A Semi-Deterministic Global Optimization Method. Application to a Control Problem of the Burgers Equation, comparing with Other Methods.

Benjamin Ivorra [†], Angel Manuel Ramos[‡] and Bijan Mohammadi [†]

[†] Mathematics and Modeling Institute, UM II
cc 051 Place Eugène Bataillon, 34095 Montpellier, France

[‡] Departamento de Matemática Aplicada, Facultad de Ciencias Matemáticas
UCM, Plaza de Ciencias n^o 3, Madrid, Spain

Short title: *A semi-deterministic global optimization method*

Abstract: *This paper aims two objectives. First, we introduce a new global optimization algorithm reformulating optimization problems in terms of boundary value problems. Then we apply this algorithm to a pointwise control problem of the viscous Burgers equation, where the control weight coefficient is progressively decreased. Results are compared with those obtained with a genetic and a L-BFGS algorithms in order to see the efficiency of our method and the necessity to use global optimization techniques.*

Keywords: Global optimization, semi-deterministic algorithm, genetic algorithm, Optimal control problem, Burgers equation.

AMS subject classifications: 65K10, 80M50, 49J20, 35Q53.

1 INTRODUCTION

Many optimization algorithms can be viewed as discrete forms of Cauchy problems for a system of ordinary differential equations in the space of control parameters [1, 2]. We will see that if one introduces an extra information on the infimum, solving global optimization problems using these algorithms is equivalent to solving Boundary Value Problems (**BVP**) for the same equations. A motivating idea is therefore to apply algorithms solving BVPs to perform this global optimization.

In this paper, we introduce and apply a particular algorithm, issued from our BVP analysis, to a pointwise control problem of the viscous burgers equation. This equation retains many of the interesting features of the Navier-Stokes equations and can be used for the modeling of weak shock waves when the flow of interest is a perturbation of a uniform sonic gas flow [3]. The weight coefficient value of the control part of the equation, denoted α, is progressively decreased in order to see the effect on the optimized solution.

Obtained results are then compared with those given by:

- A genetic algorithm (**GA**) in order to perform a comparison with a current popular method [4, 5].
- A Limited-memory Broyden-Fletcher-Goldfarb-Shanno (**L-BFGS**) algorithm [1, 6] in order to see, depending on the previous value of α, the necessity to use or not global optimization methods.

Section 2 presents our optimization methods and a short related mathematical background. In Section 3, we introduce our general pointwise control problem and intend to solve it, with various values of α, using the mentioned approaches.

2 Optimization methods

We present here a short introduction of each optimization method used to solve the problems considered in section 3.

We consider a function $J : \Omega \to \mathbb{R}$ to be minimized, where the optimization parameter x belongs to an admissible set $\Omega \subset R^N$. We make the following assumptions on the functional [1]: $J \in C^1(\Omega, \mathbb{R})$, it is coercive and the minimum of J in Ω is denoted by J_m.

2.1 Semi-Deterministic Algorithms (SDA)

Many minimization algorithms which perform the minimization of J can be seen as discretizations of continuous dynamical systems with associated initial conditions [1, 2].

A numerical global optimization of J with one of those algorithms, called here *core optimization method*, is possible if the following BVP has a solution:

$$\begin{cases} \text{First or second order Initial Value Problem} \\ |J(x(Z)) - J_m| < \epsilon \end{cases} \tag{1}$$

where $x(Z)$ is the solution of the considered dynamical system found at a given finite time $Z \in \mathbb{R}$ and ϵ the approximation precision. In practice, when

J_m is unknown, we set J_m to a lower value (for example $J_m = 0$ for a non-negative function J) and look for the best solution for a given complexity and computational effort.

This BVP is over-determined as it includes more conditions than derivatives. This over determination can be removed for instance by considering one of the initial conditions in the considered dynamical system as a new variable denoted by v. Then we could use what is known on BVP theory, for example a shooting method [1],in order to determine a suitable v solving (1).

In order to illustrate and implement previous methodology, we consider in this paper a steepest descent method, as core optimization method, which come from the discretization of the following dynamical system:

$$\begin{cases} \dfrac{dx(\zeta)}{d\zeta} = -\nabla J(x(\zeta)) \\ x(0) = x_0 \end{cases} \tag{2}$$

Thus BVP (1) can be rewritten as:

$$\begin{cases} \dfrac{dx(\zeta)}{d\zeta} = -\nabla J(x(\zeta)) \\ x(0) = x_0 \\ |J(x(Z)) - J_m| < \epsilon \end{cases} \tag{3}$$

A discrete form of the considered steepest descent algorithm with an output called $D(x_0, I, \epsilon)$ is given by:

```
• Input: $x_0, I, \epsilon$
• $x^1 = x_0$
For $n$ going from 1 to $I$
    • Determine $\rho_{\text{opt}} = \text{argmin}_\rho(J(x^n - \rho\nabla J(x^n)))$ using a dichotomy algorithm (see [1])
    • $x^{n+1} = x^n - \rho_{\text{opt}}\nabla J(x^n)$
    • If $J(x^{n+1}) < J_m + \epsilon$ EndFor
EndFor
• Output: $D(x_0, I, \epsilon) = x^{n+1}$
```

The inputs $x_0 \in \Omega$, $\epsilon \in \mathbb{R}$ and $I \in \mathbb{N}$ are respectively the initial condition, the stopping criterion and the iteration number.

In this case, BVP (1) is solved if the initial condition x_0 lies in the attraction basin of the global minimum of J and I is large enough. In order to determine such an initial condition, we consider $x_0 = v$ as a new variable in the previous algorithm to be found by the minimization of:

$$h(v) = J(D(v, I, \epsilon)) - J_m \tag{4}$$

To perform the minimization of (4), we then consider the following algorithm with an output called $A_1(v_1, N, I, \epsilon)$:

```
• **Input:** $v_1, N, I, \epsilon$
• $v_2$ chosen randomly
**For** $i$ going from 1 to $N$
  • $o_i = D(v_i, I, \epsilon)$
  • $o_{i+1} = D(v_i + 1, I, \epsilon)$
  • **If** $J(o_i) = J(o_{i+1})$ **EndFor**
  • **If** $\min\{J(o_k), k = 1, ..., i\} < J_m + \epsilon$ **EndFor**
  • $v_{i+2} = v_{i+1} - J(o_{i+1})\frac{v_{i+1}-v_i}{J(o_{i+1})-J(o_i)}$
**EndFor**
• **Output:** $A_1(v_1, N, I, \epsilon) = \text{argmin}\{J(o_k), k = 1, ..., i\}$
```

The inputs are $v_1 \in \Omega$, $(N, I) \in \mathbb{N}^2$ and $\epsilon \in \mathbb{R}$.

This line search minimization algorithm might fail. An external level to the algorithm A_1 is added in order to have a multidimensional search. As previously, we consider $v_1 = w$ as a new variable in A_1 to be found by the minimization of:

$$\tilde{h}(w) = h(A_1(w, N, I, \epsilon)) \tag{5}$$

To perform the minimization of (5), we then consider the following two-level algorithm with an output called $A_2(w_1, M, N, I, \epsilon)$:

```
• **Input:** $w_1, M, N, I, \epsilon$
• $w_2$ chosen randomly
**For** $i$ going from 1 to $M$
  • $p_i = A_1(w_i, N, I, \epsilon)$
  • $p_{i+1} = A_1(w_i + 1, N, I, \epsilon)$
  • **If** $J(p_i) = J(p_{i+1})$ **EndFor**
  • **If** $\min\{J(p_k), k = 1, ..., i\} < J_m + \epsilon$ **EndFor**
  • $w_{i+2} = w_{i+1} - J(p_{i+1})\frac{w_{i+1}-w_i}{J(p_{i+1})-J(p_i)}$
**EndFor**
• **Output:** $A_2(w_1, M, N, I, \epsilon) = \text{argmin}\{J(p_k), k = 1, ..., i\}$
```

Here $w_1 \in \Omega$, $(M, N, I) \in \mathbb{N}^3$ and $\epsilon \in \mathbb{R}$. In order to add search directions, the previous construction can be easily pursued recursively.

The choice of the initial condition w_2 in this algorithm contains the only non-deterministic feature of the SDA method. In practice we also randomly choose the initial condition $w_1 \in \Omega$.

A mathematical background for this approach as well as a validation on academic test cases or on problems including solutions of nonlinear partial differential equations are available [7, 1, 8, 9, 10].

2.2 Genetic Algorithms (GA)

Genetic algorithms approximate the global minimum of J, called in this case fitness function, through a stochastic process based on an analogy with the Darwinian evolution of species [4]: a first family, called 'population', $X^0 = \{x_l^0 \in \Omega, l = 1, ..., N_p\}$ of N_p possible solutions of the optimization problem, called 'individuals', is randomly generated in the search space Ω. Starting from this population, we build recursively N_{gen} new populations, called generations, $X^i = \{x_l^i \in \Omega, l = 1, ..., N_p\}$ with $i = 1, .., N_{\text{gen}}$ through three stochastic steps, called selection, crossover and mutation.

More precisely we present here a non usual matrix-form approach for GA:

We first rewrite X^n using the following (N_p, N)-real valued matrix form:

$$X^i = \begin{bmatrix} x_1^i(1) & \dots & x_1^i(N) \\ \vdots & \ddots & \vdots \\ x_{N_p}^i(1) & \dots & x_{N_p}^i(N) \end{bmatrix} \tag{6}$$

Selection: Each individual, x_l^i is ranked with respect to its fitness value $J(x_l^i)$ (i.e. lower is the fitness value better is the ranking). Then N_p individuals are randomly selected (individuals with better ranking have higher chances to be selected), with eventual repetitions, to become 'parents'.

Introducing $\mathcal{S}^i$ a binary (N_p, N_p)-matrix, generated according to previous ranking and selection processes, with $\mathcal{S}_{j,k}^i = 1$ if the kth individual of X^i is the 'parent' selection number j and $\mathcal{S}_{j,k}^i = 0$ if not. We define:

$$X^{i+1/3} = \mathcal{S}^i X^i. \tag{7}$$

Crossover: This process leads to a data exchange between two 'parents' and the apparition of two new individuals called 'children'. We determine, with a probability p_c, if two consecutive parents in $X^{i+1/3}$ should exchange data or if they are directly copied into the intermediate population $X^{i+2/3}$.

Introduce $\mathcal{C}^i$ a real-valued (N_p, N_p)-matrix where for each couple of consecutive lines $(2j-1, 2j)$ ($1 \leq j \leq \frac{N_p}{2}$ in case N_p is a even number or $1 \leq j \leq \frac{N_p - 1}{2}$ in case N_p is a odd number), the coefficients of the $2j-1$th and $2j$th rows are given by:

$$\mathcal{C}_{2j-1,2j-1}^i = \lambda_1, \;\; \mathcal{C}_{2j-1,2j}^i = 1 - \lambda_1, \;\; \mathcal{C}_{2j,2j-1}^i = \lambda_2, \;\; \mathcal{C}_{2j,2j}^i = 1 - \lambda_2$$

in this expression:

- $\lambda_1 = \lambda_2 = 1$ if parents are directly copied (with a probability $1 - p_c$).
- λ_1 and λ_2 are randomly chosen in $]0, 1[$ if a data exchange occurs between the two parents (with a probability p_c).

Other coefficients of $\mathcal{C}^i$ are set to 0. If N_p is a odd number, the N_pth parent is directly copied, i.e $\mathcal{C}^i_{N_p,N_p} = 1$.

This step can be summarized as:

$$X^{i+2/3} = \mathcal{C}^i X^{i+1/3} \tag{8}$$

Mutation: This process leads to new parameters values for some individuals of the population. More precisely, each children is modified (or mutated) with a fixed probability p_m.

Introduce for instance a random perturbation matrix $\mathcal{E}^i$ with a i-th line equals to:

- a random vector $\epsilon_i \in \mathbb{R}^N$, according to the admissible space Ω, if a mutation is applied to the ith children (with a probability p_m).
- 0 if no mutation is applied to the ith children (with a probability 1-p_m).

This step can then take the following form:

$$X^{i+1} = X^{i+2/3} + \mathcal{E}^i \tag{9}$$

Therefore, the new population can be written as:

$$X^{i+1} = \mathcal{C}^i \mathcal{S}^i X^i + \mathcal{E}^i \tag{10}$$

With these three basic evolution processes, it is generally observed that the best obtained individual is getting closer after each generation to the optimal solution of the problem [4].

Engineers like GAs because these algorithms do not require sensitivity computation, perform global and multi-objective optimization and are easy to parallelize. However, their drawbacks remain their weak mathematical background, their computational complexity, their slow convergence and their lack of accuracy.

As a fine convergence is difficult to achieve with GA based algorithms, it is recommended when it is possible, to complete the GA iterations by a descent method. This is especially useful when the functional is flat around the infimum [5]. The hybridization between both methods is usually called Hybrid Genetic Algorithm (HGA).

2.3 Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS)

This is a modification of the BFGS method [11]. This deterministic method is based on a limited-memory quasi-Newton algorithm well suited for large-scale bound-constrained or unconstrained optimization. It is out of the scope of this paper but the interested reader can find more details in literature (See, for instance, [1, 12]).

3 Application to Pointwise Control Problems

We consider the following viscous Burgers equation:

$$y_t - \nu y_{xx} + y y_x = f \qquad \text{in } Q = (0,1) \times (0,T) \tag{11}$$

where $T \in]0, \infty)$ is an horizon time, $\nu > 0$ is a viscosity parameter and f is a density of external forces.

We look for a control $v(t)$ forcing the solution at a point $a \in [0,1]$, completing the equation with the initial and boundary conditions used in [13] (and in other references cited here):

$$\left\{ \begin{array}{ll} y_t - \upsilon y_{xx} + y y_x = f + v\delta(x-a) & \text{in Q} \\ y_x(0,t) = 0, \ y(1,t) = 0 & \text{for } t \in [0,T] \\ y(0) = y_0 & \text{for } t \in [0,1] \end{array} \right. \tag{12}$$

where $\delta(x-a)$ denotes the Dirac measure at point a.

A variational formulation of the above system (12) is provided by $y(t) \in L^2(0,T;V_0) \cap H^1(0,T;V_0')$, such that $y(0) = y_0$ and:

$$\left\{ \begin{array}{l} \int_0^T < y_t, z >_{V_0' \times V_0} dt + \nu \int_0^T (y_x, z_x) dt + \int_0^T (y y_x, z) dt \\ \qquad\qquad\qquad = \int_0^T (f,z) dt + \int_0^T v z(a) dt \qquad\qquad \forall \ z \in L^2(0,T;V_0), \end{array} \right. \tag{13}$$

where $V_0 = \{z \in H^1(0,1) : z(1) = 0\}$ and $(\cdot,\cdot)$ denotes the scalar product in $L^2(0,1)$ defined by $(y,z) = \displaystyle\int_0^1 yz dx$.

3.1 Problem Formulation

Let us consider a target function $y_T \in L^2(0,1)$. We define the bounded control space as $\mathcal{U} = \{f \in L^2(0,T) : \text{for all } x \in [0,T] \ \ |f(x)| \leq S_{\mathrm{b}} \}$, where $S_{\mathrm{b}} \in \mathbb{R}$. The objective is to find a control $u \in \mathcal{U}$, forcing the solution at the point a, such that $y(T)$ is close to y_T in $(0,1)$ at a minimal cost (norm) for the control. To do this, we define the cost function J by:

$$J(v) = \alpha \parallel v \parallel_{\mathcal{U}}^2 + \parallel y(T) - y_T \parallel_{L^2(0,1)}^2 \tag{14}$$

where $\alpha \geq 0$.

Then, we define the optimal control problem $(\mathcal{CP})$ as:

$$\begin{cases} \text{Find } u \in \mathcal{U} \text{ such that:} \\ J(u) = \min_{v \in \mathcal{U}} J(v) \end{cases}$$

When $\alpha = 0$ there is no proof for existence of solution for $(\mathcal{CP})$. In this case, we are interested by finding a solution as close as possible to the infimum of the function.

In order to solve numerically $(\mathcal{CP})$ we need to define an associated discrete problem. Following [13] we consider:

- A time discretization step Δt, defined by $\Delta t = T/L$, where L is a positive integer. Then, if $t^l = l\Delta t$, we have $0 < t^1 < t^2 < \cdots < t^L = T$.
- A space discretization step h, defined by $h = 1/I$, where I is a positive integer. Then, if $x_i = (i-1)h$, we have $0 = x_1 < x_2 < \cdots < x_I < x_{I+1} = 1$.

Then we approximate V_0 by

$$V_{0h} = \{z \in \mathcal{C}^0[0,1] : z(1) = 0, z|_{(x_i,x_{i+1})} \in P_1, i = 1, \cdots, I\},$$

where P_1 is the space of the polynomials of degree ≤ 1. We define a_h and b_h by

$$a_h(y,z) = \int_0^1 y_x z_x dx, \qquad b_h(w,y,z) = \int_0^1 w y_x z dx.$$

Thus the discrete problem associated to $(\mathcal{CP})$ is defined by the following finite-dimensional minimization problem $(\mathcal{CP})_h^{\Delta t}$:

$$\begin{cases} \text{Find } u_h^{\Delta t} = \{u^l\}_{l=1\cdots L} \in \mathcal{U}^{\Delta t} \text{ such that:} \\ J_h^{\Delta t}(u_h^{\Delta t}) \leq J_h^{\Delta t}(v), \ \forall v = \{v^l\}_{l=1\cdots L} \in \mathcal{U}^{\Delta t}, \end{cases}$$

where the discrete control space $\mathcal{U}^{\Delta t}$ in $(\mathcal{CP})_h^{\Delta t}$ is initially equal to $[-S_\mathrm{b}\ , S_\mathrm{b}\ ]^L$ and

$$J_h^{\Delta t}(v) = \alpha \Delta t \sum_{l=1}^{L} |v^l|^2 + l\left((1-\theta) \parallel y_h^{L-1} - y_T \parallel_{L^2(0,1)}^2 + \theta \parallel y_h^L - y_T \parallel_{L^2(0,1)}^2\right)$$

with $\theta \in (0,1]$ y^L defined from the solution of the following full discretization of the Burgers equation (1):

$$\begin{cases} y_h^l \in V_{0h},\ l = 0, ..., L \text{ such that, } \forall z \in V_{0h}: \\ (y_h^0, z) = (y_0, z) \\ (\frac{y_h^1 - y_h^0}{\Delta t}, z) + \nu a_h(\frac{2}{3}y_h^1 + \frac{1}{3}y_h^0, z) + b_h(y_h^0, y_h^0, z) = (f^1, z) + \frac{2}{3}v^l z(a) \\ for \ \ l \geq 2: \\ (\frac{\frac{3}{2}y_h^l - 2y_h^{l-1} + \frac{1}{2}y_h^{l-2}}{\Delta t}, z) + \nu a_h(y_h^l, z) + b_h(2y_h^{l-1} - y_h^{l-2}, 2y_h^{l-1} - y_h^{l-2}, z) \\ \qquad\qquad = (f^l, z) + v^l z(a). \end{cases} \quad (15)$$

In order to keep a certain regularity in the computed controls and also to reduce the degrees of freedom, we generate them by spline interpolation [1] through a number of $N_S = 8$ points included in $[-S_{\rm b}\,, S_{\rm b}\,]$. Thus the new control space becomes $\mathcal{U}^{\Delta t}_{N_S} = [-S_{\rm b}\,, S_{\rm b}\,]^{N_S}$ and we denote by $(\mathcal{CP})^{\Delta t}_{h,N_S,\alpha}$ the new associated minimization problem. For each control $u \in \mathcal{U}^{\Delta t}_{N_S}$, we obtain an associated $\overline{u} \in \mathcal{U}^{\Delta t}$, which is used in the discrete state equation.

3.2 Results

We consider the test problem defined as follows: $a = 0.5$, $T = 1$, $I = 128$, $N = 1500$, $\nu = 10^{-2}$, $S_{\rm b}\, = 20$, α is user defined, $y_0 = 0$ and

$$f(x,t) = \begin{cases} 1 & \text{if } (x,t) \in (0,1/2) \times (0,T), \\ 2(1-x) & \text{if } (x,t) \in [1/2,1) \times (0,T), \end{cases}$$

We construct the target solution y_T using the predefined control $u_T(t) = 9{+}\sin(t * 0.2 * \pi)$, $t \in [0,1]$. Figure 1 shows u_T and the associated target solution $y_T = y(T; u_T)$. We point that u_T does not belong to $\mathcal{U}^{\Delta t}_{N_S}$ defined previously.

In order to solve numerically $(\mathcal{CP})^{\Delta t}_{h,N_S,\alpha}$ with α set to 0, 0.01, 0.1 and 1, respectively, we will use the three different optimization methods presented previously:

- L-BFGS algorithm is applied with the same configuration as the one used in [13]. It starts from a null control.
- The two-level SDA algorithm A_2 is used with $w_1 = 0$ and $(M, N, I, \epsilon) = (5, 5, 10, 1.e^{-6})$. These values give a good compromise between computation complexity and result accuracy (see [1, 10, 9]). As the value of J_m is unknown and $J_h^{\Delta t}$ is a non-negative function, we set $J_m = 0$.
- HGA is applied with the following values for the three associated stochastic processes (see section 2.2):
  - The population size has been set to $N_p = 180$ and the generation number to $N_{\rm gen} = 30$.
  - The selection is a roulette wheel type [4] proportional to the rank of the individual in the population.
  - The crossover is barycentric in each coordinate with a probability of $p_c = 0.45$.
  - The mutation process is non-uniform with a probability of $p_m = 0.15$.
  - A one-elitism principle, that consists in keeping the current best individual in the next generation, has also been imposed.

  - At the end of the algorithm, in order to improve the result accuracy, we perform a steepest descent method starting from the optimized result.

  This set of parameters have been previously applied with success in order to solve various complex engineering problems [5, 14].

In all algorithms, points already computed are stocked in memory in order to avoid recomputations. This technique is useful in particular for SDA and HGA techniques. Indeed, in HGA each individual can be present and repeated in various generations. In SDA, if two consecutive initial conditions give the same minimum, dynamical system trajectory is projected on the maximum or minimum boundary border. This can occur various time during the optimization process.

All these parameters are fixed and used in all computations of this paper. Optimization algorithms are run on a 3Ghz PC with 512 Mb Memory. One functional evaluation takes around 4 seconds (real-time).

For each algorithm, optimized controls and associated solutions are presented in Figure 2. Optimization data are summarized in Table 1. Convergence histories for SDA and HGA are shown in Figure 3. As it can be seen on Table 1, as α decreases the iteration number of SDA increases. This is due to the de-convexification of the cost function resulting on the apparition of local minima and thus on the diminution of recomputed points during SDA optimization process.

As we can observe on Figure 2 and Table 1:

- For $\alpha = 1$, SDA, HGA and L-BFGS lead to the same result.
- For $\alpha = 0.1$ and $\alpha = 0.01$, SDA cost function value is lower than HGA and L-BFGS ones. Difference between SDA and L-BFGS is short. However, SDA results are better in term of optimal control. This is visible on Table 1 on respective control values.
- For $\alpha = 0$, SDA and HGA over-perform the L-BFGS algorithm. SDA is faster than HGA and found a better solution.

As we can observe, for large values of α the use of a global optimization methods seems to be not necessary. On the other hand, for small values of α, the choice of a global method seems a better option because the problem becomes highly non-linear. In addition, in all cases, SDA has given better results, and in a lower time, than HGA.

4 Conclusions

A new class of optimization method has been introduced. A particular algorithm, based on the steepest descent algorithm, has been presented and applied to a pointwise control problem of the burgers equation.

Results have been compared to those obtained with a L-BFGS and a Hybrid GA method. The use of the global method has been justified for small balances of the control norm. Otherwise, a local method is efficient. This is an intuitive result as the control term increases the convexity of the considered cost function.

The presented algorithm has also been validated on several other numerical examples involving local minima. The algorithm provides an affordable approach for the solution of problems where no classical method is available [7, 1, 9, 10, 15, 8, 16, 17].

Acknowledgment: A.M. Ramos was supported by the 'Plan Nacional de I+D+I (2000-2003)' of the MCyT (Spain), through the AGL2003-06862-C02-02 project, and by the "Dirección General de Universidades e Investigación de la Consejería de Educación de la Comunidad de Madrid y de la Universidad Complutense de Madrid", through the UCM Research Group–Ref. 910480.

References

[1] B. Mohammadi and J-H. Saiac. *Pratique de la simulation numérique.* Dunod, 2002.

[2] H. Attouch and R. Cominetti. A dynamical approach to convex minimization coupling approximation with the steepest descent method. *Journal of Differential Equations*, 128(2):519–540, 1996.

[3] B. K. Shivamoggi. *Theoretical Fluid Dynamics.* Martinus Nijhoff Publishers, 1985.

[4] D. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[5] L. Dumas, V. Herbert, and F. Muyl. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers and Fluids*, 33(5):849–858, 2004.

[6] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large-scale optimization. *Mathematical Programming*, 45:503–528, 1995.

[7] B. Ivorra. Semi-deterministic global optimization. *PhD. University of Montpellier 2*, 2006.

[8] L. Debiane, B. Ivorra, B. Mohammadi, F. Nicoud, A. Ern, T. Poinsot, and H. Pitsch. A low-complexity global optimization algorithm for temperature and pollution control in flames with complex chemistry. *International Journal of Computational Fluid Dynamics*, Accepted, to be published in 2006.

[9] B. Ivorra, B. Mohammadi, and D.E. Santiago, J.G.and Hertzog. Semi-deterministic and genetic algorithms for global optimization of microfluidic protein folding devices. *International Journal of Numerical Method in Engineering*, 66:319–333, 2006.

[10] B. Ivorra, B. Mohammadi, L. Dumas, O. Durand, and P. Redont. Semi-deterministic vs. genetic algorithms for global optimization of multichannel optical filters. *International Journal of Computational Science for Engineering*, Accepted, to be published in 2006.

[11] C. G. Broyden, R. Fletcher, D. Goldfarb, and D. F. Shanno. Bfgs method. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[12] J. Liu, D.C.and Nocedal. On the limited memory bfgs method for large-scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[13] A. M. Ramos, R. Glowinski, and J. Periaux. Pointwise control of the burgers equation and related nash equilibrium problems: Computational approach. *Journal of optimization theory and applications*, 112(3):499–516, 2002.

[14] J. Skaar and K.M. Risvik. A genetic algorithm for the inverse problem in synthesis of fiber gratings. *Journal of Lightwave Technology*, 16(10):1928–1932, 1998.

[15] D. Isebe, B. Mohammadi, P. Azerad, B. Ivorra, and F. Bouchette. Optimal shape design of coastal structures. *Proceeding of Siam on Mathematical & Computational Issues in the Geosciences*, 2005.

[16] B. Mohammadi and O. Pironneau. *Applied Shape Optimization for Fluids.* Oxford University Press, 2001.

[17] B. Mohammadi and O. Pironneau. Shape optimization in fluid mechanics. *Annual Review of Fluid Mechanics.*, 36:255–279, 2004.

	L-BFGS	HGA	SDA
$\alpha = 1$			
Final Cost Function Value	6.6	6.6	6.6
Iteration Number	200	4000	600
Computational Time (mn)	13	266	42
$\alpha = 0.1$			
Final Cost Function Value	.95	1.1	.94
Control Value	.4	.44	.37
Iteration Number	200	4000	1200
Computational Time (mn)	13	266	80
$\alpha = 0.01$			
Final Cost Function Value	.15	.16	.14
Control Value	.045	.049	.04
Iteration Number	200	4000	2000
Computational Time (mn)	13	266	133
$\alpha = 0$			
Final Cost Function Value	3×10^{-2}	8×10^{-3}	5×10^{-3}
Iteration Number	200	4000	2000
Computational Time (mn)	13	266	133

Table 1: Optimization data: From **Top** to **Bottom** $\alpha = 1, \alpha = 0.1, \alpha = 0.01$, and $\alpha = 0$. Cost function values, control value in the cost function (only for $\alpha = 0.1$ and $\alpha = 0.01$), Iteration number and computational time (real-time in minutes) after optimization algorithm are reported.

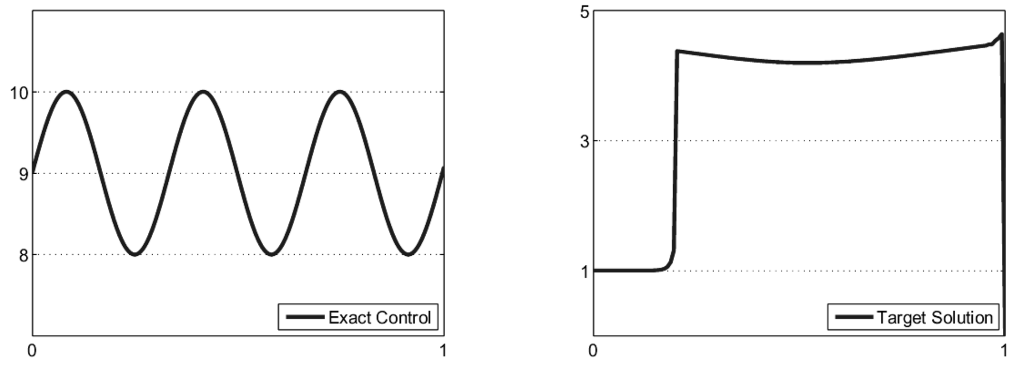

Figure 1: **Left**: Predefined control $u_T(t) = 9 + \sin(t * 0.2 * \pi), t \in [0, 1]$. **Right**: Associated target solution y_T.

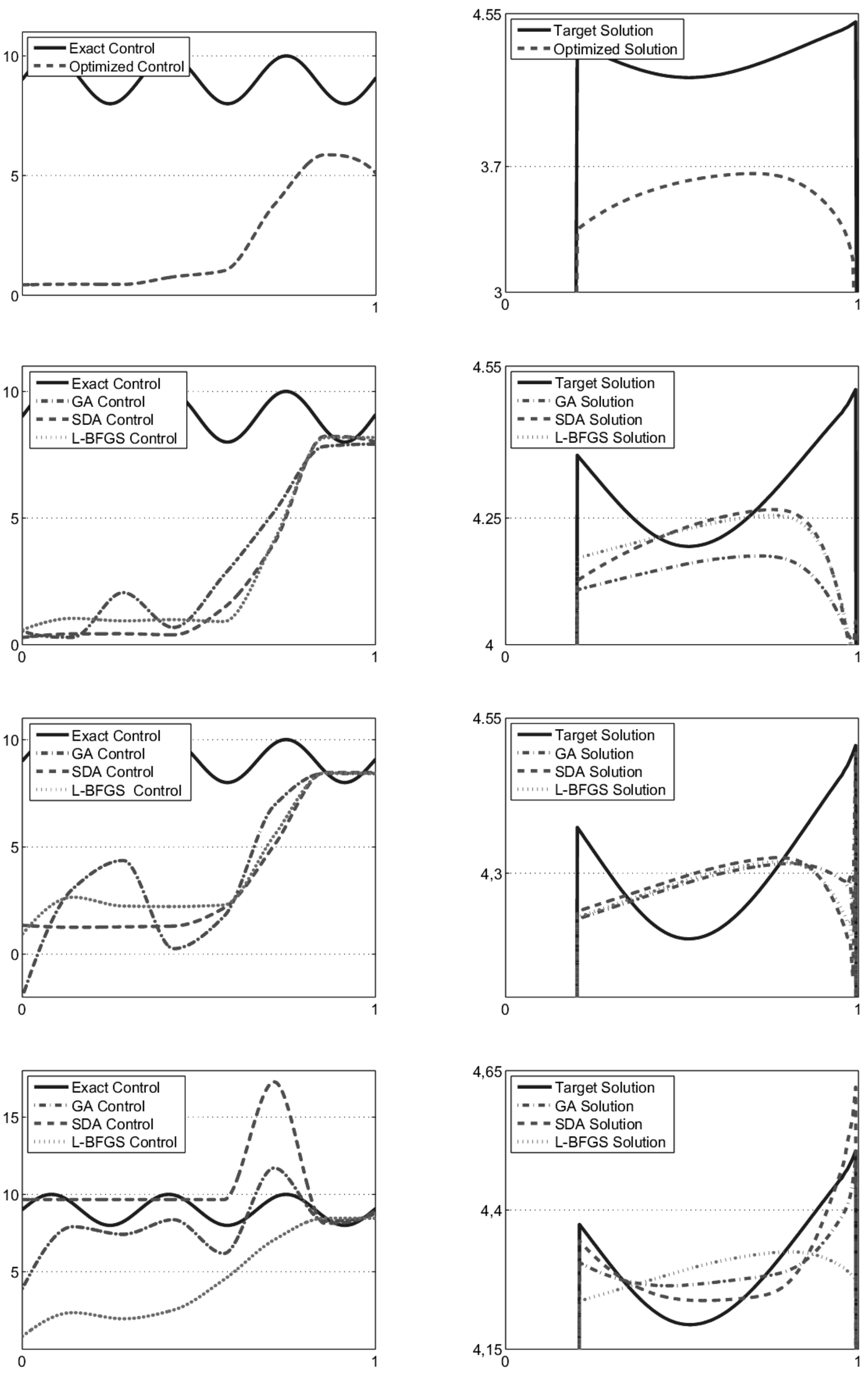

Figure 2: From (**Top**) to (**Bottom**): Results for $\alpha = 1$, $\alpha = 0.1$, $\alpha = 0.01$ and $\alpha = 0$. (**Left**) Optimized control. (**Right**) Zoom on the three optimized solutions in order to precise visual differences.

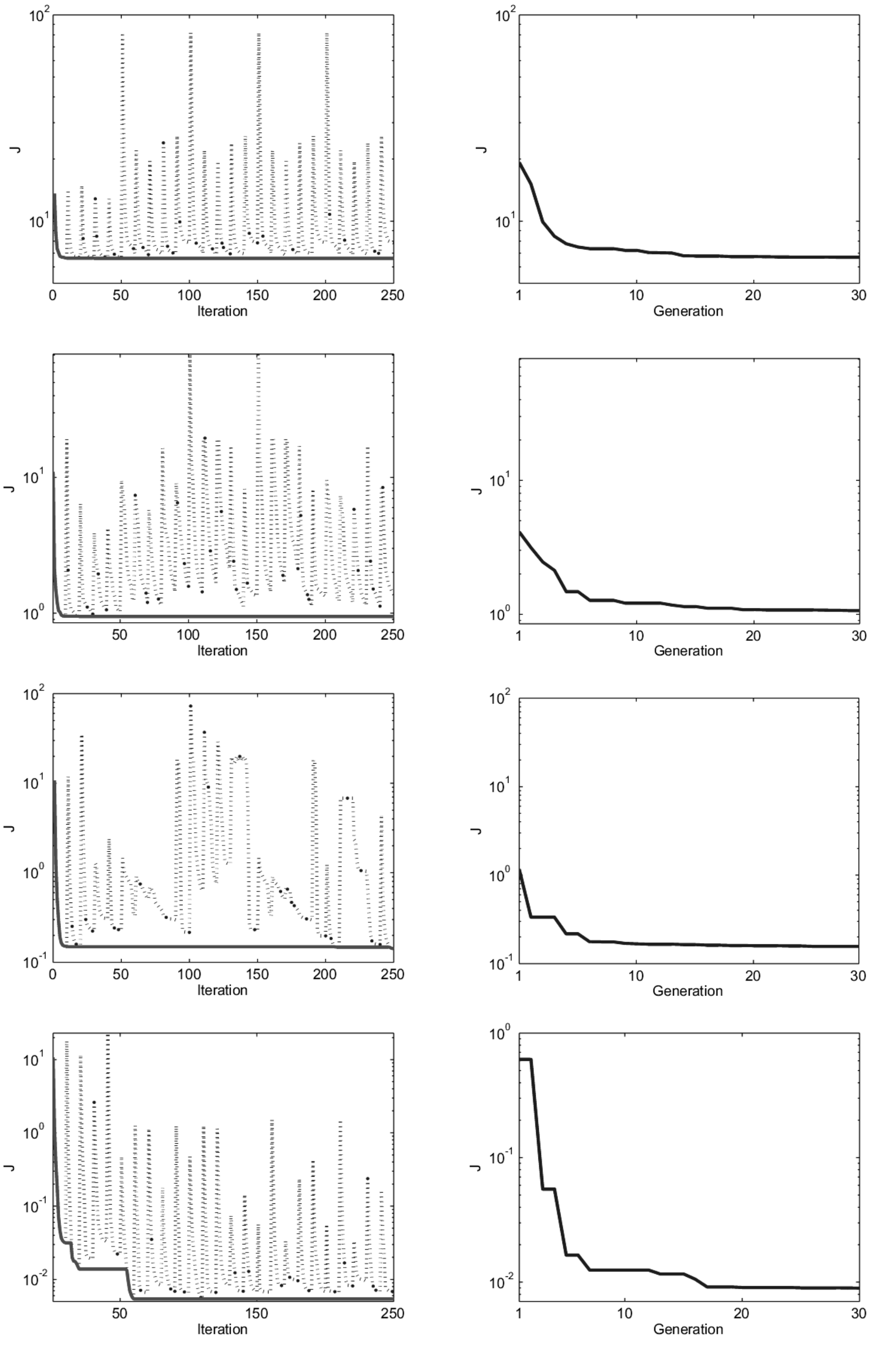

Figure 3: SDA (**left**), HGA (**right**) Convergence histories: Best element (**solid line**) and convergence history (**dashed line**). From **top** to **bottom**: $\alpha = 1, \alpha = 0.1, \alpha = 0.01, \alpha = 0$.