A multilevel algorithm for solving the trust-region subproblem

by Ph. L. Toint, D. Tomanos
and M. Weber Mendonça

Report 07/04 1rst October 2007

[2] Department of Mathematics,
FUNDP-University of Namur,
61, rue de Bruxelles, B-5000 Namur, Belgium.
Email: philippe.toint@fundp.ac.be, dimitri.tomanos@fundp.ac.be
melissa.webermendonca@fundp.ac.be

A multilevel algorithm for solving the trust-region subproblem

Ph. L. Toint, D. Tomanos and M. Weber-Mendonça

1rst October 2007

Abstract

We present a multilevel numerical algorithm for the exact solution of the Euclidean trust-region subproblem. This particular subproblem typically arises when optimizing a nonlinear (possibly non-convex) objective function whose variables are discretized continuous functions, in which case the different levels of discretization provide a natural multilevel context. The trust-region problem is considered at the highest level (corresponding to the finest discretization), but information on the problem curvature at lower levels is exploited for improved efficiency. The algorithm is inspired by the method of Moré and Sorensen (1979), for which two different multilevel variants will be analyzed. Some preliminary numerical comparisons are also presented.

Keywords: Nonlinear optimization, trust-region subproblem, numerical algorithms, multilevel methods.

1 Introduction

Trust-region methods are a well-known class of optimization techniques, recognized to be both theoretically sound and numerically efficient (see Conn, Gould and Toint, 2000 for a comprehensive description). In a series of recent papers, Gratton, Sartenaer and Toint (2007*b*, 2006*b*, 2006*a*) and Gratton, Mouffe, Toint and Weber-Mendonça (2007*a*), have specialized this class of methods to the case where the optimization problem at hand involves variables that are discretized continuous functions(1). Such problems are typically very large as the discretization mesh goes to zero, and their solution at the finest discretization level is very inefficient: the recursive trust-region methods proposed by these authors then provide a numerically much more attractive alternative, whose behaviour is also backed by a strong theoretical convergence argument. The main idea in this recursive algorithm is to consider a low level description of the objective function as a model for high level optimization.

Our objective in the present contribution is to investigate an alternative application of the trust-region paradigm to the same class of problems. Instead of considering a hierarchy of objective functions, we consider here multilevel techniques for the (exact) solution of the trust-region subproblem at the highest level, that is for the finest discretization. If the objective function is (locally) convex, then a suitable optimizing step is derived from the solution of (a variant of) Newton's equations, which often results in solving a positive-definite linear system. This is for instance the case if the local Hessian is given by a discretized Laplacian or other elliptic operator. In this case, one can very naturally consider applying a classical multigrid linear solver to this system, yielding a very efficient method to compute the step. We refer the interested reader to Trottenberg, Oosterlee and Schüller (2001), or Briggs, Henson and McCormick (2000) for excellent

(1) Note that conceptually similar techniques have also been proposed in the linesearch setting by Fisher (1998) Nash (2000) Lewis and Nash (2005), Oh, Milstein, Bouman and Webb (2005) and Wen and Goldfarb (2007), but we will not investigate this avenue in this paper.

expositions of the principles and methods of this class of algorithms, originally introduced by Brandt (1977). However, things become much less clear when the objective function is locally non-convex, in which case a suitable step is no longer given by Newton's equations. Techniques for computing a step in this case are well-known for small dimensional problems (see Hebden, 1973, Moré and Sorensen, 1979, or Section 7.3 in Conn et al., 2000), and guarantee, in most cases, that every limit point of the sequence of iterates is a second-order stationary point. However, these techniques are unfortunately very often impractical for large discretized problems because they involve factorizing a Hessian matrix defined on the fine mesh. This is particularly limiting if one considers the discretization of variational problems in three dimensions or more. Our objective in the present paper is to propose two multilevel variants of this algorithm that are suitable for these large problems but nevertheless guarantee convergence to second-order limit points. These variants are again constructed using the multigrid principle.

The paper is organized as follows. Section 2 formally describes the problem and recalls the Moré-Sorensen method for its solution in small dimensions. Multigrid methods for linear systems are then very briefly reviewed in Section 3. The multigrid trust-region solvers are then introduced in Section 4 and their numerical efficiency compared in Section 5. Some conclusions and perspectives are finally discussed in Section 6.

2 Problem Formulation

We consider the solution of an unconstrained optimization problem of the form

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

where f is a twice continuously differentiable function and bounded below. Trust-region algorithms are iterative methods to compute this solution, which, given an initial guess x_0, construct a sequence of iterates x_k ($k\geq 0$) converging to the solution of problem (2.1). At each iteration k, a step s_k is obtained by minimizing a (typically quadratic) model m_k of f in the neighbourhood $\{x\in\mathbb{R}^n \mid \|x-x_k\|\leq\Delta_k\}$ defined for some radius Δ_k and some norm $\|\cdot\|$. The standard *trust-region subproblem* defining s_k is then to solve

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

where $g_k=\nabla_x f(x_k)$ and H_k is a bounded symmetric approximation of $\nabla_{xx}f(x_k)$. The computed step s_k is then accepted (in the sense that $x_{k+1}=x_k+s_k$) if a sufficient reduction in the objective function is obtained at x_k+s_k or rejected otherwise. While in general it is possible to choose any norm to define the neighbourhood in (2.2), we focus here on the Euclidean case because it has the remarkable property that the solution of (2.2) is computationally tractable even if H_k is indefinite (Vavasis and Zippel, 1990). Approximate solutions of this subproblem may also be computed by applying iterative methods such as the conjugate-gradients or generalized Lanczos-trust-region algorithms (see Section 7.5 in Conn et al. (2000)), but we focus here on the exact solution of (2.2).

It is possible to show (Gay, 1981, Sorensen, 1982, or Corollary 7.2.2 in Conn et al., 2000) that any global minimizer s^M of (2.2) satisfies the system of linear equations(2)

$$H(\lambda^M)s^M = -g, \tag{2.3}$$

where $H(\lambda^M)\stackrel{\text{def}}{=} H+\lambda^M I$ is positive semidefinite, $\lambda^M\geq 0$ and $\lambda^M(\|s^M\|-\Delta)=0$, with s^M being unique if $H(\lambda^M)$ is positive definite. This result indicates that s^M can be seen as the unconstrained minimizer of a quadratic model whose Hessian is made

(2)In what follows, since we will describe what happens within a single "outer" trust-region iteration, we will drop the iteration indices k for simplicity.

sufficiently positive definite by adding the term $\lambda^M I$. The Hessian curvature induced by this additional term must thus be strong enough to force s^M to lie within the *trust region*

$$\mathcal{B} \stackrel{\text{def}}{=} \{s \in \mathbb{R}^n \mid \|s\| \leq \Delta\}.$$

We also know that, if $\lambda > -\lambda_{\min}(H)$, the smallest eigenvalue of H, then $H(\lambda)$ is positive definite and the system (2.3) has a unique solution,

$$s(\lambda) = -H(\lambda)^{-1} g, \tag{2.4}$$

which must satisfy the nonlinear inequality

$$\|s(\lambda)\| \leq \Delta \tag{2.5}$$

whenever $\lambda = 0$ or the nonlinear equality

$$\|s(\lambda)\| = \Delta \tag{2.6}$$

if $\lambda > 0$. The Moré-Sorensen method consists in finding the minimizer $s(\lambda^M)$ by solving either (2.3) if (2.5) holds for $\lambda = 0$, or (2.3) and (2.6) together otherwise. An efficient way to perform the latter calculation is to use the (one dimensional) *secular equation*

$$\phi(\lambda) \stackrel{\text{def}}{=} \frac{1}{\|s(\lambda)\|} - \frac{1}{\Delta} = 0, \tag{2.7}$$

where $s(\lambda)$ is given by (2.4). This formulation has been shown to have better numerical properties than the more direct use of the constraint $\|s(\lambda)\| = \Delta$. The solution of the secular equation is then found by applying a root finding method, such as a safeguarded Newton's method. Assuming that $H(\lambda)$ is positive definite, the Newton step from λ happens to be (see Section 7.3.2 in Conn et al., 2000)

$$\lambda^{\text{new}} = \lambda + \left(\frac{\|s\| - \Delta}{\Delta}\right)\left(\frac{\|s\|^2}{\|w\|^2}\right) \quad \text{where } w \text{ solves } \; Lw = s \tag{2.8}$$

with L the lower Cholesky factor of $H(\lambda)$. If $H(\lambda)$ is not positive definite, then λ is increased until this is the case. The safeguard consists in keeping the λ iterates in an interval $[\lambda^L, \lambda^U]$, in order to guarantee convergence from arbitrary starting values. Thus, if λ^{new} is not inside the interval, this value is rejected and we choose a λ^{new} as, for instance, the geometric mean between λ^L and λ^U. The resulting algorithm is then given, for fixed $\Delta > 0$ and $\epsilon^\Delta > 0$, by Algorithm 2.1.

There are many sophistications to this algorithm, in particular regarding the choice of the initial λ, that of a new λ in the interval when λ^{new} falls outside and suitable termination rules. We refer the interested reader to Sections 7.3.4 to 7.3.11 of Conn et al. (2000) for details.

In large problems, performing the Cholesky factorization to solve the linear system can be very expensive. However, since we are dealing, for each λ, with a linear system of equations, it might be possible to accelerate the solution of this system if we can take advantage of the multilevel character of the problem. Our idea is thus to modify this method so that we use the multigrid approach for the solution of the linear system.

3 Multigrid Methods for Linear Systems

Multigrid methods to solve linear systems of equations are well established, and have the support of a solid convergence theory. We will not present a full explanation of these methods here, since this is out of the scope of the present paper, but will only briefly present the most important ideas for the construction of our method.

Algorithm 2.1: Outline of the Moré-Sorensen algorithm

$[s^*, \lambda^*] = MS(H, g, \Delta, \epsilon^\Delta)$

Step 1. If $H(0)$ is positive definite and $\|s(0)\| \leq \Delta(1+\epsilon^\Delta)$, terminate with $s = s(0)$.

Step 2. Determine an interval $[\lambda^L, \lambda^U]$ and an initial λ in this interval.

Step 3. Attempt a Cholesky factorization of $H(\lambda) = LL^T$. If this succeeds, solve $LL^T s = -g$. If $\Delta(1-\epsilon^\Delta) \leq \|s\| \leq \Delta(1+\epsilon^\Delta)$, i.e. if s is near the boundary of the trust region, terminate. If not s is not near the boundary, compute λ^{new} by (2.8).

Step 4. Update the interval $[\lambda^L, \lambda^U]$:

- if $\|s\| > \Delta(1+\epsilon^\Delta)$, or if the factorization has failed, redefine $\lambda^L = \lambda$;
- if $\|s\| < \Delta(1-\epsilon^\Delta)$, redefine $\lambda^U = \lambda$.

Step 5. Choose λ sufficiently inside $[\lambda^L, \lambda^U]$ and as close as possible to λ^{new}, if it has been computed. Go to Step 3.

Consider any iterative method for the solution of the system (2.3). At some iteration k of this method, we define the *residual*

$$r_k \stackrel{\text{def}}{=} -g - H(\lambda)s_k \tag{3.9}$$

and the *error*

$$e_k \stackrel{\text{def}}{=} s_{k+1} - s_k. \tag{3.10}$$

It is easy to see that the *residual equation*

$$H(\lambda)e_k = r_k \tag{3.11}$$

is equivalent to the initial system, in the sense that the solution e_k to this system gives the new iterate for (2.3) if we set $s_{k+1} = s_k + e_k$. Multigrid methods are based in the principle that some iterative methods for the solution of the linear system (3.11) (such as Jacobi or Gauss-Seidel) will quickly eliminate *oscillatory* components of the error (3.10), while leaving *smooth* components essentially unchanged. These methods are then referred to as *smoothing* methods.

The idea of multigrid methods is that we try to solve the residual equation not for $H(\lambda)$, but for some simpler approximation of this matrix in a lower dimensional space where smooth components of the error appear oscillatory. Assume that we have a collection of full rank operators $R_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}$ and $P_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ for $i = 1, \ldots, p$ (the *restriction* and the *prolongation*, respectively) such that $P_i = \sigma_i R_i^T$, with $\sigma_i > 0$, for all $i = 1, \ldots, p$. We will call each i a *level*, with $n_p = n$ such that $H_p(\lambda) = H(\lambda)$. In this case, we can construct a simpler representation of the matrix as the *Galerkin operator* for $H_i(\lambda)$ defined by

$$H_{i-1}(\lambda) = R_i H_i(\lambda) P_i. \tag{3.12}$$

This operator is not the only choice possible. However, it has many interesting properties, such as keeping the $i-1$ level operator symmetric and positive definite, if that is the case for the original $H_i(\lambda)$, and maintaining the structure created by the discretization.

Once this is done, we may redefine the residual equation in the lower level. Given $s_{i,k}$, the step in the current level, and call the right hand side of the equation we want to solve in this level by $b_{i,k}$. We then compute $r_{i,k}$, the residual, as in (3.9), by

$$r_{i,k} = b_{i,k} - H_i(\lambda)s_{i,k}.$$

The residual equation (3.11) at this level then takes the form

$$H_i(\lambda)e_{i,k} = r_{i,k}. \tag{3.13}$$

If we now restrict this equation to level $i-1$, the right-hand side at this level is now given by $R_i r_{i,k}$ and the residual equation at level $i-1$ becomes

$$H_{i-1}(\lambda)e_{i-1} = H_{i-1}(\lambda)R_i s_{i,k} - R_i r_{i,k} \overset{\text{def}}{=} r_{i-1,0}. \tag{3.14}$$

If the norm of this restricted residual is not large enough compared with the norm of the residual at level i, i.e. if $\|r_{i-1,0}\| < \kappa_{\rm r}\|r_{i,k}\|$ for some $\kappa_{\rm r} < 1$, then there is no advantage in trying to solve the lower level system. In this case, we perform smoothing iterations similar to those used in classical multigrid methods. Otherwise, if

$$\|r_{i-1,0}\| \geq \kappa_{\rm r}\|r_{i,k}\|, \tag{3.15}$$

we then compute a solution e_{i-1} of the lower level residual equation (3.14). The corresponding upper level step can now be recovered by $s_{i,k+1} = s_{i,k} + P_i e_{i-1}$. This procedure can be applied recursively, in that the solution of the residual equation in level $i-1$ itself can be computed recursively. At the coarsest level, which corresponds to the smallest system and where recursion is no longer possible, the solution may be computed exactly, for instance by using matrix factorization.

4 The Multi-level Moré-Sorensen Algorithm

We now wish to develop an algorithm for the solution of (2.1) that follows the general pattern of the Moré-Sorensen method but which, at the same time, exploits the ideas and techniques of multigrid. If the problem is convex and the multiplier λ^* is known, we propose to use a multigrid solver for the system (2.3), thereby exploiting the hierarchy of level-dependent problem formulations described in the previous section. If the multiplier is not known, we also face, as in the standard Moré-Sorensen method, the task to find its value, again exploiting the multi-level nature of the problem. Thus, in addition to the multigrid solution of (2.3), we must, as in Algorithm 2.1, find a new value of λ if the step computed as the solution of (2.3) does not satisfy our stopping conditions. Finding the value of λ^* may in practice be considered as a two-stages process. We first need to find a lower bound $\lambda^L \geq 0$ such that $H_p(\lambda)$ is positive-semidefinite for all $\lambda \geq \lambda^L$. Assuming that $\lambda^* = 0$ does not solve the problem (in the sense of (2.5)), the second is then to determine $\lambda^* \geq \lambda^L$ such that

$$\|s_p(\lambda^*)\|_2 = \|H_p(\lambda^*)^{-1}g\|_2 = \Delta, \tag{4.16}$$

where we have simply rewritten (2.4) and (2.6) at level p, the topmost in our hierarchy. In our multigrid context, we intend to exploit the restriction of that problem on the ith level where

$$\|s_i(\lambda^*)\|_i = \|H_i(\lambda^*)^{-1}g_i\|_i = \Delta, \tag{4.17}$$

where, as in Gratton et al. (2007*b*),

$$M_i \overset{\text{def}}{=} \prod_{\ell=i+1}^{p} R_\ell, \quad Q_i \overset{\text{def}}{=} \prod_{\ell=p}^{i+1} P_\ell, \quad g_i = M_i g \quad \text{and} \quad \|x\|_i \overset{\text{def}}{=} \|Q_i x\|_2.$$

The linear system implicit in (4.17) is then solved using the multigrid technique discussed in the previous section.

4.1 Exploiting the Level Structure to Find Bounds on λ^*

Consider ensuring positive-semidefiniteness of $H_p(\lambda)$ first. Our structure exploiting approach for this question is based on the simple observation that $H_i(\lambda)$ $(i = 2, \ldots, p)$ cannot be positive-semidefinite if $H_{i-1}(\lambda)$ is not, as expressed by the following property.

Lemma 4.1 *Let $P \in \mathbb{R}^{n_i \times n_{i-1}}$ be a full (column) rank matrix. If $\lambda_1^i \leq \ldots \leq \lambda_{n_i}^i$ are the eigenvalues of $A \in \mathbb{R}^{n_i \times n_i}$, and $\lambda_1^{i-1} \leq \ldots \leq \lambda_{n_{i-1}}^{i-1}$ are the eigenvalues of $RAP \in \mathbb{R}^{n_{i-1} \times n_{i-1}}$, where $R = \frac{1}{\sigma} P^T$ for some $\sigma > 0$, then we have that*

$$\lambda_1^{i-1} \geq \frac{\sigma_{\min}^2}{\sigma} \lambda_1^i, \tag{4.18}$$

where $\sigma_{\min}$ is the smallest singular value of P.

Proof. Using the extremal properties of eigenvalues (see Golub and Van Loan, 1983), we see that

$$\lambda_1^{i-1} = \min_{\substack{x \in \mathbb{R}^{n_{i-1}} \\ \|x\|_2 = 1}} \frac{\langle x, P^T A P x \rangle}{\sigma} = \min_{\substack{x \in \mathbb{R}^{n_{i-1}} \\ \|x\|_2 = 1}} \frac{\langle Px, APx \rangle}{\sigma} = \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\langle y, Ay \rangle}{\sigma}.$$

But, since $\|y\|_2 = \|Px\|_2 \geq \sigma_{\min}$, we obtain that

$$\lambda_1^{i-1} = \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \sigma_{\min}^2} \geq \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \|y\|_2^2} \geq \min_{y \in \mathbb{R}^{n_i}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \|y\|_2^2} = \frac{\sigma_{\min}^2}{\sigma} \lambda_1^i.$$

□

This property thus implies that the value of the multiplier needed to make $H_{i-1}(\lambda)$ convex provides a computable lower bound on that needed to make $H_i(\lambda)$ convex. In many cases of interest, the value of $\sigma_{\min}$ is known and larger that one. This is for instance the case when P is the linear interpolation operator in 1, 2 or 3 dimensions. However the exact value depends on the level considered and is typically costly to compute accurately, which leads us to consider the simpler case where we only assume that $\sigma_{\min} \geq 1$, in which case (4.18) can be rewritten, at level i as

$$\lambda_1^{i-1} \geq \frac{\lambda_1^i}{\sigma_i}.$$

Once this lower bound is computed, the algorithm then proceeds to increase λ^L (in a manner that we describe below) if evidence of indefiniteness of $H_p(\lambda)$ is found. We have considered two ways to obtain this evidence. The first is to attempt to solve the system $H_p(\lambda)s = -g$ for the step at level p by a multigrid technique, and to monitor the curvature terms $\langle d, H_i(\lambda)d \rangle$ occurring in the smoothing iterations at each level i. As soon as one of these terms is shown to be negative, we know from Lemma 4.1 that the lower bound λ^L must be increased. The second is to use a multilevel eigenvalue solver like the Rayleigh Quotient Minimization Multigrid (RQMG) Algorithm (see Mandel and McCormick, 1989) to compute λ_1^p, the smallest eigenvalue of H_p, associated with the eigenvector u_1^p. The RQMG algorithm solves the variational problem

$$RQ(u_1^p) = \min_{u \neq 0} RQ(u) = \min_{u \neq 0} \frac{\langle H_p u, u \rangle}{\langle u, u \rangle}$$

by applying a smoothing strategy adapted to the Rayleigh quotient minimization at each level i . The solution to this problem is an (upper) approximation to λ_1^p which, if negative, may therefore be used to deduce the bound $\lambda^L \geq -\lambda_1^p$. Observe that the RQMG algorithm (applied with sufficient accuracy) ensures that $H_p(\lambda^L)$ is, at least in inexact arithmetic, positive semidefinite.

In addition to the lower bound λ^L (which applies to all levels), we compute an initial upper bounds λ_i^U for each level i as in the Moré-Sorensen algorithm (observe that no information can be obtained from lower levels about λ_i^U). This therefore provides intervals $[\lambda^L, \lambda_i^U]$ for acceptable λ at each level i.

4.2 Updating λ in the Positive Definite Case

If $\lambda^L = 0$, $H_p(0)$ is positive-definite (in inexact arithmetic) and $\|s(0)\|_2 \leq \Delta$, our problem is solved. If this is not the case, our second task is then to adjust $\lambda \geq \lambda^L$ such that (4.16) holds. We now describe this adjustment procedure at level i, our final intention being to solve it at level p.

Since we are looking for λ that solves the secular equation (2.7), we can apply the Newton method to this end as we did in (2.8). However, in our case, the Cholesky factor L for $H(\lambda)$ is only available at the lowest level. Fortunately, note that

$$\|w\|^2 = \langle w, w\rangle = \langle L^{-1}s, L^{-1}s\rangle = \langle s, L^{-T}L^{-1}s\rangle = \langle s, (H(\lambda))^{-1}s\rangle.$$

Thus, if we compute y as the solution to the positive-definite system

$$H(\lambda)y = s(\lambda), \tag{4.19}$$

the Newton step for the secular equation at the current level then takes the form

$$\lambda^{\text{new}} = \lambda + \left(\frac{\|s\|_i - \Delta}{\Delta}\right)\left(\frac{\|s\|_i^2}{\langle s, y\rangle}\right). \tag{4.20}$$

Since we may not rely on factorizations for an exact solution of the system (4.19), we therefore apply a multigrid method to solve for w. However, this solution may be considered as costly. An alternative option is to update λ by applying a secant method to the secular equation, which gives

$$\lambda^{+} = \lambda - \phi(\lambda)\left(\frac{\lambda - \lambda_{\text{old}}}{\phi(\lambda) - \phi(\lambda_{\text{old}})}\right). \tag{4.21}$$

(We use $\lambda_{\text{old}} = \lambda^U$ to start the iteration.)

As in the Moré-Sorensen algorithm, if λ^{new} lies outside the interval, we choose λ inside the interval. One way to do this is to take λ^{new} as the half of the interval $[\lambda^L, \lambda^U]$, which corresponds to a simple bisection step. But we can expect better results by choosing to follow (Moré and Sorensen 1979) and setting

$$\lambda^{\text{new}} = \max\left[\sqrt{\lambda^L, \lambda^U}, \lambda^L + \theta(\lambda^U - \lambda^L)\right], \tag{4.22}$$

for $\theta \in (0, 1)$, which ensures that λ^{new} is closer to λ^L.

4.3 The Complete Algorithm

We need to introduce three further comments before the formal statement of the algorithm.

We first note that once a restricted trust-region problem (4.17) has been solved at level i, this means that the corresponding λ can be used as a lower bound for all higher levels. No further updating of λ is therefore necessary at this level and all lower ones, but we may nevertheless continue to exploit level i in the multigrid solution of the linear systems occurring at higher levels. The fact that a solution at level i has already been computed is remembered in our algorithm by setting the flag $\texttt{issolved}_i$. (For coherence, we define these flags for levels $1, \ldots, p+1$.)

Our second comment is that we still need to define stopping criteria for the multigrid solution of (4.17). A first criterion is obviously to terminate the iterations when the residual of the system is sufficiently small. In practice, we choose to stop the solution of the system as soon as

$$\|r_{i,k}\| = \|g_i - H_i(\lambda)s_{i,k}\| \le \epsilon^r,$$

with $\epsilon^r \in (0,1)$. However, we might need to introduce a second stopping rule. It may indeed happen that, for a current λ (too small), the step resulting from the system has a i-norm exceeding Δ. It is of course wasteful to iterate too long to discover, upon termination, that we have to throw the solution away. In order to avoid this wasteful calculation, we exploit the fact that the norm of the multigrid iterates is typically increasing as the iterations proceed. Thus, if this norm exceeds Δ by some threshold D^{++}, we decide to terminate the iterative process (and subsequently increase λ). However, we must be careful not to alter the lower and upper bounds on λ in this subsequent update, because of the possible inaccuracy generated by the early truncation of the system and the absence of any monotonicity guarantee (at variance with methods like truncated conjugate-gradients, see Steihaug, 1983). Unfortunately, it is also possible that no λ in the current interval produces a sufficiently small step. In this case, λ grows and becomes arbitrarily close to its upper bound. We avoid this situation by increasing our threshold whenever λ is within ϵ_i^λ of λ_i^U.

Finally, we have to propagate changes in λ between levels. Thus, if we have just updated λ and the old one was λ^-, we have that

$$H_i(\lambda) = H_i(\lambda^-) + (\lambda - \lambda^-)M_iQ_i. \tag{4.23}$$

Similarly, taking into account that each residual at level ℓ is computed with respect to the linear system at level $\ell+1$, one may verify that the residual update satisfies

$$r_{i,k+1} = r_{i,k} + \sum_{\ell=i}^{p}(-1)^{(\ell-i)}(\lambda-\lambda^-)M_\ell Q_\ell s_{\ell,k+1}, \tag{4.24}$$

where $s_{\ell,k+1}$ is the current iterate computed in level ℓ.

We now present the complete multigrid algorithm for the solution of the trust-region subproblem, the Multigrid Moré-Sorensen (MMS) Algorithm on 4.1 on the following page. Note that for each level i, we start by unsetting $\texttt{issolved}_i$.

Some comments on this algorithm are necessary at this point.

1. The algorithm is called form the virtual level $p+1$, after an initialization phase which computes, once and for all and for every level, the values of $D^+ = (1+\epsilon^\Delta)\Delta$, $D^- = (1-\epsilon^\Delta)\Delta$ and $D^{++} = \sigma_i D^+$ for some $\epsilon^\Delta \in (0,1)$. A level-dependent feasible interval $[\lambda^L, \lambda_i^U]$ is also computed at this stage. The (global) lower bound λ^L is set to the maximum between 0 and the opposite of the approximation of the most negative eigenvalue produced by the RQMG algorithm; the upper bound is calculated, for each level, exactly as for the Moré-Sorensen algorithm (see Conn et al., 2000, page 192), using the appropriate restrictions of the gradient and Hessian to the considered level. An initial value of $\lambda \in [\lambda^L, \lambda_i^U]$ is finally computed using (4.22) before the call to MMS proper.

2. We may essentially identify Steps 0 to 5 as a classical multigrid solver for a linear system when $\texttt{issolved}_i$ is set. The remaining contain the update to the λ parameter, broadly following the Moré-Sorensen method.

3. As explained in Section 3, the linear system (2.3) is solved by computing a correction at coarse levels to the steps already computed at finer ones. Our restriction strategy produces an algorithm analog to the application, in our nonlinear context, of the Full Multigrid Scheme (see Briggs et al., 2000, page 42).

4. We have not specified the details of the smoothing procedure in Step 4. In our expriments, we have used the Gauss-Seidel smoother, a classic in multigrid solvers (see Briggs et al., 2000, page 10).

Algorithm 4.1: $[s_{i,*},\lambda_i]$ = **MMS**(i, H_i, $r_{i,0}$, Δ, λ^L, λ, $s_{i,0}$, `issolved`$_i$)

Step 0. Initialization. Set $k = 0$.

Step 1. Iteration Choice. If $i = 1$, go to Step 3. Otherwise, if (3.15) fails, go to Step 4 (Smoothing iteration). Else, choose to go to Step 2 or to Step 4.

Step 2. Recursive Iteration. Call MMS recursively as follows:

$$[e_{i-1,*},\lambda_{i-1}] = \text{MMS}(i-1, H_{i-1}, r_{i-1,0}, \Delta, \lambda^L, \lambda, 0_{i-1}, \texttt{issolved}_{i-1})$$

where $r_{i-1,0}$ is computed as in (3.14). Compute $s_{i,k+1} = s_{i,k} + P_i e_{i-1,*}$. If `issolved`$_i$ is unset, i.e. this is the first time we perform a recursive iteration at this level, set $\lambda^L = \lambda_{i-1}$, choose $\lambda \in [\lambda^L, \lambda_i^U]$ using (4.22), update $H_i(\lambda)$ using (4.23) and $r_{i,k+1}$ using (4.24) and set `issolved`$_i$. Go to Step 5.

Step 3. Exact Iteration. If `issolved`$_{i+1}$ is unset, call the Moré-Sorensen algorithm (2.1), returning with solution $[s_{i,*},\lambda_i]$ = MS($H_i(\lambda)$,$r_{i,0}$,Δ,ϵ^{Δ}), and set `issolved`$_i$. Otherwise, just solve the system $H_i(\lambda)s_{i,*} = r_{i,0}$ exactly by Cholesky factorization of $H_i(\lambda)$ and return with solution $(s_{i,*}, \lambda)$.

Step 4. Smoothing Iteration. Apply μ smoothing cycles on the residual equation (3.13) yielding $s_{i,k+1}$, set $r_{i,k+1} = r_{i,k} + H_i(\lambda)(s_{i,k+1} - s_{i,k})$ and go to Step 5.

Step 5. Termination. If $\|r_{i,k+1}\| < \epsilon^r$ and `issolved`$_{i+1}$ is set, return $s_{i,k+1}$ and λ. Else, go to Step 1 if `issolved`$_{i+1}$ is set or if $\|r_{i,k+1}\| \geq \epsilon^r$ and $\|s_{i,k+1}\|_i \leq D^{++}$.

Step 6. Parameter update after full system solution.
If $\|r_{i,k+1}\| < \epsilon^r$ (and `issolved`$_{i+1}$ is unset),

Step 6.1: step threshold update. If $\lambda_i^U - \lambda < \epsilon_i^{\lambda}$, set $D^{++} = 2D^{++}$.

Step 6.2: interior solution test. If $\lambda = 0$ and $\|s_{i,k+1}\|_i < D^+$, or if $\lambda \geq 0$ and $D^- \leq \|s_{i,k+1}\|_i \leq D^+$, return with solution $s_{i,*} = s_{i,k+1}$ and $\lambda_i = \lambda$.

Step 6.3: parameter and interval updates. If $\|s_{i,k+1}\|_i > D^+$, set $\lambda^L = \lambda$. If $\|s_{i,k+1}\|_i < D^-$, set $\lambda_i^U = \lambda$. Compute a new $\lambda \in [\lambda^L, \lambda_i^U]$ using (4.20) or (4.21).

Step 6.4: reset the step. Set $s_{i,k+1} = 0$, $r_{i,k+1} = r_{i,0}$, update $H_i(\lambda)$ using (4.23), and go to Step 1.

Step 7: Parameter update after incomplete system solution.
If $\|r_{i,k+1}\| \geq \epsilon^r$ (and $\|s_{i,k+1}\|_i > D^{++}$),

Step 7.1: parameter update. compute a new $\lambda \in [\lambda, \lambda_i^U]$ using (4.20) or (4.21).

Step 7.2: reset the step. Set $s_{i,k+1} = 0$, $r_{i,k+1} = r_{i,0}$, update $H_i(\lambda)$ using (4.23), and go to Step 1.

5 Preliminary Numerical Experience

In this section we present some numerical results obtained by two variants of the MMS method applied in a trust-region algorithm (Algorithm BTR on page 116 of Conn et al., 2000) for four different test problems involving three-dimensional discretizations. Some of these problems were also tested in (Gratton et al. 2006*a*) in their two-dimensional formulation. All problems presented here are defined on the unit three-dimensional cube S_3 and tested with a fine discretization of 63^3 variables, and we used 4 levels of discretization. The Laplacian operator is obtained from the classical 7-points pencil. The prolongation operator is given by linear interpolation, and the restriction as its normalized (in the $\|\cdot\|_1$ norm) transpose, thereby defining $\sigma_i = \|P_i\|_1$. We briefly review these test problems below.

5.1 Optimization Test Problems

3D Quadratic Problem 1 (3D-1): A convex quadratic problem, where we consider the three-dimensional boundary value problem defined by

$$\begin{aligned} -\Delta u(x,y,z) &= f \quad \text{in } S_3 \\ u(x,y,z) &= 0 \quad \text{on } \partial S_3, \end{aligned}$$

where f is chosen so that the analytical solution to this problem is $u(x,y,z) = 8$. This gives linear systems $A_i x = b_i$ at level i where each A_i is a symmetric positive-definite matrix. This problem is the typical *model problem* for multigrid solvers. Here, we want to find the solution to its variational formulation

$$\min_{x \in \mathbb{R}^{n_p}} \frac{1}{2} x^T A_p x - x^T b_p.$$

3D Nonlinear Problem 2 (3D-2): Another convex quadratic problem, where we consider the differential equation

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

where f is chosen so that the analytical solution to this problem is

$$u(x,y,z) = x(1-x)y(1-y)z(1-z).$$

This problem is again considered in its variational formulation, as for problem 3D-1.

Convection-Diffusion problem (C-D): Here, minimize the variational formulation of the following nonlinear partial differential equation

$$\Delta u - Ru\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z}\right) + f(x,y,z) = 0, \qquad R = 20,$$

where $f(x,y,z) = 2000x(1-x)y(1-y)z(1-z)$, over S_3 with Dirichlet boundary conditions $u = 0$ on ∂S_3.

Boundary Value Problem (BV): This is a problem inspired by the one dimensional two-point boundary value problem presented in Moré, Garbow and Hillstrom (1981) and is defined by

$$-\Delta u(s,t,z) = \tfrac{1}{2}(u(s,t,z) + t + s + z + 1)^3,$$

with

$$\begin{aligned} u(0,t,z) = u(1,t,z) = 0, &\quad 0 < t < 1, \\ u(s,0,z) = u(s,1,z) = 0, &\quad 0 < s < 1, \\ u(s,t,0) = u(s,t,1) = 0, &\quad 0 < z < 1. \end{aligned}$$

Here, we look for the solution of the least squares problem

$$\min_{s,t,z\in[0,1]} \|-\Delta u(s,t,z) - \tfrac{1}{2}(u(s,t,z)+t+s+z+1)^3\|_2^2.$$

5.2 Numerical Results

We discuss here results obtained by applying the simple BTR trust-region method for the minimization of these problems, in which the subproblem is solved[(3)] at each iteration by one of three multigrid variants of the Moré-Sorensen algorithm. The first variant (MMS-secant) is the the MMS algorithm presented in this paper, where we use the secant approach (4.21) to solve the secular equation. The second (MMS-Newton) is the same method, but using Newton's method (4.20) instead of (4.21). The third (naive MMS-secant) is a simpler version of MMS-secant in which we do not use information on λ from lower levels. In this variant, we solve the Moré-Sorensen system (2.3) by multigrid instead of using Cholesky factorization of the Hessian, but we only change λ at the topmost level. This is equivalent to setting $\texttt{issolved}_i$ for all levels $i < p+1$. We update λ by using the secant method on the secular equation, as described above. All runs were performed in Matlab v.7.1.0.183 (R14) Service Pack 3 on a 3.2 GHz Intel single-core processor computer with 2 Gbytes of RAM, using the parameters

$$\mu = 5, \quad \epsilon^{\Delta} = 0.1, \quad \epsilon^{r} = 10^{-6}, \quad \theta = 10^{-4}, \text{ and } \epsilon_i^{\lambda} = 0.01|\lambda_i^U - \lambda^L|.$$

Our results are shown in Table 5.1. In this table, $\#\lambda$ stands for the weighted number of λ-updates, where each update is weighted proportionally to the dimension of the subspace in which the update is performed. Similarly, $\#R$ stands for the weighted number of restrictions performed by the algorithm. This last number indicates how many recursive iterations were used to find the solution of the linear system over the course of optimization. The CPU reported (in seconds) is the average trust-region subproblem solution time over all optimization iterations.

	Naive MMS-secant			MMS-secant			MMS-Newton		
	$\#\lambda$	CPU	$\#R$	$\#\lambda$	CPU	$\#R$	$\#\lambda$	CPU	$\#R$
3D-1	17.3	5.7 (5)	65.6	9.2	4.8 (5)	46.4	7.5	10.5 (5)	34.1
3D-2	17.2	6.1 (6)	85.8	11.2	5.3 (7)	72.8	11.1	14.8 (6)	59.4
C-D	17.3	6.9 (6)	73.3	8.6	5.4 (6)	53.6	7.9	11.9 (6)	42.0
BV	41.0	434.4 (18)	474.4	23.9	465.4 (20)	279.2	30.3	452.7 (18)	313.6

Table 5.1: Results for three variants of the MMS method.

These results clearly demonstrate that the MMS-secant version of our algorithm performs much better than the naive version in terms of the number of λ-updates required to solve all the trust-region subproblems in an optimization run. The conclusion in terms of CPU time remains favourable for MMS-secant, even if care must be exercized here given the inaccuracy of the Matlab timer. This suggests that information obtained at lower levels is, in fact, useful for the solution of the problem and should therefore be exploited. We also note that MMS-Newton does not offer a significant advantage over MMS-secant. Even if less λ-updates are needed to find the solution, these updates are computationally much more expensive than the simple secant ones since a linear system must be solved by multigrid for in each update, resulting in an overall slower algorithm. It is also important to note that the last problem is non-convex, and thus requires much more time to be solved. This is also due to the fact that, in this case, we have to compute an initial λ^L using an estimate of the smallest eigenvalue of the Hessian in each BTR iteration by means of the RQMG algorithm.

[(3)]We require the Euclidean norm gradient of the objective function to be at most 10^{-6} for termination.

6 Conclusions

We have developed a new method for the exact solution of the trust-region subproblem which is suitable for large scale systems where the Moré-Sorensen method cannot be applied, for instance because factorizations are too costly or impossible. This method exploits the multigrid structure in order to extract curvature information from the coarse levels to speed up the computation of the Lagrange parameter associated with the subproblem.

We have presented some admittedly limited numerical experience, which shows the potential for the new method, both because it demonstrates that sizeable three-dimensional applications can be considered and because it outperforms a too naive multigrid implementation of the basic Moré-Sorensen algorithm.

Acknowledgements

The work of Dimitri Tomanos is supported by a grant of the FRIA (Fund for Research in Agriculture and Industry). The work of Melissa Weber-Mendonça is made possible by a grant of CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) from Brazil.

References

A. Brandt. Multi-level adaptative solutions to boundary value problems. *Mathematics of Computation*, **31**(138), 333–390, 1977.

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.

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

D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, **2**, 186–197, 1981.

G. H. Golub and C. F. Van Loan. *Matrix computations*. North Oxford Academic, Oxford, UK, 1983.

S. Gratton, M. Mouffe, Ph. L. Toint, and M. Weber-Mendonça. A recursive trust-region method in infinity norm for bound-constrained nonlinear optimization. Technical Report 07/01, Department of Mathematics, University of Namur, Namur, Belgium, 2007*a*.

S. Gratton, A. Sartenaer, and Ph. L. Toint. Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. Technical Report 06/01, Department of Mathematics, University of Namur, Namur, Belgium, 2006*a*.

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*, **24**(6), 676–692, 2006*b*.

S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, **(to appear)**, 2007*b*.

M. D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report T.P. 515, AERE Harwell Laboratory, Harwell, Oxfordshire, England, 1973.

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. Mandel and S. McCormick. A multilevel variational method for $Au = \lambda Bu$ on composite grids. *Journal of Computational Physics*, **80**(2), 442–452, 1989.

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.

J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, **7**(1), 17–41, 1981.

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

S. Oh, A. Milstein, Ch. Bouman, and K. Webb. A general framework for nonlinear multigrid inversion. *IEEE Transactions on Image Processing*, **14**(1), 125–140, 2005.

D. C. Sorensen. Newton's method with a model trust-region modification. *SIAM Journal on Numerical Analysis*, **19**(2), 409–426, 1982.

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

U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Elsevier, Amsterdam, The Netherlands, 2001.

S. A. Vavasis and R. Zippel. Proving polynomial-time for sphere-constrained quadratic programming. Technical Report TR 90-1182, Department of Computer Science, Cornell University, Ithaca, New York, USA, 1990.

Z. Wen and D. Goldfarb. A linesearch multigrid methods for large-scale convex optimization. Technical report, Department of Industrial Engineering and Operations Research, Columbia University, New York, July 2007.