 10^{+01}$	$-9.2227840 \times 10^{+01}$	$2.4895354 \times 10^{+02}$

- columns 5 to 8 of the matrix C_2:

$8.4377266 \times 10^{+02}$	$4.2575342 \times 10^{+02}$	$1.9190201 \times 10^{+02}$	$1.1160305 \times 10^{+02}$
$3.5444149 \times 10^{+02}$	$1.2333644 \times 10^{+02}$	$3.3374780 \times 10^{+02}$	$-3.5811160 \times 10^{+01}$
$6.2354442 \times 10^{+02}$	$5.6811820 \times 10^{+02}$	$-1.1273555 \times 10^{+03}$	$4.7875127 \times 10^{+02}$
$-9.7398703 \times 10^{+02}$	$-1.0036749 \times 10^{+03}$	$2.9278273 \times 10^{+03}$	$-1.0848726 \times 10^{+03}$
$1.1976560 \times 10^{+03}$	$6.8622449 \times 10^{+02}$	$-2.3758802 \times 10^{+03}$	$9.9112334 \times 10^{+02}$
$6.8622449 \times 10^{+02}$	$1.8212227 \times 10^{+03}$	$-3.4531339 \times 10^{+03}$	$1.0514963 \times 10^{+03}$
$-2.3758802 \times 10^{+03}$	$-3.4531339 \times 10^{+03}$	$8.1289247 \times 10^{+03}$	$-2.8111963 \times 10^{+03}$
$9.9112334 \times 10^{+02}$	$1.0514963 \times 10^{+03}$	$-2.8111963 \times 10^{+03}$	$1.0872175 \times 10^{+03}$
$9.3382078 \times 10^{+02}$	$1.3416656 \times 10^{+03}$	$-3.1661606 \times 10^{+03}$	$1.0909808 \times 10^{+03}$
$-2.5810951 \times 10^{+02}$	$-2.1640212 \times 10^{+02}$	$6.3318913 \times 10^{+02}$	$-2.5467525 \times 10^{+02}$
$-3.7001917 \times 10^{+02}$	$-4.1243999 \times 10^{+02}$	$1.0972433 \times 10^{+03}$	$-4.2184793 \times 10^{+02}$
$-2.9896513 \times 10^{+02}$	$-5.4769087 \times 10^{+01}$	$4.5207472 \times 10^{+02}$	$-2.1832809 \times 10^{+02}$

- columns 9 to 12 of the matrix C_2:

$5.9442740 \times 10^{+01}$	$2.4762670 \times 10^{+01}$	$9.6358696 \times 10^{+00}$	$-8.6156343 \times 10^{+00}$
$-6.5347562 \times 10^{+01}$	$3.2737633 \times 10^{+01}$	$4.5785273 \times 10^{+01}$	$1.1937757 \times 10^{+01}$
$4.8219628 \times 10^{+02}$	$-1.0064212 \times 10^{+02}$	$-1.8179770 \times 10^{+02}$	$-9.2227840 \times 10^{+01}$
$-1.1242647 \times 10^{+03}$	$2.7931117 \times 10^{+02}$	$4.3287249 \times 10^{+02}$	$2.4895354 \times 10^{+02}$
$9.3382078 \times 10^{+02}$	$-2.5810951 \times 10^{+02}$	$-3.7001917 \times 10^{+02}$	$-2.9896513 \times 10^{+02}$
$1.3416656 \times 10^{+03}$	$-2.1640212 \times 10^{+02}$	$-4.1243999 \times 10^{+02}$	$-5.4769087 \times 10^{+01}$
$-3.1661606 \times 10^{+03}$	$6.3318913 \times 10^{+02}$	$1.0972433 \times 10^{+03}$	$4.5207472 \times 10^{+02}$
$1.0909808 \times 10^{+03}$	$-2.5467525 \times 10^{+02}$	$-4.2184793 \times 10^{+02}$	$-2.1832809 \times 10^{+02}$
$1.2906793 \times 10^{+03}$	$-2.6413625 \times 10^{+02}$	$-4.4901402 \times 10^{+02}$	$-1.7266860 \times 10^{+02}$
$-2.6413625 \times 10^{+02}$	$1.3032741 \times 10^{+02}$	$7.7813382 \times 10^{+01}$	$6.5401511 \times 10^{+01}$
$-4.4901402 \times 10^{+02}$	$7.7813382 \times 10^{+01}$	$2.0480385 \times 10^{+02}$	$7.4193135 \times 10^{+01}$
$-1.7266860 \times 10^{+02}$	$6.5401511 \times 10^{+01}$	$7.4193135 \times 10^{+01}$	$9.0528580 \times 10^{+01}$

- columns 1 to 4 of the matrix C_3:

$7.6923077 \times 10^{+10}$	$3.5502959 \times 10^{+10}$	$1.6385981 \times 10^{+10}$	$7.5627604 \times 10^{+09}$
$3.5502959 \times 10^{+10}$	$2.0937642 \times 10^{+10}$	$1.1764294 \times 10^{+10}$	$6.3992588 \times 10^{+09}$
$1.6385981 \times 10^{+10}$	$1.1764294 \times 10^{+10}$	$7.6381725 \times 10^{+09}$	$4.6689231 \times 10^{+09}$
$7.5627604 \times 10^{+09}$	$6.3992588 \times 10^{+09}$	$4.6689231 \times 10^{+09}$	$3.1346717 \times 10^{+09}$
$3.4905048 \times 10^{+09}$	$3.4010047 \times 10^{+09}$	$2.7400807 \times 10^{+09}$	$1.9931283 \times 10^{+09}$
$1.6110022 \times 10^{+09}$	$1.7762332 \times 10^{+09}$	$1.5612211 \times 10^{+09}$	$1.2189181 \times 10^{+09}$
$7.4353947 \times 10^{+08}$	$9.1512559 \times 10^{+08}$	$8.6966246 \times 10^{+08}$	$7.2377237 \times 10^{+08}$
$3.4317184 \times 10^{+08}$	$4.6636302 \times 10^{+08}$	$4.7583767 \times 10^{+08}$	$4.1987082 \times 10^{+08}$
$1.5838253 \times 10^{+08}$	$2.3556281 \times 10^{+08}$	$2.5659013 \times 10^{+08}$	$2.3899290 \times 10^{+08}$
$7.3010173 \times 10^{+07}$	$1.1820512 \times 10^{+08}$	$1.3689702 \times 10^{+08}$	$1.3404524 \times 10^{+08}$
$3.1907869 \times 10^{+07}$	$5.8434212 \times 10^{+07}$	$7.4301607 \times 10^{+07}$	$7.8876881 \times 10^{+07}$
$-2.1055961 \times 10^{+07}$	$-3.3658883 \times 10^{+07}$	$-3.9388215 \times 10^{+07}$	$-3.9302344 \times 10^{+07}$

- columns 5 to 8 of the matrix C_3:

$3.4905048 \times 10^{+09}$	$1.6110022 \times 10^{+09}$	$7.4353947 \times 10^{+08}$	$3.4317184 \times 10^{+08}$
$3.4010047 \times 10^{+09}$	$1.7762332 \times 10^{+09}$	$9.1512559 \times 10^{+08}$	$4.6636302 \times 10^{+08}$
$2.7400807 \times 10^{+09}$	$1.5612211 \times 10^{+09}$	$8.6966246 \times 10^{+08}$	$4.7583767 \times 10^{+08}$
$1.9931283 \times 10^{+09}$	$1.2189181 \times 10^{+09}$	$7.2377237 \times 10^{+08}$	$4.1987082 \times 10^{+08}$
$1.3575536 \times 10^{+09}$	$8.8250564 \times 10^{+08}$	$5.5383334 \times 10^{+08}$	$3.3805229 \times 10^{+08}$
$8.8250564 \times 10^{+08}$	$6.0576622 \times 10^{+08}$	$3.9942974 \times 10^{+08}$	$2.5517254 \times 10^{+08}$
$5.5383334 \times 10^{+08}$	$3.9942974 \times 10^{+08}$	$2.7550985 \times 10^{+08}$	$1.8348065 \times 10^{+08}$
$3.3805229 \times 10^{+08}$	$2.5517254 \times 10^{+08}$	$1.8348065 \times 10^{+08}$	$1.2698581 \times 10^{+08}$
$2.0171595 \times 10^{+08}$	$1.5885399 \times 10^{+08}$	$1.1874888 \times 10^{+08}$	$8.5201704 \times 10^{+07}$
$1.1806298 \times 10^{+08}$	$9.6486512 \times 10^{+07}$	$7.4506549 \times 10^{+07}$	$5.5014582 \times 10^{+07}$
$7.4787171 \times 10^{+07}$	$6.5505299 \times 10^{+07}$	$5.4060107 \times 10^{+07}$	$4.2560333 \times 10^{+07}$
$-3.5445851 \times 10^{+07}$	$-2.9762013 \times 10^{+07}$	$-2.3675810 \times 10^{+07}$	$-1.8044559 \times 10^{+07}$

- columns 9 to 12 of the matrix C_3:

$1.5838253 \times 10^{+08}$	$7.3010173 \times 10^{+07}$	$3.1907869 \times 10^{+07}$	$-2.1055961 \times 10^{+07}$
$2.3556281 \times 10^{+08}$	$1.1820512 \times 10^{+08}$	$5.8434212 \times 10^{+07}$	$-3.3658883 \times 10^{+07}$
$2.5659013 \times 10^{+08}$	$1.3689702 \times 10^{+08}$	$7.4301607 \times 10^{+07}$	$-3.9388215 \times 10^{+07}$
$2.3899290 \times 10^{+08}$	$1.3404524 \times 10^{+08}$	$7.8876881 \times 10^{+07}$	$-3.9302344 \times 10^{+07}$
$2.0171595 \times 10^{+08}$	$1.1806298 \times 10^{+08}$	$7.4787171 \times 10^{+07}$	$-3.5445851 \times 10^{+07}$
$1.5885399 \times 10^{+08}$	$9.6486512 \times 10^{+07}$	$6.5505299 \times 10^{+07}$	$-2.9762013 \times 10^{+07}$
$1.1874888 \times 10^{+08}$	$7.4506549 \times 10^{+07}$	$5.4060107 \times 10^{+07}$	$-2.3675810 \times 10^{+07}$
$8.5201704 \times 10^{+07}$	$5.5014582 \times 10^{+07}$	$4.2560333 \times 10^{+07}$	$-1.8044559 \times 10^{+07}$
$5.9198495 \times 10^{+07}$	$3.9143693 \times 10^{+07}$	$3.2441978 \times 10^{+07}$	$-1.3332763 \times 10^{+07}$
$3.9143693 \times 10^{+07}$	$2.6454096 \times 10^{+07}$	$2.2554175 \times 10^{+07}$	$-9.1344475 \times 10^{+06}$
$3.2441978 \times 10^{+07}$	$2.2554175 \times 10^{+07}$	$2.2706317 \times 10^{+07}$	$-8.6560406 \times 10^{+06}$
$-1.3332763 \times 10^{+07}$	$-9.1344475 \times 10^{+06}$	$-8.6560406 \times 10^{+06}$	$3.3733471 \times 10^{+06}$

B Target functions for the shape reconstruction problem

The target functions used in the shape reconstruction test case are given by the general Bézier curve formulation:

$$F_0(t) = \sum_{k=0}^{n_0-1} B^k_{n_0-1}(t) x_{0k+1}$$

Two functions $F_{01}(t)$ and $F_{02}(t)$ are considered in this study:

- the function $F_{01}(t)$ is given by:

$$F_{01}(t) = \sum_{k=0}^{13} B^k_{13}(t) x_{0k+1}$$

where the vector of the control points X_{01} is:

$$X_{01} = (0.1, 0.5, 0.1, 0.1, 0.1, 0.1, 1.5, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 0.1)^T$$

- the function $F_{02}(t)$ is given by:

$$F_{02}(t) = \sum_{k=0}^{17} B_{17}^k(t) x_{0k+1}$$

where the vector of the control points X_{02} is:

$$X_{02} = (1, 4, -3, 3, 3, -3, 4, -1, 4, 4, -1, 4, -3, 3, 3, -3, 4, 1)^T$$

References

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

[2] A. Borzì. Multilevel methods for optimization and inverse problems. In *SIAM Annual Meeting, Minisymposium*, Boston, http://www.uni-graz.at/imawww/borzi/, 2006.

[3] D. Büche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 35 (2):183–194, 2005.

[4] J. E. Dennis and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1 (4):448–474, 1991.

[5] J. A. Désidéri. Two-level ideal algorithm for shape optimization. In *International Conference on Advances in Numerical Mathematics*, Moscow, September 2005.

[6] J. A. Désidéri, B. Abou El Majd, and A. Janka. Nested and self-adaptative bézier parameterization for shape optimization. In *International Conference on Control, Partial Differential Equations and Scientific Computing*, Beijing, China, September 2004.

[7] R. Duvigneau. Introduction aux méthodes d'optimisation sans gradient pour l'optimisation et le contrôle en mécanique des fluides. In *Optimisation et Contrôle des écoulements et des Transferts*. Ecole de printemps OCET, March 2006.

[8] B. Abou El Majd, R. Duvigneau, and J. A. Désidéri. Aerodynamic shape optimization using a full adaptative multilevel algorithm. In *ERCOFTAC Conference Design Optimization : Methods and Applications*, Canary Island, Spain, April 2006.

[9] K. C. Giannakoglou and M. K. Karakasis. Hierarchical and distributed metamodel-assisted evolutionary algorithms. In *Introduction to Optimization and Multidisciplinary Design*. von Karman Institue of Fluid Dynamics, Lecture Series, March 2006.

[10] A. Giunta, J. Dudley, R. Narducci, B. Grossman, R. Haftka, W. Mason, and L. Watson. Noisy aerodynamic response and smooth approxinations in HSCT design. *AIAA Paper*, 94-4316, 1994.

[11] A. Jameson and L. Martinelli. Optimum aerodynamic design using the navier-stokes equation. *Theorical and Computaional Fluid Dynamics*, 10:213–237, 1998.

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

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

[14] R. Lewis and S. Nash. A multigrid approach to the optimization of systems governed by differential equations. *American Institute of Aeronautics and Astronautics, Reston*, VA, AIAA-2000-4890, 2000.

[15] R. Lewis and S. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM J. Sci. Comput.*, 26:1811–1837, 2005.

[16] J. A Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

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

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