Improving direct search algorithms by multilevel optimization techniques

E. Frandi, A. Papini

Università degli Studi di Firenze
Dipartimento di Ingegneria Industriale
Report No. 9/2013

Improving Direct Search Algorithms by Multilevel Optimization Techniques

Emanuele Frandi[1] and Alessandra Papini[2]

[1]Dipartimento di Scienza e Alta Tecnologia, Università degli Studi dell'Insubria
emanuele.frandi@uninsubria.it
[2]Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze
alessandra.papini@unifi.it

Abstract

Direct Search algorithms are classical derivative-free methods for optimization. Though endowed with solid theoretical properties, they are not well suited for large-scale problems due to slow convergence and scaling issues. In this paper, we discuss how such limitations can be circumvented, on problems for which a hierarchy of objective functions is available, by using multilevel schemes which are able to accelerate the computation of a finest level solution.

Starting from a previously introduced derivative-free multilevel method, based on Coordinate Search optimization with a sampling strategy of Gauss-Seidel type, we consider also the use of sampling strategies of Jacobi type, and present several algorithmic variations which yield more accurate solutions.

Experiments performed on two model problems justify our choices, and show that accurate solutions can be obtained in a reasonable time, even in the case of large-scale instances.

1 Introduction

Direct Search methods are widely used algorithms for derivative free optimization, which enjoy remarkable theoretical properties. Both global and local convergence results can be estabilished [5, 10, 14], and worst-case complexity bounds analogous to those for steepest descent can be proved [16]. However, Direct Search methods suffer in general from several practical limitations:

- convergence tends to be very slow, which in practice forces the user to settle for solutions with only a few significant digits;
- Direct Search methods do not scale well with the problem size. Theoretical bounds for stationarity measures and error norm are dependent on the

number of variables, thus limiting their use to small or moderate-scale problems.

In some cases, restrictions on the problem size can be effectively overcome for problems admitting a hierarchy of related objective functions, such as those arising from discretizing an infinite-dimensional functional on meshes of different size. Indeed, in [6] we introduced a Coordinate Search multilevel scheme based on updates of Gauss-Seidel type, which, by exploiting a sequence of smaller optimization problems to accelerate the computation of a finest-level solution, is capable to obtain approximate solutions to large-scale problems in a reasonable time.

In this paper, which is intended as a follow-up and a complement to our previous work, we address questions related to the accuracy of the computed solution. Namely, we investigate the possibility of devising a derivative-free multilevel scheme which is able to reduce the error to the order of magnitude of the discretization error on any given grid, a property typically observed in multigrid methods for linear systems of equations.

To this aim, we perform an experimental study on the above mentioned multilevel scheme, and suggest some possible algorithmic improvements. Specifically, we consider also Coordinate Search updates of Jacobi type, and show how the accuracy of our multilevel procedure can be improved by using suitable step-size expansion strategies for approximate one-dimensional minimization (see e.g. [9]). Our proposals are discussed and justified by investigating their behavior on a two-dimensional Poisson model problem and on a nonlinear minimum surface problem. We show that in both cases we are able to obtain solutions accurate to the level of the discretization error, a property which is not preserved by the algorithm described in [6].

The paper is organized as follows. Section 2 provides a quick overview on Directional Direct Search methods, summarizing their properties as well as their deficiencies. Section 3 presents our multilevel derivative-free method. Accuracy issues and algorithmic enhancements are discussed in Section 4, and experimental results on the two model problems are reported. Finally, section 5 closes the paper with some concluding remarks.

2 Directional Direct Search methods: properties and limitations

Directional Direct Search algorithms are a class of derivative-free methods for the problem

$$\min_{x \in \mathbb{R}^n} f(x).$$

The basic idea behind this class of methods is to sample the objective function at each iteration k on a finite set of *trial points*, in the attempt to find a new point satisfying a suitable decrease condition [5, 10]. Such sampling is usually divided in two distinct phases:

- **Poll phase:** given a stepsize parameter Δ^k and a *positive spanning set*[1] $\mathcal{G}_k = \{d_1^k, d_2^k, \ldots, d_{p_k}^k\}$, evaluate f on $\mathcal{P}_k = \{x^k + \Delta^k d,\ d \in \mathcal{G}_k\}$.
- **Search phase:** evaluate f on a set of additional trial points $\mathcal{Y}_k = \{y_1^k, \ldots, y_{q_k}^k\}$.

The requirement for $\mathcal{G}_k$ to be a positive spanning set essentially serves to ensure that $\mathcal{G}_k$ contains at least one descent direction for f at x^k, which is a fundamental requirement for proving convergence [10]. The search step, instead, is optional (i.e. $\mathcal{Y}_k$ can be empty) and serves only to accelerate the method by exploring a set of alternative trial points. Note that the distinction between the poll and search phases serves to clarify their role in the algorithm but has no theoretical relevance. In fact, the candidate trial points can be evaluated in any order, as long as $\mathcal{G}_k$ is a positive spanning set and a suitable decrease condition is enforced.

Denoting with $\mathcal{S}$ and $\mathcal{U}$ the sets of successful and unsuccessful iterations respectively, the Direct Search method used throughout this paper can be schematised as in Algorithm 1.

Algorithm 1. A general scheme for Direct Search

Given $x^0 \in \mathbb{R}^n$, $\Delta^0 > 0$, $\tau > 0$, $\theta \in (0,1)$, an integer $k_{\max}$
for $k = 0, 1, 2, \ldots, k_{\max}$

1. *Sampling phase:* Look for $y \in \mathcal{P}_k \cup \mathcal{Y}_k$ satisfying a suitable decrease condition. If it exists, then set $x^{k+1} = y$ and $k \in \mathcal{S}$ (successful step). Otherwise, set $x^{k+1} = x^k$ and $k \in \mathcal{U}$ (unsuccessful step).
2. *Stepsize update:* If $k \in \mathcal{S}$, set $\Delta^{k+1} = \Delta^k$, else if $k \in \mathcal{U}$ set $\Delta^{k+1} = \theta\Delta^k$.
3. *Stopping criterion:* if $\Delta^{k+1} < \tau$, return x^{k+1} and Δ^{k+1}.

Various decrease conditions can be adopted to guarantee the global convergence of the algorithm [5, 10]. We consider the classical sufficient decrease condition

$$\text{find } y \in \mathcal{P}_k \cup \mathcal{Y}_k \text{ s.t. } f(y) < f(x^k) - \rho(\Delta^k).$$

Here, $\rho(t) : \mathbb{R}_+ \to \mathbb{R}_+$ is a continuous, positive *forcing function* such that

$$\lim_{t \to 0^+} \frac{\rho(t)}{t} = 0 \qquad \text{and} \qquad \rho(t_1) < \rho(t_2) \ \text{ if } \ t_1 < t_2.$$

In this paper, we adopt the widely used choice $\rho(t) = \gamma t^2$, $\gamma \in (0,1)$, with $\gamma = 10^{-4}$.

[1] Formally, a set $\mathcal{G} = \{d_1, d_2, \ldots, d_p\} \subset \mathbb{R}^n$ of $p \geq n+1$ vectors positively spans $\mathbb{R}^n$ if for any $v \in \mathbb{R}^n$ there exist $\lambda_1, \ldots, \lambda_p \geq 0$ s.t. $v = \sum_{i=1}^p \lambda_i d_i$.

2.1 Two strategies for Coordinate Search optimization

In the following, we consider Coordinate Search methods, where $\mathcal{G}_k = \{\pm e_i,\ i = 1 \dots n\}$, i.e. the poll points are defined using the set of positive and negative coordinate directions. There are various possible ways to define the sampling phase in a Coordinate Search algorithm. We consider the two schemes outlined below, inspired by the classical Gauss-Seidel and Jacobi updating patterns used in linear equation solvers. As observed in [6], the resulting algorithms can be effectively used to play the same role of iterative smoothers in a traditional multigrid scheme.

A sampling strategy of Gauss-Seidel type (`CS-GS`)

1. Given a current iterate x^k and stepsize Δ^k, initialize $y = x^k$.
2. **for** $i = 1, \dots, n$
   (2.1) find $w = \text{argmin}_{y \pm \Delta^k e_i}\{f(y - \Delta^k e_i), f(y + \Delta^k e_i)\}$.
   (2.2) if $f(w) < f(y) - \rho(\Delta^k)$, set $y_i = w_i$.

In this strategy, a component of the current iterate is updated as soon as a point guaranteeing sufficient decrease is found.

A sampling strategy of Jacobi type (`CS-J`)

1. Given a current iterate x^k and stepsize Δ^k, initialize $y = x^k$.
2. **for** $i = 1, \dots, n$
   (2.1) find $w = \text{argmin}_{x^k \pm \Delta^k e_i}\{f(x^k - \Delta^k e_i), f(x^k + \Delta^k e_i)\}$.
   (2.2) if $f(w) < f(x^k) - \rho(\Delta^k)$, set $y_i = w_i$.
3. if $y \neq x^k$, find $z = x^k + \alpha(y - x^k)$, $\alpha \in (0, 1]$, s.t. $f(z) < f(x^k) - \rho(\Delta^k)$. If it exists, set $y = z$, else set y to the trial point w in step (2) yielding the lowest function value.

In order to find α in step (3) above, a finite backtracking strategy can be used. In the examples of this paper, we start from $\alpha = 1$ and allow for the steplength to be halved at most 8 times.

Independently of which scheme is used, both global and local convergence results can be estabilished for Direct Search, which are stated in the theorems below. For proofs and further discussion, we refer to the comprehensive review in [10].

Theorem 1. *(Global Convergence) Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable, and suppose the set $\mathcal{L}_f^0 := \{x \in \mathbb{R}^n \,|\, f(x) \leq f(x^0)\}$ is compact. Then, Algorithm 1 produces a sequence of iterates $\{x^k\}$ satisfying*

$$\lim_{k \to +\infty} \Delta^k = 0 \text{ and } \lim_{k \to +\infty} \|\nabla f(x^k)\| = 0.$$

Under some additional conditions, it is also possible to show that the subsequence $\{x^k\}_{k\in\mathcal{U}}$ enjoys a r-linear local rate of convergence.

Theorem 2. *(Local Convergence) Suppose $f \in C^2$ and that x^* is a local minimizer s.t. $\nabla^2 f(x^*)$ is positive definite. If x^0 is sufficiently near to x^* and Δ^0 is sufficiently small, the sequence $\{x^k\}$ produced by Algorithm 1 is such that*

$$\begin{aligned} &\lim_{k\to+\infty} x^k = x^*, \\ &\|x^k - x^*\| \leq \zeta\Delta^k, \quad \textit{for } k \in \mathcal{U}, \end{aligned} \tag{1}$$

where ζ is a constant independent from k.

Remark 1. Note that, according to (1), the steplength tolerance τ can be reasonably interpreted as a rough measure of the error we expect to obtain when Algorithm 1 stops. We cannot however estimate a-priori how small should Δ^0 be in order to observe r-linear convergence. As for the constant ζ, in the case of Coordinate Search it can be shown that $\zeta \propto \frac{\sqrt{n}}{\phi_{\min}}$, where $\phi_{\min}$ is the minimum eigenvalue of $\nabla^2 f(x^*)$ (see [10], proof of Theorem 3.15), and $\sqrt{n}$ is the *cosine measure* of the set of the coordinate directions [10]. This intuitively means that, as a problem grows in size, increasingly small stepsizes have to be taken to reach a given level of accuracy.

2.2 A quadratic model problem

To illustrate the points discussed above, consider the following minimization problem, arising from the discretization of a two-dimensional Poisson equation $-\Delta x(t,s) = b(t,s)$ with Dirichlet boundary conditions on the unit square (hereafter P2D for the sake of brevity):

$$\min_{x\in\mathbb{R}^{(n-1)^2}} \tfrac{1}{2}x^T Ax - b^T x, \tag{2}$$

where $A \in \mathbb{R}^{(n-1)^2\times(n-1)^2}$ is the five-point discrete Laplacian for a uniform $(n+1)^2$-point grid, and $b \in \mathbb{R}^{(n-1)^2}$. The vector b is obtained by sampling $b(t,s) = 2s^2(1-6t^2)(1-s^2) + 2t^2(1-6s^2)(1-t^2)$ on the grid, a choice corresponding to the analytical solution

$$x_a(t,s) = t^2(1-t^2)s^2(s^2-1).$$

We solve the problem in sizes $n = 2^i$, $i = 4, 5, \ldots$ using Algorithm 1 with the `CS-GS` and `CS-J` strategies. We choose $x^0 = (1,\ldots,1)^T$, $\Delta^0 = \|x^0\|_\infty = 1$, $\theta = 0.25$ and $\tau \in \{10^{-4}, 10^{-5}\}$. We set the maximum number of allowed iterations to $k_{\max} = 20000$, and solve the problem up to the largest possible scale within $k_{\max}$ iterations and a CPU time of 1 hour. Table 1 reports, for each of the considered problems, the number of iterations required to satisfy the convergence criterion, the error in infinity-norm with respect to the exact minimizer (denoted as x^* and computed by solving the equivalent linear system

$Ax = b$, while the final approximation is denoted simply as x) and the elapsed CPU time (in seconds).

All the algorithms described in this paper have been coded in MATLAB$^{\text{TM}}$. The experiments were executed on a 3.40GHz Intel® 4-core personal computer with 16GB of main memory, running a Linux OS.

`CS-GS`		$\tau = 10^{-4}$			$\tau = 10^{-5}$	
Size	Time	Iter	$\|x - x^*\|_\infty$	Time	Iter	$\|x - x^*\|_\infty$
225	0.47	220	3.03e − 03	0.69	337	2.07e − 04
961	5.80	704	1.19e − 02	8.96	1115	8.07e − 04
3969	51.7	1550	6.31e − 02	116	3534	2.98e − 03
16129	451	3369	2.14e − 01	1330	9951	1.38e − 02
`CS-J`		$\tau = 10^{-4}$			$\tau = 10^{-5}$	
Size	Time	Iter	$\|x - x^*\|_\infty$	Time	Iter	$\|x - x^*\|_\infty$
225	0.82	364	3.03e − 03	1.21	556	2.07e − 04
961	9.61	1133	1.19e − 02	16.0	1912	8.07e − 04
3969	83.2	2522	6.31e − 02	207	6161	2.98e − 03
16129	826	6195	2.14e − 01	2260	16696	1.38e − 02

Table 1: Performance of Algorithm 1 with `CS-GS` and `CS-J` sampling on increasing size instances of the P2D problem.

Results in Table 1 are consistent with the well known practical drawbacks of Coordinate Search and Direct Search in general:

- even on relatively small problems, the method is slow. Note that by exploiting the structure of the problem (in particular, the partial separability of the objective function) it is possible to obtain fast function value updates along the coordinate directions [6, 13], whose cost is constant and independent of n. Still, large-scale problems require a prohibitive amount of time;
- the method does not scale well with the size of the problem. The number of variables increases by a factor of 4 in each instance, yet the CPU time required to satisfy the stopping criterion increases by about 10 times each time the mesh is refined;
- the obtained accuracies are generally modest and actually degrade as the size of the problem increases, a behaviour consistent with the increase of the upper bound in (1).

3 Derivative-free multilevel optimization

Suppose to have a hierarchy of problems

$$\min_{x_h \in \mathcal{X}_h} f_h(x_h), \tag{3}$$

where f_h is possibly obtained by discretizing a continuous functional on a given grid with spacing parameter h. Any discretization hierarchy is allowed in principle, though in practice we mostly consider l-level discretizations on regular grids with spacings $h, 2h, \ldots, 2^{l-1}h$.

Given a pair of successive discretizations, i.e. a finer level and a coarser level, denoted by the respective grid spacing parameters h and H, let $\mathcal{X}_h$ and $\mathcal{X}_H$ be the corresponding variable spaces, of dimension n_h and n_H respectively. Suppose that a pair of linear operators to transfer variables between grids, a *restriction* $R : \mathcal{X}_h \to \mathcal{X}_H$ and a *prolongation* $P : \mathcal{X}_H \to \mathcal{X}_h$, is defined, satisfying the *variational property*

$$P = cR^T, \qquad c > 0 \text{ constant.} \tag{4}$$

In this paper, we adopt as a default choice the classical linear interpolation and full-weighting operators, which are most often used in multigrid methods [2, 15].

Several approaches to solving problems of the form (3) have been examined in the literature [7, 8, 12]. Here, we consider the multilevel v-cycle correction scheme defined in [6], which builds on the framework introduced in [12] and is schematized in Algorithm 2.

Algorithm 2. `ML/CS`**: a multilevel Direct Search v-cycle**

Given $x_h^0 \in \mathcal{X}_h$, Δ_h^0, and integers $\nu_1, \nu_2 > 0$

1. *Bottom level solver:* If h is the coarsest grid, starting from x_h^0 and Δ_h^0, apply Algorithm 1 to f_h until convergence to obtain x_h^+, and return.
2. *Pre-smoothing:* starting from x_h^0 and Δ_h^0, apply ν_1 iterations of Algorithm 1 on f_h, to obtain $\tilde{x}_h$ and $\tilde{\Delta}_h$.
3. *Restriction:* define a coarse-level initial guess $\tilde{x}_H = R\tilde{x}_h \in \mathcal{X}_H$.
4. *Lower level optimization:* starting from $\tilde{x}_H$ and Δ_h^0, obtain x_H^+ by recursively calling `ML/CS` to minimize an appropriate *surrogate model*.
5. *Prolongation:* define a fine-level search direction $e_h = P(x_H^+ - \tilde{x}_H)$.
6. *Correction:* find $x_h^+ = \tilde{x}_h + \alpha e_h$, $\alpha > 0$, s.t. $f_h(x_h^+) \leq f_h(\tilde{x}_h)$.
7. *Post-smoothing:* starting from $\tilde{\Delta}_h$, refine x_h^+ by applying ν_2 iterations of Algorithm 1.

Specifically, the procedure above describes a (ν_1, ν_2)-cycle, where ν_1 and ν_2 are small positive integers defining the number of smoothing iterations. Usually, a few iterations per cycle suffice to attain proper multigrid efficiency [2, 6, 12, 15]. Moreover, if the above procedure is iterated on the finest-level, it can be proved that the resulting algorithm converges globally to a stationary point [6, 12].

The correction step $x_h^+ = \tilde{x}_h + \alpha e_h$ in Algorithm 2, that is, the analogue of the classical multigrid correction step, can be interpreted as an additional search step to accelerate the underlying algorithm, introduced between the ν_1 iterations of the pre-smoothing and the ν_2 iterations of the post-smoothing. Hence, a key issue for the success of this procedure is the computation of a good search direction e_h in the correction step.

In this regard, although the choice of the surrogate model in step (4) does not affect the theoretical properties of Algorithm 2, selecting an appropriate coarse-level objective function is critical in practice. Several well-known multilevel models, such as those in [12] and [7, 8], use as a surrogate model the function

$$f_H^s(z) = f_H(z) - v_H^T z, \tag{5}$$

where

$$v_H = \nabla f_H(\tilde{x}_H) - R\nabla f_h(\tilde{x}_h), \tag{6}$$

a definition which requires the availability of gradients at any level and any point.

Remark 2. It is not hard to see that, for any quadratic problem $f_h(x_h) = \frac{1}{2}x_h^T A_h x_h - b_h^T x_h$, with A_h positive definite (such as the P2D problem defined previously), the correction defined by the model (5) is in fact equivalent to the classical multigrid correction based on solving a residual equation. To see this, we first observe that, by optimizing the surrogate model $f_H^s(x_H)$, we are seeking for the unique point x_H^+ s.t. $\nabla f_H^s(x_H^+) = 0$. Now,

$$\nabla f_H^s(x_H) = \nabla f_H(x_H) - v_H = A_H x_H - b_H - (\nabla f_H(\tilde{x}_H) - R\nabla f_h(\tilde{x}_h)).$$

The solution is then given by

$$\begin{aligned} x_H^+ &= A_H^{-1}(\nabla f_H(\tilde{x}_H) + b_H - R\nabla f_h(\tilde{x}_h)) = A_H^{-1}(A_H\tilde{x}_H + R(b_h - A_h\tilde{x}_h)) \\ &= \tilde{x}_H + A_H^{-1}R(b_h - A_h\tilde{x}_h), \end{aligned}$$

therefore $x_h^+ = \tilde{x}_h + P(x_H^+ - \tilde{x}_H) = \tilde{x}_h + PA_H^{-1}R(b_h - A_h\tilde{x}_h)$ in the correction step[2]. Consider now the equivalent linear system $A_h x_h = b_h$. In linear multigrid, the correction is performed by solving for e_H the residual equation $A_H e_H = \tilde{b}_H$, where $\tilde{b}_H = R(b_h - A_h\tilde{x}_h)$ is the restriction of the residual at $\tilde{x}_h$, and defining a new point $x_h^\star = \tilde{x}_h + Pe_H$ [2, 15]. Analytically,

$$x_h^\star = \tilde{x}_h + PA_H^{-1}R(b_h - A_h\tilde{x}_h).$$

We then have $x_h^+ = x_h^\star$, and therefore the two correction schemes are equivalent.

In a derivative-free context, a surrogate model alternative to $f_H^s(z)$ in (5), which does not rely on the explicit computation of $\nabla f_h(\tilde{x}_h)$ and $\nabla f_H(\tilde{x}_H)$ but

[2]We are not taking into consideration the line-search in step (6), as it is not essential for convergence in this case [12].

still retains the key features of the original model, is needed. A suitable surrogate function for this purpose was introduced in [6], which is defined as follows:

$$q_H^s(z) := \frac{f_H(z) + f_H(2\tilde{x}_H - z)}{2} + (Rg_h)^T z, \tag{7}$$

where g_h is some approximation of the fine-level gradient $\nabla f_h(\tilde{x}_h)$.

By exploiting the structure of the poll phase in either the `CS-GS` or `CS-J` strategy, a convenient gradient approximation g_h can be readily obtained without the need of computing any additional function evaluation. Specifically, initialize $g_h \leftarrow (0, \ldots, 0)^T$ and suppose that, at iteration k, Algorithm 1 is sampling f_h around y along the i-th axis, with stepsize Δ_h^k. Then, the i-th coordinate of g_h is updated as follows:

$$g_{h,i} \leftarrow \frac{f_h(y + \Delta_h^k e_i) - f_h(y - \Delta_h^k e_i)}{2\Delta_h^k}. \tag{8}$$

Note that in the case of a sampling of Jacobi type, where $y = x_h^k$ for each coordinate, (8) is in fact a centered difference approximation of $\nabla f_h(x_h^k)$. This is not the case for a Gauss-Seidel sampling, as y is not constant during the coordinate cycle.

The impact ot inexact gradients in a multilevel optimization context has been considered in [11], where it is observed that the correction scheme tends to smooth the errors on the gradient, which therefore have a limited effect on convergence.

We refer to [6] for a complete explanation of the motivations behind the introduction of the model (7). For the purposes of this paper, it suffices to recall its main properties, which we state in the following Proposition.

Proposition 1. *The surrogate (7) satisfies*

$$\nabla q_H^s(\tilde{x}_H) = Rg_h \approx R\nabla f_h(\tilde{x}_h).$$

Furthermore, it can be shown that

$$q_H^s(z) = f_H^s(z) + \nabla f_H(\tilde{x}_H)^T \tilde{x}_H + (R(g_h - \nabla f_h(\tilde{x}_h)))^T z + \mathcal{O}(\|z - \tilde{x}_H\|^3).$$

Note in particular that, up to a constant term and a third-order remainder which is hopefully very small, the difference between the surrogates f_H^s and q_H^s depends only on the error on the fine-level gradient.

3.1 A full-multilevel scheme: the FML/CS algorithm

In practice, v-cycle correction schemes are usually combined with a warm-start initialization strategy based on a mesh refinement, known as *full multigrid* or FMG in the context of linear systems [2, 15]. Such strategy can be readily extended to the optimization case, in which case we speak of a *full multilevel* scheme [7, 12]. Algorithm 3 describes our specialization of full multilevel to the case of a Coordinate Search based algorithm.

Algorithm 3. `FML/CS`**: a Direct Search full-multilevel method**

Given $x_h^0 \in \mathcal{X}_h$, Δ_h^0, and an integer ν_0

1. If h is the bottom grid, apply Algorithm 1 until convergence and return.
2. Downdate the initial guess, $x_H^0 = Rx_h^0$.
3. Recursively call `FML/CS` to minimize f_H from x_H^0, Δ_h^0, to obtain x_H and Δ_H.
4. Define a new initial guess, $x_h = Px_H$.
5. Starting from x_h and a suitable stepsize Δ_h, minimize f_h by calling `ML/CS` ν_0 times.

An appropriate choice of Δ_h in step (5) constitutes a crucial point to control the practical behaviour of the algorithm. A possible strategy, outlined in [6], is given by

$$\bar{\Delta} = \min\left\{\Delta_h^0, C\Delta_H\right\}, \qquad \Delta_h = \max\left\{\bar{\Delta}, \tfrac{\Delta_H}{\theta}\right\}, \tag{9}$$

where $C > 1$ is a positive constant compensating for the amount of error possibly reintroduced by the prolongation. The latter safeguard ensures that $\Delta_h > \tau_{\min}$, where $\tau_{\min}$ refers to the tolerance chosen for the solver at the bottom level.

Taking into account the observations above, we can think of the framework presented in this Section as an approach to the issue of derivative-free multilevel optimization from a double point of view. On one hand, the correction scheme `MG/CS` is interpreted as an accelerating strategy for the Direct Search method in Algorithm 1. On the other, `FML/CS` in Algorithm 3 can be seen as a derivative-free instance of a full multilevel method for optimization problems.

Remark 3. An important observation is in order, which lies at the basis of the strategies devised in Section 4. Though the v-cycle scheme can in principle be iterated, full multigrid techniques are usually *not* intended as iterative methods, and the parameter ν_0 is usually set to 1 [2]. Rather, full multigrid is classically interpreted as a direct method for linear systems, where the interplay between different grids is structured in such a way that the loss in accuracy incurred when interpolating on a finer level is compensated by the correction provided by the v-cycle on that level.

Generally, the error on a FMG solution at the finest level has the same order of magnitude of the discretization error, and as such the solution does not need further refinement, meaning that the v-cycle can be employed as a correction strategy rather than as a part of an iterative method [2, 15]. This gives rise to the main question addressed in this paper, that is, whether it is possible to construct a derivative-free approximate scheme which retains this behaviour in practice.

Considering again the P2D example of Section 2.2, we repeat a similar set of experiments using the `FML/CS` algorithm with the surrogate model (7), this

CS-GS		$\tau_{\min} = 10^{-4}$			$\tau_{\min} = 10^{-5}$	
Size	Time	Speedup	$\|x - x^*\|_\infty$	Time	Speedup	$\|x - x^*\|_\infty$
225	0.14	x3.42	4.76e − 03	0.16	x4.31	4.36e − 03
961	0.18	x32.2	4.05e − 03	0.19	x47.2	3.41e − 03
3969	0.55	x94.0	3.79e − 03	0.57	x204	2.98e − 03
16129	0.96	x303	3.80e − 03	1.94	x686	2.71e − 03
65025	1.49	–	3.80e − 03	7.35	–	2.71e − 03
261121	3.45	–	3.80e − 03	13.4	–	2.71e − 03
1046529	11.4	–	3.80e − 03	21.6	–	2.71e − 03
CS-J		$\tau_{\min} = 10^{-4}$			$\tau_{\min} = 10^{-5}$	
Size	Time	Speedup	$\|x - x^*\|_\infty$	Time	Speedup	$\|x - x^*\|_\infty$
225	0.19	x4.32	4.76e − 03	0.21	x5.76	4.36e − 03
961	0.23	x41.8	3.84e − 03	0.24	x66.7	3.35e − 03
3969	0.60	x139	3.21e − 03	0.63	x329	2.87e − 03
16129	1.81	x456	3.22e − 03	2.15	x1051	2.24e − 03
65025	2.32	–	3.22e − 03	7.25	–	6.01e − 04
261121	4.34	–	3.22e − 03	31.7	–	5.13e − 04
1046529	12.5	–	3.22e − 03	56.6	–	5.14e − 04

Table 2: Performance of Algorithm 3, with `CS-GS` and `CS-J` sampling and stepsize strategy (9), on increasing size instances of the P2D problem.

time solving the problem up to a size of about one million variables. On the coarsest grid, the problem has size $n_h = 49$. We use Algorithm 1 with both `CS-GS` and `CS-J` sampling, employing the same parameters as in Section 2.2. The stepsize strategy used to choose Δ_h in step (5) of `FML-CS` is the one in (9), with $C = 4$. Results in Table 2 show the total CPU time (in seconds), the speedup with respect to the plain Direct Search solution in Table 1, and the error in infinity-norm. It can be seen how the full-multilevel strategy provides enormous speedups with respect to a plain Direct Search solution, with comparable or better accuracy. However, it is apparent that the accuracy does not scale well with the grid spacing. In particular, above a certain problem size the magnitude of the error tends to remain constant, in contrast with the ideal behaviour observed in classical multigrid, where the error has magnitude $\mathcal{O}(h^2)$.

The key point here is that in multilevel methods each level can smooth out only certain frequencies of the error (see [2] and the discussion in [6], Section 4.2). In principle, full multilevel initialization should solve the problem on a given grid as accurately as possible before interpolating, since low-frequency inaccuracies propagating from lower levels are very difficult to eliminate on the finer grids. Results in Table 2 also show that this issue cannot be fixed by using a stricter tolerance alone, as in general the additional running time is not compensated by a proportionally better accuracy.

The source of such inaccuracies lies in Algorithm 1 being unable to smooth out effectively certain components of the error, a behaviour which can be ascribed to inexact minimization. The errors introduced by employing an approximate surrogate can also play a role. Section 4 discusses these issues in

detail and outlines some possible solutions.

4 Accuracy issues and expansion steps

Let us start by considering again the quadratic model problem. Given the equivalence between the multilevel correction and the residual equation in the linear case, as shown in Remark 2, it is clear that, if a nonlinear Jacobi or Gauss-Seidel optimization method is employed as a smoother, then the resulting scheme becomes equivalent to a classic linear multigrid method, thus retaining all the related properties in terms of the accuracy of the obtained solution. Namely, as pointed out in Remark 3, the capability to output, for any given grid h, a solution x_h such that $\|x_h - x_h^*\| = \mathcal{O}(h^2)$.

Here, we address the question of whether it is possible to replicate the same behaviour with an inexact minimization scheme and employing the approximate surrogate model (7). We show that this is indeed possible by combining several elements:

- more accurate coordinate-wise minimizations during the search phase of the Coordinate Search algorithm, to mimic the behavior of a nonlinear Jacobi or Gauss-Seidel method;
- improving the full multilevel initialization in Algorithm 3, with a different and more demanding strategy for choosing Δ_h;
- possibly, tuning cycle parameters such as the numbers ν_1 and ν_2 of smoothing iterations.

On the coarsest level, where the problem is very small, it is easy, by choosing an appropriately small tolerance $\tau_{\min}$, to obtain a solution accurate to the level of the discretization error using Algorithm 1. The main difficulty, which we try to address by the above expedients, lies in devising a scheme capable to eliminate all the components of the error corresponding to the high frequencies on the current grid before interpolating. If the correction scheme is not effective, such components will be very difficult to eliminate on the finer levels, causing accuracy bottlenecks such as those seen in Table 2.

4.1 Smoothing behavior of ML/CS

The key to make multigrid work lies in the interplay between the smoothing on different grids [2, 15]. As argued in [6], the same is true in the optimization case: the smoother on the finer level quickly damps high-frequency modes of the error but is unable to reduce low-frequency ones. The correction step then recursively eliminates the remaining components, by providing a vector which approximates the opposite of the error vector, while the post-smoothing phase takes then care of the remaining oscillatory components.

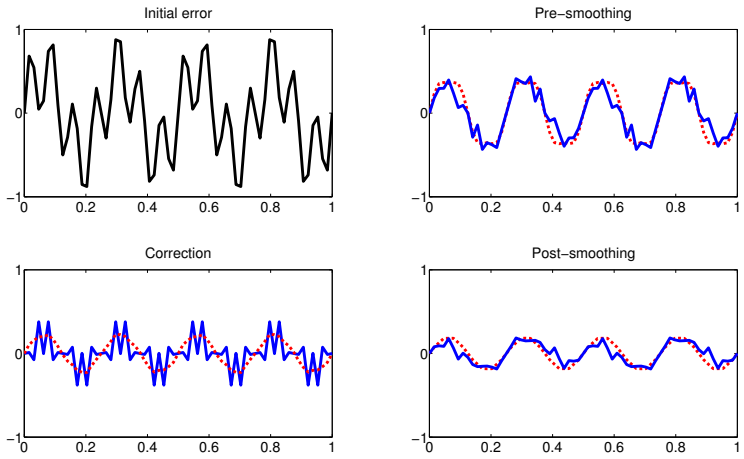


Figure 1: V-cycle - linear Jacobi (dashed line) and `CS-J` (continuous line), $\nu_1 = \nu_2 = 2,\ n_h = 63$.

Example 1. Figure 1 shows this mechanism at work on a one-dimensional Poisson problem with finest grid size $n = 63$, zero right-hand side and initial guess

$$x_i^0 = \frac{1}{2}\left(\sin\frac{8\pi i}{n} + \sin\frac{28\pi i}{n}\right), \quad i = 1, \ldots, 63.$$

The red dashed lines correspond to the error obtained by a linear multigrid Gauss-Seidel v-cycle, as described in [2], and the continuous blue lines to the error obtained by the `ML/CS` algorithm with a `CS-J` sampling and initial stepsize $\Delta^0 = 1$. The cycle parameters for both algorithms are $\nu_1 = \nu_2 = 2$.

However, if the experiment above is repeated with the initial guess

$$x_i^0 = \frac{1}{2}\left(\sin\frac{8\pi i}{n} + \sin\frac{24\pi i}{n}\right), \quad i = 1, \ldots, 63,$$

it is apparent, as shown in Figure 2, that the multilevel smoothing mechanism is not working properly. In particular, the stepsize is too large to update all components effectively, causing large high-frequency peaks to persist after the pre-smoothing phase.

4.2 Expanding stepsizes in Coordinate Search

A classical way to enhance the performance of Direct Search methods is to perform expansion steps along a promising direction [9]. Roughly speaking, given an expansion parameter $\eta > 1$, the idea is to exploit a good direction $d \in \mathcal{G}_k$ by attempting longer trial steps: $x^k + \eta\Delta^k d$, $x^k + \eta^2\Delta^k d, \ldots$ and so on.

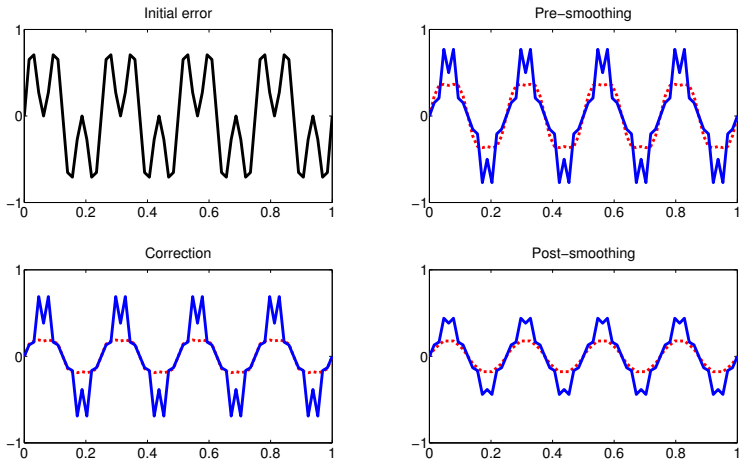


Figure 2: V-cycle - linear Jacobi (dashed line) and `CS-J` (continuous line), $\nu_1 = \nu_2 = 2,\ n_h = 63$.

A possible procedure is summarized below for the specific case of Coordinate Search.

Stepsize expansion for Coordinate Search

Suppose that, during step (1) of either `CS-GS` or `CS-J`, the direction e_i is being explored (the case $-e_i$ is analogous).
if $f(x^k + \Delta^k e_i) < f(x^k) - \rho(\Delta^k)$ **then**
  $\Delta_{\text{loc}} = \Delta^k$.
  while $f(x^k + \eta\Delta_{\text{loc}} e_i) < f(x^k) - \rho(\eta\Delta_{\text{loc}})$ **do**
    $\Delta_{\text{loc}} = \eta\Delta_{\text{loc}}$.
  end while
end if
Set $y_i = x_i + \Delta_{\text{loc}}$.

From a theoretical perspective, the additional function evaluations can be regarded as part of the search phase. In the case of a Coordinate Search sampling, they correspond to additional trial points along the coordinate directions. This design potentially provides several advantages:

- additional flexibility in choosing smaller initial stepsizes, as a mechanism automatically increasing the step length when necessary is introduced;
- one-dimensional searches are more accurate, yielding steps closer to those one would obtain by using exact coordinate-wise minimization;

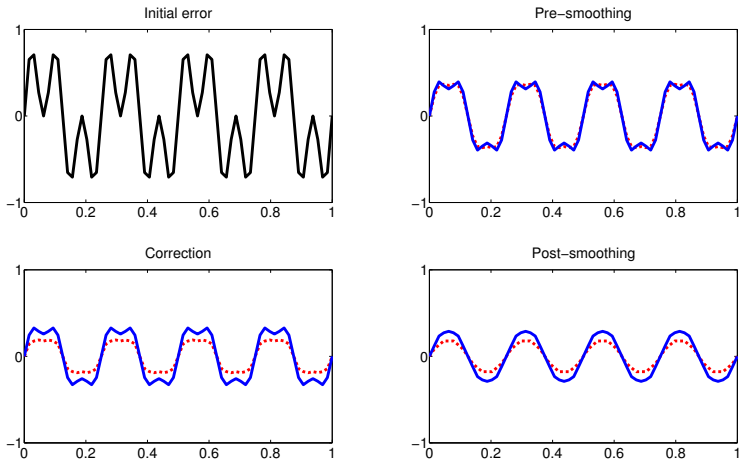


Figure 3: Linear Jacobi (dashed line) and `CS-J` + expansion (continuous line), $\nu_1 = \nu_2 = 2,\ n_h = 63$.

- situations as the one in Figure 2, where the use of a fixed stepsize performs poorly because different coordinates require different stepsizes, can be prevented.

To illustrate the above points, we consider again the second experiment of Example 1. Figure 3 shows the improvements obtained over the case of Figure 2 when the strategy above is implemented in Algorithm 1, choosing $\eta = 2$ and $\Delta^0 = 1/64$. As expected, expanding the stepsize allows to smooth out the high-frequency components of the error more effectively. Note that the different choice of Δ^0 is crucial here. Our experiments showed that, taking an initial stepsize $\Delta^0 = 1$ as before, the expansion strategy does not yield any benefit, while, if we set $\Delta^0 = 1/64$ without the expansion strategy, Coordinate Search becomes very slow and is unable to obtain any smoothing effect in just ν_1 iterations. In principle, the idea is to design a smoother where we start from very small stepsizes and let the search along each coordinate direction define corrections of different magnitude as appropriate. Naturally, more accurate updates require more function evaluations to update all the n coordinates, a tradeoff which could however be favourable if the improvements in the correction scheme obtained with the expansion technique allowed us to reach more accurate solutions. The experiments presented later in this Section show that this is indeed the case.

4.3 Managing the stepsize in the full multilevel scheme

As already recalled, controlling the stepsize in the `FML/CS` algorithm plays a fundamental role in obtaining accurate solutions. The stepsize expansion introduced in the previous paragraph can be successfully exploited in this sense, in that it permits to take smaller steps than normally allowed in Direct Search methods. Essentially, the idea is to try keeping the starting stepsize in step (5) of `FML/CS` proportional to the discretization error on level h. To this aim, we propose the following simple strategy, which we found to be very effective in practice:

$$\Delta_h = c^{(l_h - l_{\min})} \tau_{\min}, \tag{10}$$

where $c < 1$ is a constant, $\tau_{\min}$ is the tolerance on the bottom grid, and l_h and $l_{\min}$ denote the indexes of the current and coarsest level respectively.

The resulting scheme is depicted in Figure 4. Note that this strategy is more demanding than the one in (9), where the stepsize on the finer grid actually starts from a larger value than on the coarser grid. Here, on the opposite, thanks to the expansion strategy, we do not have to start from larger stepsizes to handle solutions far from the optimum.

As for the value of the constant c, two factors should be taken into consideration. First, supposing that, on a given problem, we know that the discretization error reduces by a certain factor ξ at each grid refinement (e.g. $\xi = 4$ in the case of the P2D problem), we can use this factor to reduce the initial stepsize as we proceed to finer grids. Then, we can take into account Theorem 2, where the constant ζ, which in turn depends on $\sqrt{n_h}$, gives a rough indication of how the problem size can worsen the upper bound on the error in proximity of a solution (for discretized problems on $2D$ regular grids, where $n_h \approx 4n_H$, the upper bound doubles when moving to the next finer grid), potentially meaning that the stepsize could need a further reduction for the error to reach a certain magnitude.

However, since there is no guarantee on the tightness of the bound provided by Theorem 2, the choice of c is largely problem-dependent, and heuristic to a certain degree. For example, as seen in the experiments below, in the case of the P2D problem, choosing $c = 1/4$ yielded very good results in practice, while on the minimum surface model problem of Section 4.5 the scheme seemed to benefit from accounting for an additional multiplicative factor $\sqrt{n_h/n_H} \approx 2$.

Note that (10) forces the finer grids to use very small stepsizes. Therefore, for any given level h, we need to adjust the tolerance τ_h for Algorithm 1 in the corrisponding v-cycle. Preliminary experiments revealed that, when using expansion steps, coupled with a sufficiently small initial Δ_h, asking for many stepsize reductions is in most cases not necessary to produce good correction steps. Allowing for just one reduction proved enough in most cases, hence we opt for choosing $\tau_h = \theta \Delta_h$.

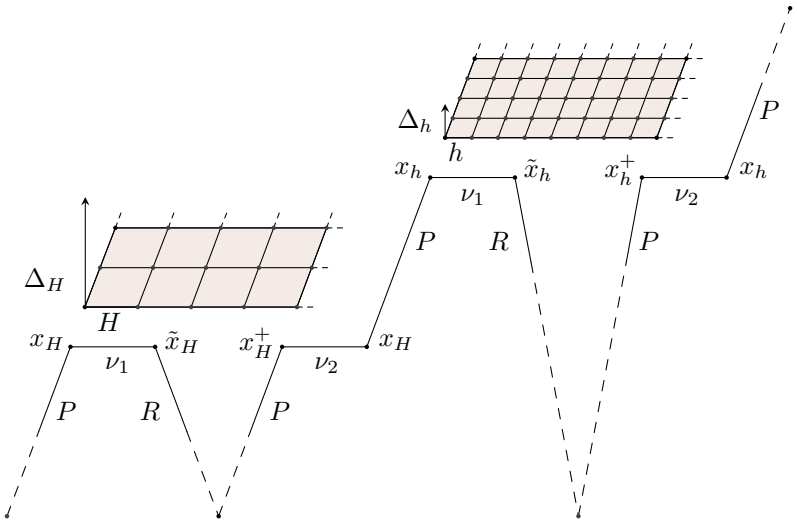


Figure 4: Sketch of the full-multilevel scheme with stepsize strategy (10). Roughly speaking, the starting stepsize in the Coordinate Search smoother is scaled by a factor c as the grid spacing is halved.

4.4 Results on the quadratic model problem

We first evaluate the effectiveness of the proposed strategy by repeating the experiment of Section 3.1 on the P2D model problem. The tolerance on the coarsest grid is set to $\tau_{\min} = 4 \cdot 10^{-5}$, a value which suffices to attain the discretization error at the bottom level with the Coordinate Search solver.

In Table 3, we report the performance of Algorithm 3 with regard to both computational cost (CPU time and total number of function evaluations on all levels) and accuracy of the solution. As customarily done in multigrid literature [2], errors are shown in the discrete L_2-norm, which is defined, on a regular discrete grid of spacing h, as the L_2-norm scaled by a factor accounting for the geometry of the problem:

$$\|x_h\|_{L_2} = \left(h^d \sum_i x_{h,i}^2 \right)^{\frac{1}{2}},$$

where d is the dimensionality of the discretized domain [2]. DE denotes the discretization error $x^* - x_a$ (here, x_a is the restriction of the analytical solution to the discrete mesh). We also report the infinity-norm of the error for comparison

with the results in Tables 1 and 2. Finally, in order to estimate the impact of using the derivative-free model (7), in the last column we report the error for the solution x_c found by using the classical surrogate function (5).

`CS-GS`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x-x^*\|_{L_2}$	$\|x-x^*\|_\infty$	$\|x_c-x^*\|_{L_2}$
3969	$3.55e+05$	1.87	$6.44e-06$	$1.97e-04$	$6.77e-04$	$6.82e-06$
16129	$1.24e+06$	6.05	$1.61e-06$	$3.14e-05$	$5.53e-05$	$1.92e-06$
65025	$4.59e+06$	22.5	$4.03e-07$	$5.59e-05$	$1.31e-04$	$4.84e-07$
261121	$1.69e+07$	82.0	$1.01e-07$	$3.59e-06$	$1.00e-05$	$1.23e-07$
1046529	$6.66e+07$	316	$2.52e-08$	$1.68e-05$	$3.84e-05$	$2.92e-08$
`CS-J`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x-x^*\|_{L_2}$	$\|x-x^*\|_\infty$	$\|x_c-x^*\|_{L_2}$
3969	$3.28e+05$	1.88	$6.44e-06$	$5.61e-06$	$3.20e-05$	$1.01e-05$
16129	$1.11e+06$	5.20	$1.61e-06$	$1.55e-06$	$1.01e-05$	$3.02e-06$
65025	$4.12e+06$	20.1	$4.03e-07$	$3.97e-07$	$2.73e-06$	$6.97e-07$
261121	$1.59e+07$	69.3	$1.01e-07$	$1.06e-07$	$7.60e-07$	$1.91e-07$
1046529	$6.24e+07$	277	$2.52e-08$	$2.73e-08$	$2.00e-07$	$4.75e-08$

Table 3: Performance of Algorithm 3 with strategy (10) and stepsize expansion on the P2D problem.

Results in Table 3 show that the new stepsize strategy, particularly in conjunction with the `CS-J` smoothing scheme, displays exactly the same behaviour as a linear multigrid method. The error norm scales very well with the grid size, showing that an accuracy at the level of the discretization error is attained at each level. Of course, such a result comes at the expense of an additional computational cost.

Also, note that the `CS-GS` variant of the algorithm, while offering the same level of performance in terms of CPU time, performs worse with respect to the accuracy of the obtained solution. The reason for this arguably lies in the different ways in which the Jacobi and Gauss-Seidel schemes compute the inexact gradient g_h. Focusing on a given grid, consider the `CS-J` sampling scheme. Suppose Algorithm 1 in the pre-smoothing phase stops at $x_h^{\nu_1} = \tilde{x}_h$, and suppose the last Jacobi search step was successful (which most often happens in practice if the stepsize is managed properly). Since the approximation g_h defined by (8) is equivalent to a centered finite difference approximation of $\nabla f_h(x_h^{\nu_1-1})$, which is exact in the case of a quadratic model problem, we have

$$\|\nabla f_h(\tilde{x}_h) - g_h\| = \|\nabla f_h(x_h^{\nu_1}) - \nabla f_h(x_h^{\nu_1-1})\| \leq L\|x_h^{\nu_1} - x_h^{\nu_1-1}\|$$

for some constant $L > 0$. This is to mean that the error on the gradient can be bounded as a function of the distance between the last two Coordinate Search iterates, which we can expect to be small when Δ^k is small. The `CS-GS` strategy, on the other hand, does not allow for an analogous interpretation, as all the coordinates can potentially change during one iteration.

Note that these results are not in contrast with the discussion in [11], as what we consider here is the impact of the approximation on the accuracy of the solution obtained after running `FML/CS`, and we do not perform additional v-cycles until convergence. In this case, as seen from the results above, the effect of the error on the gradient may possibly be more critical.

4.5 A nonlinear test problem

We now show how the previously observed behaviour extends also to the case of more general nonlinear optimization problems. We consider as a model the classical minimal surface problem on a square $S = [t_l, t_u] \times [s_l, s_u]$, consisting in finding the surface of smallest area between those with a given profile on ∂S [1]:

$$\min_{x \in \mathcal{V}} \int_S \sqrt{1 + \|\nabla x(t,s)\|_2^2}\, dt\, ds, \tag{11}$$

where $\mathcal{V} = \{x \in \mathcal{H}^1(S) \,|\, x(t,s) = x_0(t,s) \text{ on } \partial S\}$. The functional (11) is discretized on a uniform $(n+1)^2$-point triangulation of S (see [1, 6] for details), to obtain the following objective function:

$$f_h(x_h) := \frac{h^2}{2} \sum_{i,j=0}^{n-1} \sqrt{1 + a^2 + b^2} + \sqrt{1 + c^2 + d^2},$$

with

$$a = \frac{x_{i,j+1} - x_{i,j}}{h},\ b = \frac{x_{i+1,j+1} - x_{i,j+1}}{h},\ c = \frac{x_{i+1,j+1} - x_{i+1,j}}{h},\ d = \frac{x_{i+1,j} - x_{i,j}}{h},$$

where $x_{i,j} \approx x(t_l + ih, s_l + jh)$, $i, j = 0, \ldots, n$, $h = 1/n$. The $(n-1)^2$-dimensional vector x_h corresponds to the internal components $x_{i,j}$, $i, j = 1, \ldots, n-1$, while the boundary components are set according to the given boundary conditions.

We start by conducting the same experiment described for P2D on a special instance of problem (11), where the solution is given by the *Enneper surface*, defined as $x(t,s) = u(t,s)^2 - v(t,s)^2$, $(t,s) \in [-0.5, 0.5]^2$, where u and v are implicitly defined as the unique solutions of the nonlinear equations

$$\begin{aligned} u + uv^2 - \tfrac{1}{3}u^3 - t &= 0, \\ -v - u^2 v + \tfrac{1}{3}v^3 - s &= 0, \end{aligned} \tag{12}$$

and are obtained by solving (12) to machine precision with the Newton method [1]. The discretization error was obtained by solving the discrete problem to very high accuracy with the FMINUNC routine in the MATLAB$^{\text{TM}}$ Optimization Toolbox. Therefore, we only report results for small and medium-scale problems.

In Tables 4 and 5 below, we summarize the results obtained by running Algorithm 3 as done for the P2D model problem, i.e.:

- Using the stepsize strategy (9), with $C = 4$ and $\tau_{\min} = 10^{-4}$.

- Using the stepsize strategy (10) with $c = 4$, the stepsize expansion technique, and $\tau_{\min} = 10^{-5}$ on the coarsest level.

The cycle parameters are in both cases $\nu_0 = 1$ and $\nu_1 = \nu_2 = 2$.

The results, reported in Tables 4 and 5, lead to conclusions very similar to those drawn in Section 4.4. With strategy (9), the algorithm is extremely fast, but the accuracy of the solution is bottlenecked by the lack of the expansion strategy. The inability of taking appropriately small steps as the grids are refined causes the error not to scale well with the problem size. In the case of strategy (10) coupled with stepsize expansions, on the other hand, the error seems to scale better, reducing by a factor of about 2 at each grid refinement.

`CS-GS`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$4.80\text{e}+03$	0.07	$3.52\text{e}-04$	$8.46\text{e}-04$	$1.46\text{e}-03$	$8.46\text{e}-04$
225	$1.05\text{e}+04$	0.21	$1.04\text{e}-04$	$1.54\text{e}-03$	$4.70\text{e}-03$	$1.54\text{e}-03$
961	$2.47\text{e}+04$	0.42	$2.71\text{e}-05$	$1.97\text{e}-03$	$4.72\text{e}-03$	$1.39\text{e}-03$
3969	$1.05\text{e}+05$	1.56	$6.84\text{e}-06$	$1.62\text{e}-03$	$4.18\text{e}-03$	$1.37\text{e}-03$
`CS-J`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$3.23\text{e}+03$	0.07	$3.52\text{e}-04$	$3.53\text{e}-04$	$7.26\text{e}-04$	$3.53\text{e}-04$
225	$7.13\text{e}+03$	0.22	$1.04\text{e}-04$	$1.48\text{e}-03$	$4.46\text{e}-03$	$1.48\text{e}-03$
961	$2.36\text{e}+04$	0.43	$2.71\text{e}-05$	$1.75\text{e}-03$	$4.37\text{e}-03$	$1.75\text{e}-03$
3969	$9.12\text{e}+04$	1.42	$6.84\text{e}-06$	$1.31\text{e}-03$	$3.02\text{e}-03$	$1.30\text{e}-03$

Table 4: Performance of Algorithm 3 with `CS-GS` and `CS-J` smoothing on the minimum surface problem, using the stepsize strategy (9).

`CS-GS`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$1.17\text{e}+04$	0.14	$3.52\text{e}-04$	$3.53\text{e}-04$	$7.56\text{e}-04$	$3.53\text{e}-04$
225	$3.86\text{e}+04$	0.44	$1.04\text{e}-04$	$1.00\text{e}-03$	$2.46\text{e}-03$	$5.61\text{e}-04$
961	$1.41\text{e}+05$	1.30	$2.71\text{e}-05$	$5.28\text{e}-04$	$1.34\text{e}-03$	$2.65\text{e}-04$
3969	$5.43\text{e}+05$	5.04	$6.84\text{e}-06$	$3.13\text{e}-04$	$7.60\text{e}-04$	$9.43\text{e}-05$
`CS-J`						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$1.01\text{e}+04$	0.14	$3.52\text{e}-04$	$3.52\text{e}-04$	$7.56\text{e}-04$	$3.52\text{e}-04$
225	$2.62\text{e}+04$	0.37	$1.04\text{e}-04$	$1.40\text{e}-03$	$3.43\text{e}-03$	$9.96\text{e}-04$
961	$1.31\text{e}+05$	1.36	$2.71\text{e}-05$	$6.26\text{e}-04$	$1.58\text{e}-03$	$4.99\text{e}-04$
3969	$4.91\text{e}+05$	4.89	$6.84\text{e}-06$	$2.66\text{e}-04$	$7.04\text{e}-04$	$2.15\text{e}-04$

Table 5: Performance of Algorithm 3 with `CS-GS` and `CS-J` smoothing on the minimum surface problem, using the new stepsize strategy.

Though results in Table 5 show a substantial improvement over those in Table 4, in that a reduction in the error is observed as the grid is refined, it is

apparent that the computed error is already losing an order of magnitude with respect to DE on the second-to-coarsest grid.

In order to understand this phenomenon, and to devise a more suitable strategy, we take a step back and conduct an experiment to assess the smoothing capabilities of the multilevel scheme on this problem. For this purpose, we consider the special case where $S = [0, 1]^2$ and the boundary conditions are set to zero, so that the exact solution is given by the null vector and the current guess coincides with the error. We start from the initial guess

$$x_{i,j} = \frac{1}{4}\left(\sin\left(\frac{2\pi i}{n}\right) + \sin\left(\frac{16\pi i}{n}\right)\right)\left(\sin\left(\frac{2\pi j}{n}\right)\sin\left(\frac{16\pi j}{n}\right)\right).$$

We run a single `ML/CS` v-cycle using a `CS-J` smoother, on a problem with size $n_h = 3969$. The smallest grid corresponds to a problem of size $n_h = 49$. We choose $\Delta_h^0 = 1$, and employ the expansion strategy described in Section 4.2. By performing some preliminary experiments, we immediately noticed that on this example a higher number of Coordinate Search iterations was needed to attain a clear and significant smoothing effect, and as such we set $\nu_1 = \nu_2 = 10$ for this test.

Results in Figure 5 show a behaviour very similar to that observed on the one-dimensional Poisson problem. The pre-smoothing phase damps the oscillatory components of the error, leading to a new error which is smaller by one order of magnitude and exhibits a smoother profile. The correction step approximates the opposite of the error vector and reduces the error by one further order of magnitude, but in doing so re-introduces some high-frequency modes, which are then taken care of by the post-smoothing phase.

In Figure 6, we compare the behaviour of the stepsize strategies (9) and (10) in a `FML/CS` cycle. Using a `CS-J` smoother and the classical surrogate model (5), we run Algorithm 3 on the same problem as above, with the same initial guess and cycle parameters. It can be seen how the stepsize strategy (10), in conjunction with the expansion steps of Section 4.2, yields a very small and smooth final error, while the strategy (9) loses over two orders of magnitude in norm and is unable to reconstruct a smooth error profile.

Finally, in Figure 7, we compare the errors obtained by Algorithm 3 with `CS-GS` and `CS-J` smoothing, using the alternative surrogate (7) and employing the new stepsize strategy. It is apparent how the error vectors both have the same shape and order of magnitude, but the solution corresponding to the `CS-J` strategy appears smoother and closer to the result obtained with the classical model (5), a phenomenon arguably related to the fact that in the second case the approximation of g_h introduces a larger error with respect to the exact gradient.

We now return to the example of Tables 4 and 5, and try to adjust the cycle parameters taking into account the results discussed above. Specifically, we now employ a more demanding strategy with $\nu_1 = \nu_2 = 10$, and $c = 8$.

Results are reported in Table 6, and show that the error on the solution indeed reaches the order of magnitude of the discretization error on each grid, as expected from a multigrid scheme. Of course, the computational burden is

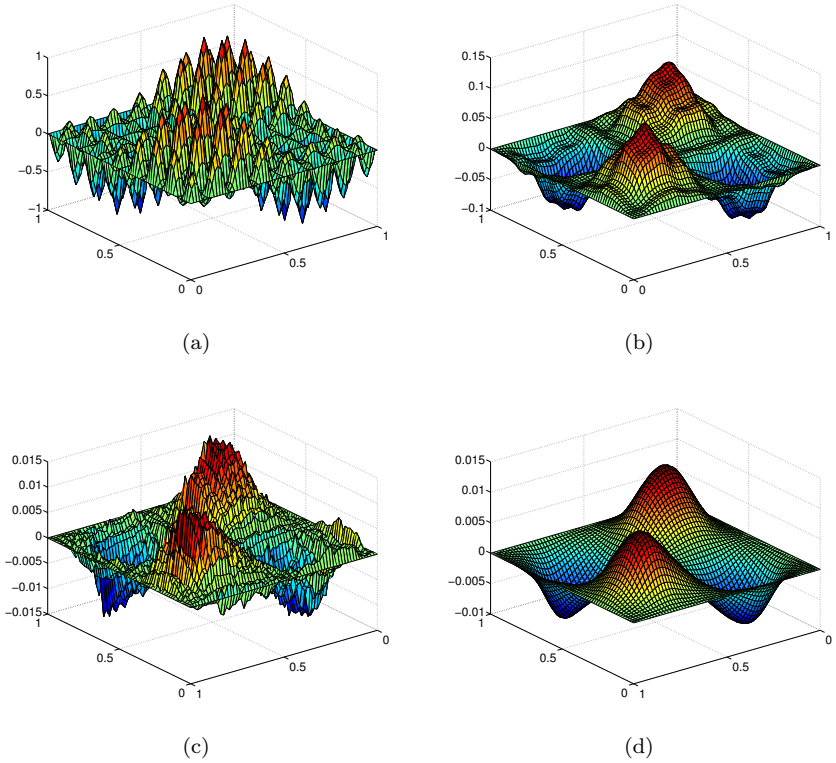
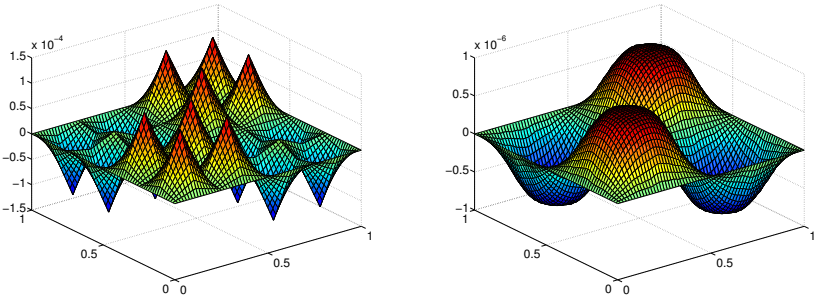


Figure 5: Error correction in a single v-cycle with `ML/CS` and `CS-J` smoothing: initial error (a), error after the pre-smoothing (b), after the correction (c), and after the post-smoothing (d).

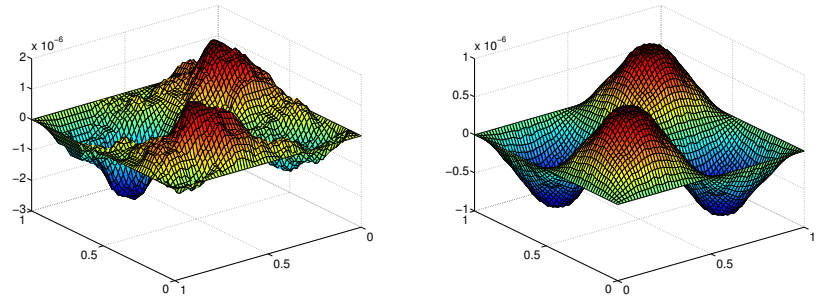
considerably higher because of the high number of smoothing iterations needed to damp all the oscillatory components of the error.

The results above also confirm that the impact of using the alternative surrogate (7) with an inexact gradient is very limited, particularly when using the `CS-J` strategy, since solutions of the same accuracy are found with both the classical and derivative-free model. Finally, the `CS-J` smoothing strategy appears to yield slightly better results compared to `CS-GS`. Note however that, compared to the quadratic case, the difference is generally less pronounced on this example, arguably due to the fact that `CS-J` computes an approximation g_h which is not anymore an exact approximation of $\nabla f(x_h^{\nu_1-1})$, and therefore partially loses its advantage over `CS-GS`.



(a) (b)

Figure 6: Error correction in a `FML/CS` cycle, with the strategy (9) (a) and with the strategy (10) (b), obtained with the classical surrogate model and `CS-J` smoothing.



(a) (b)

Figure 7: Error correction in a `FML/CS` cycle with the strategy (10) using the model (7): `CS-J` smoothing (a) vs. `CS-GS` smoothing (b).

5 Conclusions

In this paper, we presented several algorithmic improvements on the multilevel derivative-free optimization method proposed in [6]. We showed how multilevel schemes based on Directional Direct Search methods can be enhanced by appropriate stepsize strategies, to obtain a procedure possessing the same accuracy properties of traditional multigrid methods, overcoming the typical limitations of Direct Search in this regard.

The obtained results suggest that the topic of improving derivative-free methods by multigrid-inspired techniques deserves further investigation. In par-

CS-GS						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$1.17e+04$	0.14	$3.52e-04$	$3.53e-04$	$7.56e-04$	$3.53e-04$
225	$7.57e+04$	0.79	$1.04e-04$	$1.27e-04$	$3.02e-04$	$1.20e-04$
961	$4.24e+05$	4.19	$2.71e-05$	$6.84e-05$	$1.70e-04$	$4.03e-05$
3969	$1.99e+06$	18.9	$6.84e-06$	$3.06e-05$	$7.10e-05$	$1.20e-05$
CS-J						
Size	Fevals	Time	$\|DE\|_{L_2}$	$\|x - x^*\|_{L_2}$	$\|x - x^*\|_\infty$	$\|x_c - x^*\|_{L_2}$
49	$1.01e+04$	0.13	$3.52e-04$	$3.52e-04$	$7.56e-04$	$3.52e-04$
225	$7.57e+04$	0.85	$1.04e-04$	$1.55e-04$	$3.21e-04$	$1.81e-04$
961	$4.24e+05$	4.27	$2.71e-05$	$5.43e-05$	$1.20e-04$	$6.49e-05$
3969	$1.84e+06$	18.1	$6.84e-06$	$1.66e-05$	$3.79e-05$	$1.99e-05$

Table 6: Performance of Algorithm 3 on the minimum surface problem, with the new stepsize strategy and additional smoothing iterations.

ticular, algorithms using different multilevel schemes, along with applications to imaging problems [3, 4], are currently being investigated and will be the subject of a future work.

References

[1] B. M. Averick and J. J. Moré. The Minpack-2 test problem collection. Technical report, Argonne National Laboratory, Argonne, Illinois, USA, 1991.

[2] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2nd edition, 1999.

[3] R. H. Chan and K. Chen. A multilevel algorithm for simultaneously denoising and deblurring images. *SIAM Journal on Scientific Computing*, 32 (2):1043–1063, 2010.

[4] T. F. Chan and K. Chen. An optimization-based multilevel algorithm for total variation image denoising. *Multiscale Modeling and Simulation*, 5(2): 615–645, 2006.

[5] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, 2009.

[6] E. Frandi and A. Papini. Coordinate search algorithms in multilevel optimization. *Optimization Methods and Software*, DOI: 10.1080/10556788.2013.841691, 2013.

[7] S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19 (1):414–444, 2008.

[8] S. Gratton, M. Mouffe, A. Sartenaer, Ph. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. *Optimization Methods and Software*, 25(3):359–386, 2010.

[9] L. Grippo and M. Sciandrone. Nonmonotone derivative-free methods for nonlinear equations. *Computational Optimization and Applications*, 37: 297–328, 2007.

[10] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[11] R. M. Lewis and S. G. Nash. Using inexact gradients in a multilevel optimization algorithm. *Computational Optimization and Applications*, 56(1): 39–61, 2013.

[12] S. G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14:99–116, 2000.

[13] C. J. Price and Ph. L. Toint. Exploiting problem structure in pattern search methods for unconstrained optimization. *Optimization Methods and Software*, 3:479–491, 2006.

[14] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[15] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[16] L. N. Vicente. Worst-case complexity of direct search. *EURO Journal on Computational Optimization*, 1(1-2):143–153, 2013.