Numerical Experience with a Recursive Trust-Region Method for Multilevel Nonlinear Optimization

by S. Gratton[1], A. Sartenaer[2] and Ph. L. Toint[2]

Report 06/01 2 June 2006

[1] CERFACS,
av. G. Coriolis, Toulouse, France,
Email: serge.gratton@cerfacs.fr

[2] Department of Mathematics,
University of Namur,
61, rue de Bruxelles, B-5000 Namur, Belgium,
Email: annick.sartenaer@fundp.ac.be, philippe.toint@fundp.ac.be

Numerical Experience with a Recursive Trust-Region Method for Multilevel Nonlinear Optimization

Serge Gratton, Annick Sartenaer and
Philippe L. Toint

2 June 2006

Abstract

We consider an implementation of the recursive multilevel trust-region algorithm proposed by Gratton, Sartenaer and Toint (2004), and provide significant numerical experience on multilevel test problems. A suitable choice of the algorithm's parameters is identified on these problems, yielding a very satisfactory compromise between reliability and efficiency. The resulting default algorithm is then compared to alternative optimization techniques such as mesh refinement and direct solution of the fine-level problem. The sensibility of the default variant with respect to most important algorithmic parameters is finally investigated.

Keywords: nonlinear optimization, multilevel problems, simplified models, recursive algorithms, numerical performance

1 Introduction

In a recent paper, Gratton et al. (2004) discuss preliminary experimental efficiency and convergence properties of a new recursive multilevel trust-region algorithm for unconstrained optimization, which is partly inspired by multigrid techniques in partial differential equations (see Briggs, Henson and McCormick, 2000, for an introduction to this topic) and similar ideas in linesearch-based optimization methods by Fisher (1998) or Nash (2000) and Lewis and Nash (2005). The main feature of the new method is to allow the exploitation, in a trust-region framework, of the fact that many large-scale optimization problems have a hierarchy of different descriptions, possibly involving different number of variables. This is for instance the case for applications that arise in an infinite-dimensional context and are subsequently discretized on a hierarchy of fine to coarse meshes. Problems of this type are important and do arise in practice, for instance in optimal surface design, optimal control, and variational data assimilation (Fisher, 1998), e.g. for weather forecasting, which is our main motivation. The purpose of our paper is to present the numerical experience gained so far with a particular implementation of the considered algorithm (and some variants) applied to a few significant test problems, and to illustrate what we believe is the strong potential of methods of this type.

We start by recalling the definition and motivation of recursive trust-region methods in Section 2. Section 3 then elaborates on the practical description of the algorithmic variants analyzed in this paper. The numerical experiments are then described in Section 4 which contains both a description of the test problems used and of the results obtained with several variants. These variants are compared between themselves and with more standard implementations of Newton's method such as mesh-refinement techniques. Some conclusions and perspectives are finally presented in Section 5.

2 A recursive multilevel trust-region method

We consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.1}$$

where f is a twice-continuously differentiable objective function which maps $\mathbb{R}^n$ into $\mathbb{R}$ and is bounded below. The trust-region method[(1)] that we consider here is iterative, in the sense that, given an initial point x_0, it produces a sequence $\{x_k\}$ of iterates. At each iterate, such a method builds a model of $f(x)$ around x_k which is assumed to be adequate in a sphere of radius $\Delta_k > 0$ centered at x_k, called the *trust region*. A step s_k is then computed that induces a sufficient reduction in the model inside the trust region. The objective function is computed at the trial point, $x_k + s_k$, and this trial point is accepted as the next iterate if and only if ρ_k, the ratio of achieved reduction (in the objective function) to predicted reduction (in its local model), is larger than a small positive constant η_1. The radius of the trust region is finally updated: it is decreased if the trial point is rejected and left unchanged or increased if ρ_k is sufficiently large.

Many practical trust-region algorithms, including that presented here, use a *quadratic model*. A sufficient decrease in this model inside the trust region is then obtained by (approximately) solving

$$\min_{\|s\| \leq \Delta_k} m_k(x_k + s) = f(x_k) + \langle g_k, s \rangle + \tfrac{1}{2} \langle s, H_k s \rangle, \tag{2.2}$$

where $g_k \stackrel{\text{def}}{=} \nabla f(x_k)$, H_k is a symmetric $n \times n$ approximation of $\nabla^2 f(x_k)$, $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\| \cdot \|$ the Euclidean norm.

An *alternative model* may however be chosen if a hierarchy of problem expressions is known. To be more specific, suppose that a collection of functions $\{f_i\}_{i=0}^r$ is available, each f_i being a twice-continuously differentiable function from $\mathbb{R}^{n_i}$ to $\mathbb{R}$ (with $n_i \geq n_{i-1}$). We assume that $n_r = n$ and $f_r(x) = f(x)$ for all $x \in \mathbb{R}^n$, giving back our original problem. We also make the assumption that f_i is "more costly" to minimize than f_{i-1} for each $i = 1, \ldots, r$. This is typically the case if the f_i represent increasingly finer discretizations of the same infinite-dimensional objective. To fix terminology, we will refer to a particular i as a *level*. We use the first subscript i in all subsequent subscripted symbols to denote a quantity corresponding to the i-th level (meaning in particular, if applied to a vector, that this vector belongs to $\mathbb{R}^{n_i}$).

[(1)] A comprehensive description and analysis of trust-region methods may be found in Conn, Gould and Toint (2000).

The construction of a cheap alternative model is then based on the idea of using f_{r-1} to derive a lower-level model h_{r-1} of $f(x) = f_r(x)$ in the neighbourhood of the current iterate. This model is then used to define the step in the trust-region algorithm whenever possible. If more than two levels are available ($r > 1$), we can apply this idea recursively, the approximation process stopping at level 0, where the quadratic model is always used.

Some relation must of course exist between the variables of two successive functions of the collection set $\{f_i\}_{i=0}^r$ to ensure that f_{i-1} can be useful when minimizing f_i, $i = 1, \ldots, r$. We thus assume that, for each $i = 1, \ldots, r$, there exist a full-rank linear operator R_i from $\mathbb{R}^{n_i}$ into $\mathbb{R}^{n_{i-1}}$ (the restriction) and another full-rank operator P_i from $\mathbb{R}^{n_{i-1}}$ into $\mathbb{R}^{n_i}$ (the prolongation) such that

$$\sigma_i P_i = R_i^T, \tag{2.3}$$

for some known constant $\sigma_i > 0$, where P_i and R_i are interpreted as restriction and prolongation between a fine and a coarse grid (see, for instance, Briggs et al., 2000). At variance with the convergence analysis of Gratton et al. (2004) which is formally simplified by choosing $\sigma_i = 1$ for every i, we no longer make this assumption below, but assume instead that the restriction operators are normalized to ensure $\|R_i\| = 1$.

Let us now explain how we construct a lower-level model h_{i-1} based on f_{i-1}. Consider some iteration k at level i (with current iterate $x_{i,k}$[(2)]). We first need to restrict $x_{i,k}$ to create the starting iterate $x_{i-1,0}$ at level $i-1$, that is, $x_{i-1,0} = R_i x_{i,k}$. The lower-level model can then be defined as the function

$$h_{i-1}(x_{i-1,0} + s_{i-1}) \stackrel{\text{def}}{=} f_{i-1}(x_{i-1,0} + s_{i-1}) + \langle v_{i-1}, s_{i-1} \rangle, \tag{2.4}$$

where $v_{i-1} = R_i g_{i,k} - \nabla f_{i-1}(x_{i-1,0})$ with $g_{i,k} \stackrel{\text{def}}{=} \nabla h_i(x_{i,k})$. By convention, we set $v_r = 0$, such that, for all s_r,

$$h_r(x_{r,0} + s_r) = f_r(x_{r,0} + s_r) = f(x_{r,0} + s_r) \;\text{ and }\; g_{r,k} = \nabla h_r(x_{r,k}) = \nabla f(x_{r,k}).$$

The model h_{i-1} thus results from a modification of f_{i-1} by a linear term that enforces the relation $g_{i-1,0} = \nabla h_{i-1}(x_{i-1,0}) = R_i g_{i,k}$. This first-order modification[(3)] ensures that the first-order behaviours of h_i and h_{i-1} are coherent in a neighbourhood of $x_{i,k}$ and $x_{i-1,0}$, respectively. Indeed, if s_i and s_{i-1} satisfy $s_i = P_i s_{i-1}$, we then have that

$$\langle g_{i,k}, s_i \rangle = \langle g_{i,k}, P_i s_{i-1} \rangle = \frac{1}{\sigma_i} \langle R_i g_{i,k}, s_{i-1} \rangle = \frac{1}{\sigma_i} \langle g_{i-1,0}, s_{i-1} \rangle, \tag{2.5}$$

where we have also used (2.3). We refer the interested reader to Gratton et al. (2004) for further details.

Our task, when entering level $i = 0, \ldots, r$, is then to (locally) minimize h_i starting from $x_{i,0}$. At iteration k of this minimization, we first choose, at iterate $x_{i,k}$, either the model $h_{i-1}(x_{i-1,0} + s_{i-1})$ (given by (2.4)) or

$$m_{i,k}(x_{i,k} + s_i) = h_i(x_{i,k}) + \langle g_{i,k}, s_i \rangle + \tfrac{1}{2} \langle s_i, H_{i,k} s_i \rangle \tag{2.6}$$

(2) The second subscript, k, is the index of the current iteration within level i.

(3) It is not unusual to find the first-order modification (2.4) in multigrid applications in the context of the "full approximation scheme", where it is usually called the "tau correction" (see, for instance, Chapter 3 of Briggs et al., 2000, or Hemker and Johnson, 1987).

where the latter is the usual truncated Taylor series in which $H_{i,k}$ is a symmetric $n_i \times n_i$ approximation to the second derivatives of h_i at $x_{i,k}$. If the lower-level model h_{i-1} is used, the problem to be (possibly approximately) solved becomes

$$\min_{\|s_{i-1}\|_{i-1} \leq \Delta_{i,k}} h_{i-1}(x_{i-1,0} + s_{i-1}), \tag{2.7}$$

starting from $x_{i-1,0}$ and where $\|\cdot\|_{i-1}$ is a level-dependent norm defined by $\|\cdot\|_r = \|\cdot\|$ and $\|s_{i-1}\|_{i-1} = \|P_i s_{i-1}\|_i$ $(i = 1, \ldots, r)$.

We now describe our recursive multilevel trust-region algorithm (see Algorithm RMTR on page 5). We assume, in this algorithm, that the prolongations P_i and the restrictions R_i are known, as well as the functions $\{f_i\}_{i=0}^r$. We use the constants $\kappa_{\rm g}$, η_1, η_2, γ_1 and γ_2 satisfying the conditions $\kappa_{\rm g} \in (0, \min[1, \min_i \|R_i\|])$, $0 < \eta_1 \leq \eta_2 < 1$, and $0 < \gamma_1 \leq \gamma_2 < 1$. An initial trust-region radius for each level, $\Delta_i^{\rm s} > 0$, is also defined, as well as level-dependent gradient-norm tolerances, $\epsilon_i^{\rm g} \in (0, 1)$, and trust-region tolerances, $\epsilon_i^{\Delta} \in (0, 1)$, for $i = 0, \ldots, r$. The algorithm's initial data consists of the level index i $(0 \leq i \leq r)$, a starting point $x_{i,0}$, the gradient $g_{i,0}$ at this point, the radius Δ_{i+1} of the level $i+1$ trust region, the tolerances $\epsilon_i^{\rm g}$ and ϵ_i^{Δ}, and $\Delta_i^{\rm s}$.

Further motivation for this algorithm can be found in Gratton et al. (2004), together with a proof that, under reasonable assumptions, every limit point of the sequence of produced iterates must be a first-order critical point, in the sense that $\lim_{k\to\infty} \nabla f_r(x_{r,k}) = 0$. In particular, the functions f_i must have uniformly bounded Hessians for $i = 0, \ldots, r$. A few additional comments are however helpful.

1. The minimization of $f(x) = f_r(x_r) = h_r(x_r)$ (up to the gradient-norm tolerance $\epsilon_r^{\rm g} < \|\nabla f_r(x_{r,0})\|$) is achieved by calling RMTR(r, $x_{r,0}$, $\nabla f_r(x_{r,0})$, $\Delta_{r+1,0}$, $\epsilon_r^{\rm g}$, ϵ_r^{Δ}, $\Delta_r^{\rm s}$), for some starting point $x_{r,0}$ and initial trust-region radius $\Delta_r^{\rm s}$, and where we define $\Delta_{r+1,0} = \infty$. For coherence of notations, we thus view this call as being made from some (virtual) iteration 0 at level $r+1$.

2. The two conditions in (2.8) ensure that the model h_{i-1} is considered only when it is potentially useful. Indeed, the first condition requires that $\|g_{i-1,0}\| = \|R_i g_{i,k}\|$ be large enough compared to $\|g_{i,k}\|$, to avoid that the current iterate appears to be first-order critical for h_{i-1} in $\mathbb{R}^{n_{i-1}}$ while it is not for h_i in $\mathbb{R}^{n_i}$. The second condition imposes that first-order criticality for h_{i-1} has not been achieved yet.

3. The "sufficient decrease" in the model (2.6) imposed in Step 3 means, as usual for trust-region methods, that the step $s_{i,k}$ must satisfy two specific conditions (see Chapter 6 of Conn et al., 2000). The first is known as the *Cauchy point condition* and imposes sufficient decrease relative to the local first-order behaviour of the objective function. It requires that

$$m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k}) \geq \kappa_{\rm red} \|g_{i,k}\| \min\left[\frac{\|g_{i,k}\|}{1 + \|H_{i,k}\|}, \Delta_{i,k}\right] \tag{2.11}$$

   for some constant $\kappa_{\rm red} \in (0, 1)$. If one desires convergence to second-order critical points, i.e., first-order critical points at which the Hessian of the objective function is positive

Algorithm 2.1: RMTR(i, $x_{i,0}$, $g_{i,0}$, Δ_{i+1}, ϵ_i^{g}, ϵ_i^{Δ}, Δ_i^{s})

Step 0: Initialization.
Compute $v_i = g_{i,0} - \nabla f_i(x_{i,0})$ and $h_i(x_{i,0})$. Set $\Delta_{i,0} = \min[\Delta_i^{\mathrm{s}}, \Delta_{i+1}]$ and $k = 0$.

Step 1: Model choice.
Go to Step 3 (Taylor step) if $i = 0$ or if the condition

$$\|R_i g_{i,k}\| \geq \kappa_{\mathrm{g}} \|g_{i,k}\| \ \text{ and } \ \|R_i g_{i,k}\| > \epsilon_{i-1}^{\mathrm{g}} \tag{2.8}$$

fails. Otherwise, choose to go to Step 2 (recursive step) or to Step 3.

Step 2: Recursive step computation.
Call RMTR($i-1$, $R_i x_{i,k}$, $R_i g_{i,k}$, $\Delta_{i,k}$, $\epsilon_{i-1}^{\mathrm{g}}$, ϵ_{i-1}^{Δ}, $\Delta_{i-1}^{\mathrm{s}}$), yielding an approximate solution $x_{i-1,*}$ of (2.7). Then define $s_{i,k} = P_i(x_{i-1,*} - R_i x_{i,k})$, set $\delta_{i,k} = h_{i-1}(R_i x_{i,k}) - h_{i-1}(x_{i-1,*})$ and go to Step 4.

Step 3: Taylor step computation.
Choose $H_{i,k}$ and compute a step $s_{i,k}$ that sufficiently reduces the model (2.6) and such that $\|s_{i,k}\|_i \leq \Delta_{i,k}$. Set $\delta_{i,k} = m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k})$.

Step 4: Acceptance of the trial point.
Compute $h_i(x_{i,k}+s_{i,k})$ and define $\rho_{i,k} = (h_i(x_{i,k}) - h_i(x_{i,k}+s_{i,k}))/\delta_{i,k}$. If $\rho_{i,k} \geq \eta_1$, then define $x_{i,k+1} = x_{i,k} + s_{i,k}$; otherwise define $x_{i,k+1} = x_{i,k}$.

Step 5: Termination.
Compute $g_{i,k+1}$. If $\|g_{i,k+1}\|_\infty \leq \epsilon_i^{\mathrm{g}}$ or $\|x_{i,k+1} - x_{i,0}\|_i > (1 - \epsilon_i^{\Delta})\Delta_{i+1}$, then return with the approximate solution $x_{i,*} = x_{i,k+1}$.

Step 6: Trust-region radius update.
Set

$$\Delta_{i,k}^{+} \in \begin{cases} [\Delta_{i,k}, +\infty) & \text{if } \rho_{i,k} \geq \eta_2, \\ [\gamma_2 \Delta_{i,k}, \Delta_{i,k}] & \text{if } \rho_{i,k} \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_{i,k}, \gamma_2 \Delta_{i,k}] & \text{if } \rho_{i,k} < \eta_1, \end{cases} \tag{2.9}$$

and

$$\Delta_{i,k+1} = \min\left[\Delta_{i,k}^{+}, \Delta_{i+1} - \|x_{i,k+1} - x_{i,0}\|_i\right]. \tag{2.10}$$

Increment k by one and go to Step 1.

semi-definite, a second condition, the *eigen point condition*, is required. It imposes that, if $\tau_{i,k}$, the smallest eigenvalue of $H_{i,k}$, is negative, then

$$m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k}) \geq \kappa_{\mathrm{eip}} |\tau_{i,k}| \min[\tau_{i,k}^2, \Delta_{i,k}^2] \tag{2.12}$$

for some constant $\kappa_{\mathrm{eip}} \in (0, \frac{1}{2})$. However, we will see below that we might settle for a weaker condition, obviously leading to a weaker result.

4. The termination test $\|x_{i,k+1} - x_{i,0}\|_i > (1 - \epsilon_i^{\Delta})\Delta_{i+1}$ in Step 5 as well as (2.10) guarantee

that iterates at a lower level in a recursion remain in the trust region defined at the calling level.

5. Iteration k at level i is said to be *successful* if $\rho_{i,k} \geq \eta_1$.

Clearly, our algorithmic description so far leaves a number of practical choices unspecified, and is best viewed at this stage as a theoretical shell which potentially contains both efficient and inefficient algorithms. Can efficient algorithms be found in this shell? It is the purpose of the next section to investigate this question.

3 A practical algorithm

In this section, we provide the missing details for the particular implementations whose numerical performance is reported in this paper. These details are of course influenced by our focus on discretized problems, where the different levels correspond to different discretization grids, from coarser to finer.

3.1 Taylor iterations: smoothing and solving

The most important issue is how to enforce sufficient decrease at Taylor iterations, that is, when Step 3 is executed. At the coarsest level ($i = 0$), the cost of fully minimizing (2.6) inside the trust region remains small, since the subproblem is of low dimension. We thus solve the subproblem using the well-known method by Moré and Sorensen (1979) (see also Section 7.3 in Conn et al., 2000), whose cost is dominated by that of a small number of small-scale Cholesky factorizations and is thus very acceptable. At finer levels ($i > 0$), we consider two possibilities for the Taylor step computation. The first is based on the conjugate-gradient method and the second uses an adaptation of multigrid smoothing techniques, as explained below.

A first obvious way to compute a Taylor step is to use the TCG (Truncated Conjugate-Gradient) algorithm designed for the standard trust-region algorithm for large-scale optimization (Steihaug (1983), Toint (1981), see also Section 7.5 in Conn et al. (2000)). This algorithm is known to ensure (2.11) and can be viewed as a conjugate-gradient process which is applied regardless of the positive definiteness of the Hessian of the model, finding a useful point at the boundary of the trust-region when a negative curvature is encountered or when the iterate leaves the trust region.

Our second strategy for the Taylor step computation is based on multigrid solvers which constitute a very interesting alternative to conjugate gradient[(4)] in our context of discretized problems on successively finer grids. The main characteristics of multigrid algorithms (we refer the reader to Briggs et al., 2000 for an excellent introduction) are based on the observation that different "frequencies" are present in the solution of the finest grid problem (or even of the infinite-dimensional one), and become only progressively visible in the hierarchy from coarse to fine grids. Low frequencies are visible from coarse grids and up, but higher ones can

[(4)] Indeed, if one assumes that the model (2.6) is strictly convex and the trust-region radius $\Delta_{i,k}$ sufficiently large, minimizing (2.6) is equivalent to (approximately) solving the classical Newton equations $H_{i,k}s_i = -g_{i,k}$.

only be distinguished when the mesh-size of the grid becomes comparable to the frequency in question. In multigrid strategies, some algorithms, called *smoothers*, are known to very efficiently reduce the high frequency components of the error on a grid (that is, in most cases, the components whose "wavelength" is comparable to the grid's mesh-size). But these algorithms have little effect on the low frequency error components. It is observed however that such components on a fine grid appear more oscillatory on a coarser grid. They may thus be viewed as high frequency components on some coarser grid and be in turn reduced by a smoother. Moreover, this is done at a lower cost since computations on coarse grids are typically much cheaper than on finer ones. The multigrid strategy consists therefore in alternating between solving the problem on coarse grids, essentially annihilating low frequency components of the error, and on fine grids, where high frequency components are reduced (at a higher cost). This last operation is often called *smoothing* because the effect of reducing high frequency components without altering much the low frequency ones has a "smoothing effect" of the error's behaviour. We next adapt, in what follows, the multigrid smoothing technique to the computation of a Taylor step satisfying the requirements of Step 3 of Algorithm RMTR.

A very well-known multigrid smoothing technique is the Gauss-Seidel method, in which each equation of the Newton system is solved in succession[(5)]. To extend this procedure to our case, rather than successively solving equations, we perform successive one-dimensional minimizations of the model (2.6) along the coordinate axes, provided the curvature of this model along each axis is positive. More precisely, if j is an index such that the jth diagonal entry of $H_{i,k}$ is strictly positive, we perform the following updates

$$\alpha_j = -[g]_j/[H_{i,k}]_{jj}, \quad [s]_j \leftarrow [s]_j + \alpha_j \text{ and } g \leftarrow g + \alpha_j H_{i,k} e_{i,j},$$

which correspond to the minimization of (2.6) along the j-th axis (starting each minimization from s such that $\nabla m_{i,k}(x_{i,k} + s) = g$), where we denote by $[v]_j$ the j-th component of the vector v and by $[M]_{ij}$ the (i,j)-th entry of the matrix M, and where $e_{i,j}$ is the jth vector of the canonical basis of $\mathbb{R}^{n_i}$. This process is the well-known Sequential Coordinate Minimization (see, for instance, Ortega and Rheinboldt (1970), Section 14.6), which we abbreviate here as *SCM smoother*. Note that a SCM smoothing may consist of one or more cycles[(6)], a cycle being defined as a succession of n_i (at level i) one-dimensional minimizations of the model (i.e., one minimization along each coordinate axis).

In order to enforce convergence to first-order points, we still have to ensure that a sufficient model decrease (2.11) has been obtained within the trust region after a SCM smoother has been applied. To do so, we start the first smoothing cycle by selecting the axis corresponding to the largest component of the gradient, $g_{i,k}$, in absolute value. Indeed, if this component is the ℓ-th one and if $d_\ell = -\text{sign}([g_{i,k}]_\ell)e_{i,\ell}$, minimization of the model $m_{i,k}$ along d_ℓ within the trust region is guaranteed to yield a *Cauchy step* $\alpha_\ell d_\ell$ such that (2.11) holds (see Gratton et al. (2004) for details). Since the remaining minimizations in the first SCM smoothing cycle (and the following ones, if any) only decrease the value of the model further, (2.11) still holds for the complete step. We also check, after each cycle of a SCM smoothing, if the step obtained is outside the trust region. If this is the case, we stop the smoothing process and

[(5)] See Briggs et al., 2000, page 10, or Golub and Van Loan, 1989, page 510, or Ortega and Rheinboldt, 1970, page 214, amongst many others.

[(6)] We will discuss the number of cycles choice in Section 4.

apply a variant of the *dogleg* strategy (see Powell, 1970, or Conn et al., 2000, Section 7.5.3), by minimizing the model $m_{i,k}$ along the segment $[\alpha_\ell d_\ell, s]$ *restricted to the trust region*. The final step is then given by $\alpha_\ell d_\ell + \alpha_s(s - \alpha_\ell d_\ell)$, where α_s is the multiple of $s - \alpha_\ell d_\ell$ that achieves the minimizer.

To complete the description of our adapted smoothing technique, we need to specify what is done if a negative curvature is encountered along one of the coordinate axis during a cycle of a SCM smoothing. Assume that this axis is the j-th one, the model minimizer along e_j thus lies on the boundary of the trust region. It is then very easy to compute the associated model reduction. We then remember the *largest* of these reductions (along with the corresponding step) if negative curvature is met along more than one axis within the cycle, and this largest reduction is compared to the reduction obtained by minimizing the model along the axes with positive curvature so far. The smoothing process is stopped (i.e., no new cycle is started), and the step giving the maximum reduction is finally selected.

Note that the above SCM smoothing technique does not guarantee that the eigen point condition (2.12) holds, since it limits its exploration of the model's curvature to the coordinate axes. We merely have that

$$m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k}) \geq \tfrac{1}{2}|\mu_{i,k}|\Delta_{i,k}^2,$$

where $\mu_{i,k}$ is now the most negative diagonal element of $H_{i,k}$. As a consequence, there is little hope that the resulting algorithm can be proved convergent to second-order critical points without additional assumptions on the problem. Convergence can however be proved to first-order critical points at which the diagonal of objective function's Hessian is non-negative (see Gratton, Sartenaer and Toint, 2006).

The details of the choice between TCG and SCM for the Taylor iterations are given in Section 3.6.

3.2 Linesearch

The implementation whose numerical performance is discussed in Section 4 uses a version that combines the traditional trust-region techniques with a linesearch, in the spirit of Toint (1983, 1987), Nocedal and Yuan (1998) and Gertz (1999) (see Conn et al., 2000, Section 10.3.2). More precisely, if $\rho_{i,k} < \eta_1$ in Step 4 of Algorithm RMTR and the step is "gradient related" in the sense that

$$|\langle g_{i,k}, s_{i,k}\rangle| \geq \epsilon_{\rm gr}\|g_{i,k}\|\,\|s_{i,k}\|$$

for some $\epsilon_{\rm gr} \in (0,1)$, the step corresponding to a new iteration and a smaller trust-region radius can be computed by backtracking along $s_{i,k}$, instead of recomputing a new one using a TCG iteration or a SCM smoothing. On the other hand, if some iteration at the topmost level is successful and the minimizer of the quadratic model in the direction $s_{r,k}$ lies sufficiently far beyond the trust-region boundary, then an optional doubling of the step can be performed until the objective function stops decreasing, a strategy reminiscent of the "internal doubling" procedure of Dennis and Schnabel (1983) (see Conn et al., 2000, Section 10.5.2), or the "magical step" technique of Conn, Vicente and Visweswariah (1999) and Conn et al. (2000), Section 10.4.1. The theoretical arguments developed in these references guarantee that global convergence of the modified algorithm to first-order critical points is not altered.

3.3 Second-order and Galerkin models

The gradient correction v_{i-1} in (2.4) ensures that h_i and h_{i-1} coincide at first order (up to the constant σ_i) in the range of the prolongation operator, since

$$\langle g_{i,k}, P_i s_{i-1} \rangle = \frac{1}{\sigma_i} \langle R_i g_{i,k}, s_{i-1} \rangle = \frac{1}{\sigma_i} \langle g_{i-1,0}, s_{i-1} \rangle.$$

We can also achieve coherence of the models at second order by choosing

$$h_{i-1}(x_{i-1,0} + s_{i-1}) = f_{i-1}(x_{i-1,0} + s_{i-1}) + \langle v_{i-1}, s_{i-1} \rangle + \tfrac{1}{2} \langle s_{i-1}, W_{i-1} s_{i-1} \rangle, \tag{3.13}$$

where $W_{i-1} = R_i H_{i,k} P_i - \nabla^2 f_{i-1}(x_{i-1,0})$, since we then have that

$$\langle P_i s_{i-1}, H_{i,k} P_i s_{i-1,} \rangle = \frac{1}{\sigma_i} \langle s_{i-1}, \nabla^2 h_{i-1}(x_{i-1,0}) s_{i-1} \rangle.$$

The second-order model (3.13) is of course more costly, as the matrix W_{i-1} must be computed when starting the minimization at level $i-1$ and must also be used to update the gradient of h_{i-1} at each successful iteration at level $i-1$.

Another strategy consists to choose $f_{i-1}(x_{i-1,0} + s_{i-1}) = 0$ for all s_{i-1} in (3.13). This strategy amounts to consider the lower-level objective function as the "restricted" version of the quadratic model at the upper level (this is known as the "Galerkin approximation") and is interesting in that no evaluation of f_{i-1} is required. This is also close in spirit to the Jacobian correction terms used in the recent paper by Yavneh and Dardyk (2006). When this model is strictly convex and the trust region is large enough, one minimization in Algorithm RMTR (without premature termination) corresponds to applying a Galerkin multigrid linear solver on the associated Newton's equation. Note that this choice is allowed within the theory presented in Gratton et al. (2004), since the zero function is obviously twice-continuously differentiable, bounded below and has uniformly bounded Hessians.

3.4 Hessian of the models

Computing a model Hessian $H_{i,k}$ is often one of the heaviest task in Algorithm RMTR. Our choice in the experiments described in Section 4 is to use the exact second derivative matrix of the objective functions f_i. This is motivated by several considerations. The first is that the true Newton method remains one of the standards in large-scale nonlinear optimization algorithms, and therefore is a natural point of comparison for other techniques in this field. The second is that we would have little confidence in a method that would not perform adequately when exact derivatives are used instead of their approximations. The third is that suitable Hessian approximations schemes for multigrid optimization are so far unstudied and far from obvious. However, we have attempted to limit the associated cost with an automatic strategy that avoids recomputing the Hessian at each iteration when the gradient variations are still well predicted by the available $H_{i,k-1}$. More specifically, we choose to recompute the Hessian at the beginning of iteration (i,k) $(k > 0)$ whenever the preceding iteration was unsuccessful (i.e. $\rho_{i,k-1} < \eta_1$) or when

$$\|g_{i,k} - g_{i,k-1} - H_{i,k-1} s_{i,k-1}\| > \epsilon_H \|g_{i,k}\|,$$

where $\epsilon_H \in (0,1)$ is a small user-defined constant. Otherwise, we use $H_{i,k} = H_{i,k-1}$. A default value of $\epsilon_H = 0.15$ appears to give satisfactory results in most cases.

3.5 Prolongations and restrictions

We have chosen to define the prolongation and restriction operators P_i and R_i as follows. The prolongation is chosen as the *linear interpolation* operator, and the restriction is its transpose normalized to ensure that $\|R_i\| = 1$. These operators are never assembled, but are rather applied locally for improved efficiency. Cubic interpolation could also be used in principle, but it produces denser Galerkin models, and our experience is that the algorithm is computationally less efficient. These operators are consistent with our test problems, but we note that our algorithms (and their theoretical justifications) allow for more sophisticated choices for complex problems arising from difficult partial differential equations.

3.6 Free and fixed form recursions

An interesting feature of the RMTR framework of Gratton et al. (2004) is that its convergence properties are preserved if the minimization at lower levels ($i = 0, \ldots, r-1$) is stopped after the *first successful iteration*. The flexibility this induces allows to consider different recursion patterns, namely *fixed form* and *free form* ones. In a fixed form recursion pattern, a maximum number of successful iterations at each level is specified (like in V- and W-cycles in multigrid algorithms, see Briggs et al. (2000)). If no such premature termination is used but the minimization at each level is carried out until one of the classical termination conditions on the gradient norm and step size (see Step 5 of Algorithm RMTR) is satisfied, then the actual recursion pattern is uniquely determined by the progress of minimization at each level (hence yielding a free form recursion pattern).

In Section 4, we compare three recursion forms. In the first form, which we call V-form, the minimization at the lower levels consists in one successful SCM smoothing iteration, followed by either a successful TCG iteration or a successful recursive iteration, itself followed by a second successful SCM smoothing iteration[(7)]. The second form is called W-form and is defined as a V-form to which is added one successful TCG or recursive iteration, and a final SCM smoothing iteration. The third form is the free form recursion as explained above, in which we impose however that SCM smoothing iterations and other types of (successful) iterations alternate at all levels but the coarsest. Indeed, during our experiments, we have found this alternance very fruitful (and rather natural in the interpretation of the algorithm as an alternance of high frequency reductions and low frequency removals).

Note that for each recursion form, any remaining iteration is skipped if one of the termination conditions in Step 5 of Algorithm RMTR is satisfied.

3.7 Computing the starting point at the fine level

We also take advantage of the multilevel recursion idea to compute the starting point $x_{r,0}$ at the finest level by applying Algorithm RMTR successively at levels 0 up to $r-1$ (a starting point at the lowest level being supplied by the user). In our experiments based on regular meshes (see Section 4), the accuracy on the gradient norm that is required for termination at level $i < r$ is given by

$$\epsilon_i^g = \min(0.01, \epsilon_{i+1}^g / \nu_i^{\psi}), \tag{3.14}$$

[(7)] A the coarsest level 0, SCM smoothing iterations are skipped and recursion impossible.

where ϵ_r^g is the user-supplied gradient-accuracy requirement for the topmost level, ψ is the dimension of the underlying continuous problem and ν_i is the discretization mesh-size along one of these dimensions. Once computed, the solution at level i is then prolongated to level $i+1$ using *cubic interpolation*.

3.8 Constants choice and recursive termination thresholds

We conclude the description of our practical algorithm by specifying our choice for the constants, the level-dependent gradient thresholds ϵ_i^g and the trust-region boundary thresholds ϵ_i^Δ that appear in Algorithm RMTR. We set

$$\kappa_g = 0.01, \quad \eta_1 = 0.01, \quad \eta_2 = 0.95, \quad \gamma_1 = 0.05 \text{ and } \gamma_2 = 1.00, \tag{3.15}$$

as this choice appears most often appropriate. The value 1 is also often satisfactory for the Δ_i^s. The gradient thresholds are chosen according to the rule (3.14). The role of the tolerances ϵ_i^Δ on the steplength compared to the trust-region boundary (see Step 5 in Algorithm RMTR) is less crucial. A default choice of 0.05 seems adequate and is used in our tests.

4 Numerical tests

The algorithm described above has been coded in MATLAB© (Release 7.0.0) and all experiments below were run on a Dell© computer with 2 GBytes of RAM.

4.1 Test problems

We have considered a set of minimization problems in infinite-dimensional spaces, involving differential operators. These operators are discretized on a hierarchy of regular grids such that the coarse grid at level $i-1$ is defined by taking every-other point in the grid at level i: the ratio between the grid spacing of two consecutive levels in each coordinate direction is therefore 2. The grid transfer operators P_i are defined as in classical geometric multigrid settings, using interpolation operators. The restriction operators R_i are such that (2.3) holds, where $\sigma_i = \|P_i\|^{-1}$. Ensuring this condition requires the knowledge of $\|P_i\|^{-1}$, which is easily done for 1D problems by a direct computation using an eigenvalue solver. For dimension 2 or 3, the grid transfer operators consist in applying the 1D operators in the direction of each coordinate axis (see Section A of the Appendix).

Given this hierarchy of discretized problems, we are initially interested in the solution of the problem on the finest grid. All algorithms discussed below are applied to this problem.

Some of the test cases presented here are very close to the ones in Nash (2000) and Lewis and Nash (2005): they are convex minimization problems whose solution is a 1D or 2D function. We also additionally introduce a quadratic 3D problem, a 2D nonconvex problem, an optimal control problem and an inverse problem. We denote by S_2 and S_3 respectively the unit square and cube: $S_2 = [0,1]\times[0,1] = \{(x,y), 0 \le x \le 1, 0 \le y \le 1\}$ and $S_3 = [0,1]\times[0,1]\times[0,1] = \{(x,y,z), 0 \le x \le 1, 0 \le y \le 1, 0 \le z \le 1\}$. Some of the objective functions of the test problems take the form of an integral over an infinite-dimensional space, which we approximate using a straightforward averaging scheme: for example, for 2D problems, the

contribution of a square of vertices (x_i, y_i), (x_{i+1}, y_i), (x_{i+1}, y_{i+1}) and (x_i, y_{i+1}) is $(u_{i,j} + u_{i+1,j} + u_{i+1,j+1} + u_{i,j+1})h^2/4$, where $u_{i,j}$ is the value of the function to be integrated at vertex (x_i, y_i), and h is the grid spacing in the $x-$ and $y-$directions.

4.1.1 DN: a Dirichlet-to-Neumann transfer problem

Let S be the square $[0, \pi] \times [0, \pi]$ and let Γ be its lower edge defined by $\{(x, y), 0 \leq x \leq \pi, y = 0\}$. The Dirichlet-to-Neumann transfer problem (Lewis and Nash, 2005) consists in finding the function $a(x)$ defined on $[0, \pi]$, that minimizes

$$\int_0^\pi \left(\partial_y u(x, 0) - \phi(x)\right)^2 dx,$$

where $\partial_y u$ is the partial derivative of u with respect to y, and where u is the solution of the boundary value problem

$$\begin{array}{rcll} \Delta u & = & 0 & \text{in } S, \\ u(x, y) & = & a(x) & \text{on } \Gamma, \\ u(x, y) & = & 0 & \text{on } \partial S \backslash \Gamma. \end{array}$$

The problem is a 1D minimization problem, but the computations of the objective function, gradient and Hessian involve a partial differential equation in 2D. To introduce oscillatory components in the solution, we define $\phi(x) = \sum_{i=1}^{15} \sin(i\, x) + \sin(40\, x)$. The discretization of the problem is performed by finite differences with the same grid spacing in the two directions. The discretized problem is a linear least-squares problem.

4.1.2 Q2: a simple quadratic example

We consider here the two-dimensional model problem for multigrid solvers in the unit square domain S_2

$$\begin{array}{rcl} -\Delta u(x, y) & = & f \text{ in } S_2 \\ u(x, y) & = & 0 \text{ on } \partial S_2, \end{array}$$

where f is such that the analytical solution to this problem is $u(x, y) = 2y(1 - y) + 2x(1 - x)$. This problem is discretized using a 5-point finite-difference scheme, giving linear systems $A_i x = b_i$ at level i where each A_i is a symmetric positive-definite matrix. Algorithm RMTR will be used on the variational minimization problem

$$\min_{x \in \mathbb{R}^{n_r}} \tfrac{1}{2} x^T A_r x - x^T b_r, \tag{4.16}$$

which is obviously equivalent to the linear system $A_r x = b_r$. The main purpose of this example is to illustrate that Algorithm RMTR exhibits performances similar to traditional linear multigrid solvers on a quadratic model problem.

4.1.3 Q3: a 3D quadratic example

This example is a 3D version of Example 4.1.2. We consider the differential equation

$$\begin{array}{rcll} -(1 + \sin(3\pi x)^2)\Delta u(x, y, z) & = & f & \text{in } S_3, \\ u(x, y, z) & = & 0 & \text{on } \partial S_3. \end{array}$$

The right-hand side f is chosen such that $u(x,y,z) = x(1-x)y(1-y)z(1-z)$ is the desired solution. We consider $\bar{A}_i x = \bar{b}_i$, a finite-difference discretization of the problem at level i, where the Laplacian is discretized using the standard 7-point finite-difference approximation on a uniform 3D mesh. The system is then made symmetric by multiplying $\bar{A}_i$ and $\bar{b}_i$ by a diagonal matrix D_i which is the discretization of the function $1 + \sin(3\pi x)^2$. The discretized problem reads then $A_i y = b_i$, where A_i is a symmetric positive-definite matrix and the solutions x and y are related by $D_i y = x$. As for Example 4.1.2, Algorithm RMTR is applied to the variational formulation (4.16).

4.1.4 Surf: the minimum surface problem

The domain of calculus of variation consists in finding stationary values v of integrals of the form $\int_a^b f(v, \dot{v}, x)\, dx$, where $\dot{v}$ is the first-order derivative of v. Algorithm RMTR can be applied to discretized versions of problems of this types. As a representative of these, we consider the minimum surface problem

$$\min_{v \in K} \int_0^1 \int_0^1 \left(1 + (\partial_x v)^2 + (\partial_y v)^2\right)^{\frac{1}{2}}\, dx\, dy,$$

where $K = \left\{v \in H^1(S_2) \ s.t.\ v(x,y) = v_0(x,y) \text{ on } \partial S_2\right\}$. The boundary condition v_0 is chosen as

$$v_0(x,y) = \begin{cases} f(x), & y = 0, \quad 0 \le x \le 1, \\ 0, & x = 0, \quad 0 \le y \le 1, \\ f(x), & y = 1, \quad 0 \le x \le 1, \\ 0, & x = 1, \quad 0 \le y \le 1, \end{cases}$$

where $f(x) = x(1-x)$. This convex problem is discretized using a finite element basis defined using a uniform triangulation of S_2, with same grid spacing h along the 2 coordinate directions. The basis functions are the classical P1 functions which are linear on each triangle and take value 0 or 1 at each vertex.

4.1.5 Inv: an inverse problem from image processing

We consider the image deblurring problem stated on page 130 of Vogel (2002). In this case, the columns of the unknown deblurred image are stacked into a vector f. A doubly block Toeplitz matrix T is computed using the `blur` function of Hansen's toolbox (Hansen 1994), which also yields the blurred image d. The image deblurring problem can be written

$$\min \mathcal{J}(f) \quad \text{where} \quad \mathcal{J}(f) = \frac{1}{2}\|Tf - d\|_2^2 + TV(f),$$

where $TV(f)$ is the discretization of the total variation function

$$\int_0^1 \int_0^1 \left(1 + (\partial_x f)^2 + (\partial_y f)^2\right)^{\frac{1}{2}}\, dx\, dy.$$

The discretization scheme for $TV(f)$ is the same as for the minimum surface problem. The problem is convex.

4.1.6 Opt: an optimal control problem

We consider the following optimal control problem, introduced by Borzi and Kunisch (2006), and related to the solid ignition model:

$$\min_f \mathcal{J}(u(f), f) = \int_{S_2} (u - z)^2 + \frac{\beta}{2} \int_{S_2} (e^u - e^z)^2 + \frac{\nu}{2} \int_{S_2} f^2,$$

where

$$\begin{aligned} -\Delta u + \delta e^u &= f \quad \text{in } S_2, \\ u &= 0 \quad \text{on } \partial S_2. \end{aligned}$$

This convex problem is discretized using finite differences in the square S_2. For the numerical tests, we chose the following numerical values: $\nu = 10^{-5}$, $\delta = 6.8$, $\beta = 6.8$ and $z = \frac{1}{\pi^2}$.

4.1.7 NC: a nonconvex example

We introduce the nonlinear least-squares problem

$$\min_{u,\gamma} \mathcal{J}(u, \gamma) = \int_{S_2} (u - u_0)^2 + \int_{S_2} (\gamma - \gamma_0)^2 + \int_{S_2} f^2,$$

where

$$\begin{aligned} -\Delta u + \gamma u - f_0 &= f \quad \text{in } S_2, \\ u &= 0 \quad \text{on } \partial S_2, \end{aligned}$$

with the unknown functions u and γ being defined on the unit square S_2. The functions $\gamma_0(x, y)$ and $u_0(x, y)$ are defined on S_2 by $\gamma_0(x, y) = u_0(x, y) = \sin(x(1 - x)) \sin(y(1 - y))$. The function f_0 is such that $-\Delta u_0 + \gamma_0 u_0 = f_0$ on S_2. This problem corresponds to a penalized version of a constrained optimal control problem, and is discretized using finite differences. The nonconvexity of the resulting discretized fine-grid problem has been assessed by a direct eigenvalue computation on the Hessian of the problem.

4.2 Performance of the multilevel Algorithm RMTR

In a first set of experiments, we focus on the comparison of the following three algorithms:

- the "all on finest" (**AF**) algorithm, which is a standard Newton trust-region algorithm (with TCG as subproblem solver) applied at the finest level, without recourse to coarse-level computations;
- the mesh refinement technique (**MR**), where the discretized problems are solved in turn (from the coarsest level (level 0) to the finest one (level r)), using the same standard Newton trust-region method. The starting point at level $i+1$ is obtained by prolongating (using P_{i+1}) the solution obtained at level i;
- the full multilevel (**FM**) Algorithm RMTR.

The starting point of each algorithm is a random vector of appropriate size generated, on the coarsest level (which is also the finest for Algorithm AF), by the Matlab command `rand` with initial seed set to zero. The algorithms were terminated when the infinity norm of the

gradient at the finest level was below 5×10^{-9} for Q2, Surf and Opt, below 10^{-7} for Q3, DN and NC, and below 5×10^{-5} for Inv.

The performance of these algorithms (and of their variants in Section 4.3) is analyzed in terms of number of objective function, gradient and Hessian evaluations as it is often the case in optimization, but we also provide an analysis in terms of calls to basic computational kernels (the typical relative cpu-time costs of all these kernels are reported in Section B of the Appendix). Even if they broadly confirm our analysis, we do not report the exact Matlab cpu-times for two reasons. First because they are subject to significant fluctuation depending on machine load. Secondly because, as resulting from the execution of an interpreted language, they penalize function calls more than is typical from a less experimental implementation in a compiled language.

4.2.1 The default full multilevel (FM) algorithm

Our first task has consisted in selecting reasonable values for the parameters of Algorithm RMTR. This was achieved by intensive testing on all our test cases, which indicated that excellent results in terms of computational cost are obtained when the algorithmic parameters are set as follows.

The Newton quadratic model is taken at the finest level. The Galerkin models, obtained by setting $f_i = 0$ and by ensuring the second order coherence, as in Section 3.3, are chosen on the coarse levels i for $i < r$. A fixed form recursion pattern is chosen, where a recursive iteration (Step 2) is always attempted whenever allowed by condition (2.8). Our experience shows that the best results are obtained when a W-form recursion is performed at each level (see Section 3.6). A single smoothing cycle is allowed at SCM smoothing iterations (see Section 3.1). The algorithm starts with the initialization phase described in Section 3.7.

4.2.2 Performance results on quadratic problems

We start by showing that, for the quadratic convex problems DN, Q2 and Q3, our default algorithm has the same behaviour as the linear multigrid approach on the equivalent linear system. In Table 4.1, we report the total number of smoothing cycles (for Algorithm FM) and the total number of matrix-vector products (for Algorithms MR and AF), needed at the *finest level*, to solve a quadratic problem for a given problem size. These operations are relevant because they constitute the dominant computational kernels for the three algorithms: we have already indicated the role played by smoothing cycles in Algorithm RMTR, while Hessian-vector products occur at the core of the TCG subproblem solver in both Algorithms AF and MR[(8)]. Moreover, they provide a meaningful comparison framework because one smoothing cycle requires approximately the same number of floating-point operations as a matrix-vector product (see Section B in the Appendix for a cpu-time comparison). For quadratic problems, the gradient and the cost function are cheap by-products of the TCG algorithm and of a smoothing cycle, and may thus be ignored in the comparison. The Hessian at the finest level is computed prior to the call to the algorithm, and kept in memory for the computations.

[(8)] TCG iterations are also possible in principle (for $i > 0$) in Step 3 of Algorithm FM whenever (2.8) fails. However, this situation has never been observed in our experiments.

Problem	Method	Problem size						
DN		15	31	63	127	255	511	
	FM	8	8	9	9	9	9	
	MR	19	41	49	94	177	359	
	AF	24	36	55	95	197	341	
Q2		15^2	31^2	63^2	127^2	255^2	511^2	1023^2
	FM	11	11	10	9	7	4	4
	MR	45	89	175	336	650	502	737
	AF	53	96	208	370	823	1017	1604
Q3		15^3	31^3	63^3				
	FM	17	13	9				
	MR	71	48	85				
	AF	98	124	176				

Table 4.1: Quadratic problems: number of smoothing cycles or Hessian-vector products (at finest level) versus problem size.

We observe in Table 4.1 that Algorithm FM is clearly superior to both Algorithms MR and AF. For Problem DN, these last algorithms perform almost equally bad, while the number of smoothing cycles remains very stable for Algorithm FM. For Problems Q2 and Q3, Algorithms MR and AF require significantly more matrix-vector products as dimension increases, with AF needing twice as many as MR. The behaviour of Algorithm FM is stable and even improves for large problem sizes.

For an ideal multigrid algorithm on a quadratic problem, we expect the number of smoothing cycles to be fairly independent of the mesh size and dimension. We see that this holds for Algorithm FM, which indicates that the trust-region machinery introduced in the multigrid setting does not alter this property.

4.2.3 Performance results on non-quadratic problems

For non-quadratic problems, the significant computational kernels now include objective function, gradient and Hessian evaluations, in addition to those already accounted for in the case of quadratic problems. The results are shown in Tables 4.2 to 4.5. The problem size at the finest level is the largest possible which could be solved in less than one day on our computer and within the limits of available memory. Observe first that, in terms of computational cost, the rightmost columns of the tables are the most significant. Also note that the results no longer include Algorithm AF, whose performance is significantly worse, as shown above.

A general comment is that for Problems Surf, Inv and Opt, the number of smoothing cycles in FM is drastically smaller than the number of matrix-vector products in MR. For Problem NC, MR is performing very well in number of matrix-vector products, and there is only a little margin for improvement for FM. We also see in Table 4.5 that the number of Hessian evaluations, when it is not negligible, is much worse for MR.

We see in Tables 4.3 and 4.4 that the number of gradient evaluations does never exceed the number of objective function evaluations for each problem and algorithm. This is due

Problem	Method	Problem size						
Surf		15^2	31^2	63^2	127^2	255^2	511^2	1023^2
	FM	15	17	16	19	27	30	33
	MR	138	498	879	1620	2891	5829	14885
Inv		15^2	31^2	63^2	127^2	255^2		
	FM	36	58	127	163	231		
	MR	2380	7252	6853	11797	13827		
Opt		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	14	26	37	40	42	39	
	MR	24	93	348	1207	4249	14826	
NC		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	2	2	2	3	8	6	
	MR	0	246	0	0	3	11	

Table 4.2: Non-quadratic problems: number of smoothing cycles or Hessian-vector products (at finest level) versus problem size.

Problem	Method	Problem size						
Surf		15^2	31^2	63^2	127^2	255^2	511^2	1023^2
	FM	21	26	24	30	150	161	167
	MR	5	16	8	15	40	185	921
Inv		15^2	31^2	63^2	127^2	255^2		
	FM	143	222	332	498	575		
	MR	91	223	355	805	1046		
Opt		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	18	41	61	70	72	62	
	MR	3	3	3	3	3	3	
NC		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	4	4	4	5	16	13	
	MR	1	2	1	1	2	2	

Table 4.3: Non-quadratic problems: number of objective function evaluations (at finest level) versus problem size.

to the fact that a gradient and an objective function value are evaluated each time a model for the trust-region algorithm is computed. Furthermore, some additional objective function evaluations are made during the linesearch procedure. Considering again Surf and Inv, on the one hand we see that, at the finest level, FM requires less evaluations than MR. On the other hand, on problems Opt and NC, MR needs less evaluations than FM.

More globally, FM is clearly superior to MR on problems Surf and Inv: FM requires less calls to all the kernels at the finest level and often at lower ones. For problem Opt, FM requires much less linear algebra than MR, but more function and gradient evaluations. Let us compare the cpu-time costs of an objective function and a gradient evaluation versus that

Problem	Method	Problem size						
Surf		15^2	31^2	63^2	127^2	255^2	511^2	1023^2
	FM	21	24	23	27	35	37	40
	MR	5	11	8	10	31	161	890
Inv		15^2	31^2	63^2	127^2	255^2		
	FM	59	83	158	201	271		
	MR	71	195	301	733	953		
Opt		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	18	35	52	58	60	56	
	MR	3	3	3	3	3	3	
NC		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	4	4	4	5	16	12	
	MR	1	2	1	1	2	2	

Table 4.4: Non-quadratic problems: number of gradient evaluations (at finest level) versus problem size.

Problem	Method	Problem size						
Surf		15^2	31^2	63^2	127^2	255^2	511^2	1023^2
	FM	3	5	5	10	6	5	7
	MR	4	10	7	10	31	164	891
Inv		15^2	31^2	63^2	127^2	255^2		
	FM	34	55	123	158	228		
	MR	72	198	306	742	962		
Opt		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	1	1	1	1	1	1	
	MR	2	2	2	1	1	1	
NC		15^2	31^2	63^2	127^2	255^2	511^2	
	FM	1	1	1	1	1	2	
	MR	0	1	0	0	1	1	

Table 4.5: Non-quadratic problems: number of Hessian evaluations (at finest level) versus problem size.

of a smoothing cycle on Opt. At the finest level, an objective function or gradient evaluation involves approximatively $14n_r$ and $56n_r$ floating-point operations, respectively. Due to the pentadiagonal structure of the Hessian, a smoothing cycle (or a matrix-vector product with the Hessian) involves $10n_r$ operations. These operations counts imply that MR is much more expensive than FM on this example, since the gain in the number of smoothing cycles is clearly much superior to the loss in objective function and gradient evaluations. However MR is superior to FM on problem NC, because the number of calls to all kernels is nearly always larger for FM than for MR.

4.3 Comparison of algorithmic variants

In our second set of numerical experiments, we compare variants of Algorithm RMTR in terms of number of calls to the relevant kernels. The comparisons are made with respect to the "default" Algorithm FM described in Section 4.2.1. We then consider eight additional variants, obtained from the default version by changing one algorithmic parameter as follows:

W2: two smoothing cycles are allowed per SCM smoothing iteration, instead of one;

W3: three smoothing cycles are allowed per SCM smoothing iteration, instead of one;

V1: V-form recursions are performed instead of W-form recursions (see Section 3.6);

F1: free form recursion is considered instead of W-form recursion (see Section 3.6);

LMOD: the Galerkin model (with $f_{i-1} = 0$) is replaced by (2.4), which enforces coherence of the linearized models between successive levels;

QMOD: the Galerkin model (with $f_{i-1} = 0$) is replaced by (3.13), which enforces coherence of the full quadratic models between successive levels;

NOLS: no linesearch is performed (see Section 3.2);

LINT: the cubic interpolation that is used in the initialization phase is replaced by a linear interpolation (see Section 3.7).

For each of the test problems, we display the number of calls to the different kernels in the four finest levels, where most of the computational work is performed.

4.3.1 Performance of the variants

Tables 4.6 to 4.8 display the number of smoothing cycles/matrix-vector products and the number of Hessian evaluations for the quadratic problems. The number of calls to all kernels are reported in Tables 4.9 to 4.12 for the non-quadratic ones.

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		63	127	255	511		63	127	255	511
Cycles/ mat.vec.	default	35	22	16	9	default	35	22	16	9
	W2	44	34	22	14	LMOD	131	66	30	12
	W3	63	42	24	12	QMOD	35	22	16	9
	V1	34	22	17	9	NOLS	34	21	14	8
	F1	35	22	16	9	LINT	37	20	15	9
Hessian evals.	default	1	1	1	1	default	1	1	1	1
	W2	1	1	1	1	LMOD	59	28	10	1
	W3	1	1	1	1	QMOD	13	7	4	1
	V1	1	1	1	1	NOLS	1	1	1	1
	F1	1	1	1	1	LINT	1	1	1	1

Table 4.6: Comparison of the algorithmic variants on Problem DN at level 511.

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		127^2	255^2	511^2	1023^2		127^2	255^2	511^2	1023^2
Cycles/ mat.vec.	default	18	11	6	4	default	18	11	6	4
	W2	18	12	6	2	LMOD	79	38	9	3
	W3	214	18	12	6	QMOD	25	11	6	3
	V1	17	10	6	4	NOLS	18	11	6	4
	F1	18	11	6	4	LINT	21	16	11	7
Hessian evals.	default	1	1	1	1	default	1	1	1	1
	W2	1	1	1	1	LMOD	15	12	4	1
	W3	1	1	1	1	QMOD	11	5	2	1
	V1	1	1	1	1	NOLS	1	1	1	1
	F1	1	1	1	1	LINT	1	1	1	1

Table 4.7: Comparison of the algorithmic variants on Problem Q2 at level 1023^2.

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		7^3	15^3	31^3	63^3		7^3	15^3	31^3	63^3
Cycles/ mat.vec.	default	31	23	15	9	default	31	23	15	9
	W2	32	34	24	12	LMOD	82	39	19	8
	W3	36	42	24	12	QMOD	31	23	15	9
	V1	29	23	15	9	NOLS	32	24	17	11
	F1	31	23	15	9	LINT	36	27	18	9
Hessian evals.	default	1	1	1	1	default	1	1	1	1
	W2	1	1	1	1	LMOD	23	10	4	1
	W3	1	1	1	1	QMOD	11	7	4	1
	V1	1	1	1	1	NOLS	1	1	1	1
	F1	1	1	1	1	LINT	1	1	1	1

Table 4.8: Comparison of the algorithmic variants on Problem Q3 at level 63^3.

We first compare the default, W2 and W3 variants, which differ only in the maximum number of smoothing cycles per SCM smoothing iteration (the default variant may be thought of as W1). In terms of linear algebra costs, the global conclusion support our default choice, which is best on all problems but Q2, albeit the difference in this case remains small. The picture is less clear if we consider the evaluations of the problem-dependent kernels: W3 dominates here except on problem Inv. The observed trade-off between linear algebra and evaluation of the nonlinear problem functions is therefore consistent with the intuition that investing more effort in one benefits the other.

We now turn to the comparison of the recursion forms and consider the default variant together with V1 and F1. All three variants compete closely in terms of linear algebra costs, with the exception of the bad performance of V1 on problem Opt. On the nonlinear kernels, we note the similar performance of F1 and the default, while V1 is less predictable. These results also indicate that our stopping rules in terms of gradient accuracy leave little room for longer recursion patterns.

If we now consider the choice between the default Galerkin model and the linearly/quadra-

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		127^2	255^2	511^2	1023^2		127^2	255^2	511^2	1023^2
Cycles/ mat.vec.	default	51	48	29	33	default	51	48	29	33
	W2	63	47	55	49	LMOD	340	307	260	150
	W3	87	61	48	33	QMOD	49	40	32	35
	V1	51	48	29	33	NOLS	52	43	47	117
	F1	51	48	29	33	LINT	43	38	40	28
Objective evals.	default	29	44	34	167	default	29	44	34	167
	W2	19	26	162	178	LMOD	617	560	458	335
	W3	21	27	24	35	QMOD	87	74	61	147
	V1	29	44	34	167	NOLS	29	29	45	131
	F1	29	44	34	167	LINT	24	29	157	158
Gradient evals.	default	26	33	23	40	default	26	33	23	40
	W2	16	16	32	43	LMOD	535	465	363	187
	W3	18	16	16	24	QMOD	82	66	51	42
	V1	26	33	23	40	NOLS	28	28	42	125
	F1	26	33	23	40	LINT	21	23	33	35
Hessian evals.	default	12	14	7	7	default	12	14	7	7
	W2	7	6	10	17	LMOD	96	73	66	37
	W3	10	8	9	17	QMOD	30	22	21	18
	V1	12	14	7	7	NOLS	12	9	6	84
	F1	12	14	7	7	LINT	4	6	4	5

Table 4.9: Comparison of the algorithmic variants on Problem Surf at level 1023^2.

tically coherent models (2.4) and (3.13), the conclusion from our tests is easy: the Galerkin model very often outperforms the others, and the linearly coherent model is typically the worst. As expected, all these choices are essentially equivalent in terms of linear algebra on quadratic problems, but differ in their use of the nonlinear kernels.

The effect of the linesearch is globally positive, as can be verified by comparing the NOLS and default variants. The default variant is nearly always better in terms of linear algebra, at the price of a moderate increase in objective function evaluations. The effect is most dramatic on problem Surf, where the NOLS variant is clearly less effective.

Finally, the default variant, which uses cubic interpolation when prolongating lower grid solutions in the initialization process, is in general comparable to the LINT variant, where linear interpolation is used for these prolongations. The nonlinear nature of the objective function seems to be the decisive factor: when strong nonlinearities occur (such as in Surf), the advantage of cubic interpolation is reduced in that the prolongated solutions may contain substantial high-frequency components and more smoothing is then necessary.

5 Conclusion and perspectives

We have presented an implementation of the recursive multilevel trust-region algorithm proposed by Gratton et al. (2004), as well as significant numerical experience on multilevel

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		31^2	63^2	127^2	255^2		31^2	63^2	127^2	255^2
Cycles/ mat.vec.	default	122	213	225	231	default	122	213	225	231
	W2	129	607	459	412	LMOD	58	136	252	224
	W3	279	207	241	462	QMOD	161	203	229	245
	V1	142	195	188	225	NOLS	131	184	262	268
	F1	122	213	225	231	LINT	146	247	252	219
Objective evals.	default	97	204	433	575	default	97	204	433	575
	W2	107	397	431	584	LMOD	181	321	669	626
	W3	173	225	358	679	QMOD	347	430	482	549
	V1	115	240	422	549	NOLS	50	107	213	341
	F1	97	204	433	575	LINT	108	267	424	596
Gradient evals.	default	39	105	172	271	default	39	105	172	271
	W2	48	298	271	328	LMOD	95	194	321	258
	W3	131	107	154	361	QMOD	282	335	326	279
	V1	57	130	173	258	NOLS	48	99	200	316
	F1	39	105	172	271	LINT	43	121	177	262
Hessian evals.	default	26	90	142	228	default	26	90	142	228
	W2	35	270	238	289	LMOD	42	110	207	171
	W3	115	87	126	324	QMOD	184	222	229	244
	V1	39	109	137	221	NOLS	31	78	155	244
	F1	26	90	142	228	LINT	31	98	139	215

Table 4.10: Comparison of the algorithmic variants on Problem Inv at level 255^2.

test problems. A suitable choice of the algorithm's parameters has been identified on these problems, yielding a good compromise between reliability and efficiency. The resulting default algorithm has then be compared to alternative optimization techniques, such as mesh refinement and direct solution of the fine-level problem. Finally, the sensibility of the default variant with respect to most important algorithmic parameters has been investigated, allowing possible further improvements depending on the problem's specific nature.

The authors are well aware that continued experimentation is needed on a larger spectrum of applications, but the numerical experience gained so far is very encouraging. Further algorithmic sophistications are of course possible, including, for example, a refined strategy for the choice between smoothing and truncated conjugate-gradient iterations and possible modifications in the norm used for defining the trust-region. Crucially, Hessian approximation schemes adapted to large-scale discretized problems must also be developed and integrated in the techniques described here.

The extension of the method beyond geometric multigrid applications is also currently considered in conjunction with algebraic multigrid techniques. A more ambitious development involving the inclusion of constraints into the problem formulation is the object of ongoing research.

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		63^2	127^2	255^2	511^2		63^2	127^2	255^2	511^2
Cycles/ mat.vec.	default	406	207	102	39	default	406	207	102	39
	W2	614	308	142	42	LMOD	1065	1614	2368	4208
	W3	747	375	177	54	QMOD	292	176	112	50
	V1	168	146	128	132	NOLS	396	204	101	42
	F1	498	255	105	35	LINT	384	218	109	43
Objective evals.	default	43	58	70	62	default	43	58	70	62
	W2	26	39	39	45	LMOD	1698	2263	2782	4353
	W3	17	21	27	35	QMOD	534	318	190	75
	V1	61	80	98	179	NOLS	35	44	51	60
	F1	43	55	65	55	LINT	52	61	68	69
Gradient evals.	default	34	46	55	56	default	34	46	55	56
	W2	23	30	33	39	LMOD	1667	2214	2733	4343
	W3	17	21	27	35	QMOD	518	302	174	68
	V1	40	56	74	168	NOLS	35	44	51	59
	F1	34	43	50	47	LINT	40	49	56	59
Hessian evals.	default	1	1	1	1	default	1	1	1	1
	W2	1	1	1	1	LMOD	385	309	237	10
	W3	1	1	1	1	QMOD	93	52	20	3
	V1	1	1	1	1	NOLS	1	1	1	1
	F1	1	1	1	1	LINT	1	1	1	1

Table 4.11: Comparison of the algorithmic variants on Problem Opt at level 511^2.

A Some useful transfer operator properties

In our implementation, we use the level-dependent norms $\|\cdot\|_r = \|\cdot\|$ and $\|s_{i-1}\|_{i-1} = \|P_i s_{i-1}\|_i$ $(i = 1, \ldots, r)$. Assuming that the 2-norm is chosen at the finest level, we then have that $\|s_{i-1}\|_{i-1} = \sqrt{\langle s_{i-1}, M_{i-1} s_{i-1}\rangle}$, where M_{i-1}, for $i = 1 \ldots, r$, is defined by

$$M_{i-1} = P_i^T P_{i+1}^T \ldots P_r^T P_r \ldots P_{i+1} P_i.$$

The matrices M_{i-1} and their inverse are useful for instance if a TCG iteration is performed to solve the local subproblem (Conn et al. 2000, p.205). We shall see in this section that for a 2D or 3D problem, the matrices P_i, M_{i-1}, M_{i-1}^{-1}, and the norm $\|P_i\|$ (needed to compute σ_i) are easily computed from their 1D counterparts provided that the 2D or 3D-interpolation operator is obtained by applying the 1D operator in each direction.

For 2D problems, let $X \in \mathbb{R}^{n_i \times n_i}$ be a matrix representing the quantity $x = vec(X) \in \mathbb{R}^{n_i^2}$ to be interpolated. Here the *vec* operator stacks all the matrix columns into one vector. Let $p_i \in \mathbb{R}^{n_{i+1} \times n_i}$ be the matrix representing the 1D interpolation. The matrix $p_i X \in \mathbb{R}^{n_{i+1} \times n_i}$ hence represents the vector obtained by interpolating x in the first direction. To interpolate in the other direction, we perform the operation

$$Y = (p_i (p_i X)^T)^T = p_i X p_i^T,$$

and get the prolongation $y = vec(Y)$ of x in the columns of the $n_{i+1} \times n_{i+1}$ matrix Y. But,

	Variant	Number of kernel calls				Variant	Number of kernel calls			
Levels		63^2	127^2	255^2	511^2		63^2	127^2	255^2	511^2
Cycles/ mat.vec.	default	16	17	16	6	default	16	17	16	6
	W2	52	42	24	15	LMOD	131	58	18	5
	W3	75	57	31	13	QMOD	80	28	20	12
	V1	47	47	28	7	NOLS	11	12	10	5
	F1	17	20	20	6	LINT	18	14	14	5
Objective evals.	default	6	5	14	13	default	6	5	14	13
	W2	5	8	11	19	LMOD	216	103	40	13
	W3	5	8	10	10	QMOD	132	55	43	29
	V1	5	14	34	14	NOLS	5	2	12	10
	F1	5	2	18	12	LINT	8	4	16	10
Gradient evals.	default	6	5	14	12	default	6	5	14	12
	W2	5	8	10	17	LMOD	215	102	35	10
	W3	5	8	10	9	QMOD	131	54	39	23
	V1	5	14	34	13	NOLS	5	2	12	8
	F1	5	2	18	11	LINT	8	4	16	8
Hessian evals.	default	1	1	1	2	default	1	1	1	2
	W2	1	1	1	1	LMOD	29	16	6	1
	W3	1	1	1	1	QMOD	17	10	12	5
	V1	1	1	2	1	NOLS	1	1	1	1
	F1	1	1	1	1	LINT	1	1	1	2

Table 4.12: Comparison of the algorithmic variants on Problem NC at level 63^2.

by definition of the Kronecker product, this can be written as

$$y = vec(Y) = vec(p_i X p_i^T) = (p_i \otimes p_i) vec(X) = (p_i \otimes p_i)x.$$

Therefore, the 2D interpolation operator P_i is simply the matrix $p_i \otimes p_i$. From this and the definition of M_{i-1}, it follows that

$$M_{i-1} = (p_i \otimes p_i)^T (p_{i+1} \otimes p_{i+1})^T \ldots (p_r \otimes p_r)^T (p_r \otimes p_r) \ldots (p_{i+1} \otimes p_{i+1})(p_i \otimes p_i).$$

Now, since $(p_i \otimes p_i)^T = p_i^T \otimes p_i^T$, since for compatible dimensions, $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, and since for invertible matrices, $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, we easily obtain that

$$M_{i-1} = m_{i-1} \otimes m_{i-1} \qquad \text{and} \qquad M_{i-1}^{-1} = m_{i-1}^{-1} \otimes m_{i-1}^{-1},$$

where $m_{i-1} = p_i^T p_{i+1}^T \ldots p_r^T p_r \ldots p_{i+1} p_i$. It is therefore possible to represent M_i and M_i^{-1} compactly, using the 1D quantities. Observe finally that $\|P_i\| = \|p_i \otimes p_i\| = \|p_i\|^2$ is also easily computable. These results generalize to 3D problems.

B Time performance of the main computational kernels

We present here the costs in seconds of a call to the objective function, gradient and Hessian evaluation, and the time needed to perform a smoothing cycle and a matrix-vector product

by the Hessian. These kernels are typically the most time-consuming parts in optimization algorithms. The elapsed times for these kernels are reported in Table 2.13 for the finest level of each problem. We observe that for all the considered problems, a smoothing cycle and a matrix-vector product by the Hessian, require the same time. This is not surprising, because both operations involve also the same amount of floating-point operations. We also notice that for the non-quadratic problems, a cost evaluation is cheaper than a gradient evaluation, which in turn is cheaper than a Hessian evaluation. Note also that, in our test cases, the linear algebra operations are cheaper than the cost evaluations. This is due to the fact that the former operations are built-in or compiled Matlab commands. For the quadratic problems, the Hessian is constant and stored in memory, this is why accessing it is cheaper than a gradient and a cost evaluation.

Problem	size	obj. funct.	gradient	Hessian	Hessian × vect	smooth. cycle
DN	511	$7.6\ 10^{-3}$	$3.3\ 10^{-2}$	$1.7\ 10^{-3}$	$3.3\ 10^{-3}$	$2.4\ 10^{-3}$
Q2	1023^2	$1.1\ 10^{+0}$	$1.3\ 10^{+0}$	$2.0\ 10^{-1}$	$1.1\ 10^{-1}$	$9.6\ 10^{-2}$
Q3	63^3	$8.5\ 10^{-1}$	$9.0\ 10^{-1}$	$1.7\ 10^{-1}$	$2.8\ 10^{-2}$	$2.8\ 10^{-2}$
Surf	1023^2	$3.5\ 10^{-1}$	$5.9\ 10^{-1}$	$7.4\ 10^{+0}$	$1.2\ 10^{-1}$	$1.2\ 10^{-1}$
Inv	255^2	$6.2\ 10^{-2}$	$1.7\ 10^{-1}$	$6.0\ 10^{-1}$	$6.4\ 10^{-2}$	$5.0\ 10^{-2}$
Opt	511^2	$4.5\ 10^{-1}$	$4.4\ 10^{-1}$	$2.6\ 10^{+0}$	$4.7\ 10^{-2}$	$4.1\ 10^{-2}$
NC	511^2	$5.1\ 10^{-1}$	$1.1\ 10^{+0}$	$4.1\ 10^{+0}$	$9.3\ 10^{-2}$	$7.9\ 10^{-2}$

Table 2.13: Elapsed time in seconds for objective function evaluation, gradient evaluation, Hessian evaluation, Hessian-vector product and smoothing cycle.

References

A. Borzi and K. Kunisch. A globalisation strategy for the multigrid solution of elliptic optimal control problems. *Optimization Methods and Software*, **21**(3), 445–459, 2006.

W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, USA, 2nd edn, 2000.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. Number 01 *in* 'MPS-SIAM Series on Optimization'. SIAM, Philadelphia, USA, 2000.

A. R. Conn, L. N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. *SIAM Journal on Optimization*, **9**(4), 924–947, 1999.

J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1983. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.

M. Fisher. Minimization algorithms for variational data assimilation. *in* 'Recent Developments in Numerical Methods for Atmospheric Modelling', pp. 364–385. ECMWF, 1998.

E. M. Gertz. *Combination Trust-Region Line-Search Methods for Unconstrained Optimization*. PhD thesis, Department of Mathematics, University of California, San Diego, California, USA, 1999.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, second edn, 1989.

S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. Technical Report 04/06, Department of Mathematics, University of Namur, Namur, Belgium, 2004.

S. Gratton, A. Sartenaer, and Ph. L. Toint. Second-order convergence properties of trust-region methods using incomplete curvature information, with an application to multigrid optimization. *Journal of Computational and Applied Mathematics*, (to appear), 2006.

P. C. Hansen. Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems. *Numerical Algorithms*, **6**, 1–35, 1994.

P. W. Hemker and G. M. Johnson. Multigrid approach to Euler equations. *in* S. F. McCormick, ed., 'Multigrid methods', Vol. 3 of *Frontiers in Applied Mathematics*, pp. 57–72, Philadelphia, USA, 1987. SIAM.

M. Lewis and S. G. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM Journal on Scientific Computing*, **26**(6), 1811–1837, 2005.

J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, **16**(1), 1–20, 1979.

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

J. Nocedal and Y. Yuan. Combining trust region and line search techniques. *in* Y. Yuan, ed., 'Advances in Nonlinear Programming', pp. 153–176, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.

J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, London, 1970.

M. J. D. Powell. A new algorithm for unconstrained optimization. *in* J. B. Rosen, O. L. Mangasarian and K. Ritter, eds, 'Nonlinear Programming', pp. 31–65, London, 1970. Academic Press.

T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, **20**(3), 626–637, 1983.

Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. *in* I. S. Duff, ed., 'Sparse Matrices and Their Uses', pp. 57–88, London, 1981. Academic Press.

Ph. L. Toint. VE08AD, a routine for partially separable optimization with bounded variables. *Harwell Subroutine Library*, **2**, 1983.

Ph. L. Toint. VE10AD, a routine for large scale nonlinear least squares. *Harwell Subroutine Library*, **2**, 1987.

C. R. Vogel. *Computational Methods for Inverse Problems*, Vol. 23 of *Computational Frontiers in Applied Mathematics*. SIAM, Philadelphia, USA, 2002.

I. Yavneh and G. Dardyk. A multilevel nonlinear method. *SIAM Journal on Scientific Computing*, **28**(1), 24–46, 2006.