# Approximating Hessians in multilevel unconstrained optimization

by V. Malmedy[1,2] and Ph. L. Toint[2]

Report 08/19 28 November 2008

[1] Fonds de la Recherche Scientifique — FNRS
5, Rue d'Egmont, B-1000 Bruxelles, Belgium

[2] Department of Mathematics
FUNDP — University of Namur
61, Rue de Bruxelles, B-5000 Namur, Belgium
E-mail: vincent.malmedy@fundp.ac.be, philippe.toint@fundp.ac.be

# Approximating Hessians in multilevel unconstrained optimization

Vincent Malmedy Philippe L. Toint

28 November 2008

**Abstract**

We consider Hessian approximation schemes for large-scale multilevel unconstrained optimization problems, which typically present a sparsity and partial separability structure. This allows iterative quasi-Newton methods to solve them despite of their size. Structured finite-difference methods and updating schemes based on the secant equation are presented and compared numerically inside the multilevel trust-region algorithm proposed by Gratton, Mouffe, Toint and Weber-Mendonça (2008*b*).

**Keywords:** unconstrained optimization, multilevel problems, sparsity, partial separability, numerical experiences.

## 1 Introduction

Many optimization problems are formulated in infinite-dimensional spaces or represent such a problem in a large-scale finite-dimensional space. This is for instance the case in optimal control of dynamical systems, where problems are typically described by discretizations of either ordinary or partial differential equations, but also in the field of data assimilation in areas like meteorology, nuclear energy or hydrology (Fisher, 1998). A structure common to these problems is that they are posed not only in a single (typically infinite dimensional) space, but also in a hierarchy of nested spaces corresponding to different levels of discretization, from coarsest to finest. We consider here the following framework, where we are interested in solving an unconstrained optimization problem described, at the finest level, by

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

where the objective function $f$ is assumed to be twice continuously differentiable and bounded below. As is typical for functions arising in the context of discretization problems, $f$ is assumed to be *partially separable*, in the sense that there exists a decomposition of the form

$$f(x) = \sum_{i=1}^{m} f_i(x), \tag{1.2}$$

where each *element function* $f_i : \mathbb{R}^n \to \mathbb{R}$ $(i = 1, \ldots, m)$ depends only on a few variables, say those in some subset $\mathcal{I}_i \subset \{1, \ldots, n\}$ with $|\mathcal{I}_i| = n_i \ll n$. The concept of partial separability was introduced by Griewank and Toint (1982*a*) and has been successfully used in existing algorithms to significantly reduce the cost of approximating second derivatives (see Griewank and Toint, 1982*b*, Nocedal and Wright, 1999, and also Conn, Gould and Toint, 1992, for an example of practical implementation). The original concepts is slightly more general, but we will focus our attention on this case in what follows. As a consequence of (1.2), we immediately deduce that

$$\nabla_x f(x) = \sum_{i=1}^{m} \nabla_x f_i(x) \quad \text{and} \quad \nabla_{xx} f(x) = \sum_{i=1}^{m} \nabla_{xx} f_i(x), \tag{1.3}$$

where the vectors $\nabla_x f_i(x)$ and matrices $\nabla_{xx} f_i(x)$ have only a small number of potentially nonzero components: those corresponding to the variables indexed by $\mathcal{I}_i$.

Our objective is then to solve problem (1.1) by applying efficient variants of Newton method. These variants are characterized by their globalization strategy (linesearch or, in our case, trust region), by the nature of the approximations made to the derivatives (1.3) and also by the use of the hierarchy of discretization levels in a "multigrid-like" algorithm. Methods of this type have been proposed by Fisher (1998), Nash (2000), Oh, Milstein, Bouman and Webb (2003), and Wen and Goldfarb (2007) in the linesearch context, and by Gratton, Sartenaer and Toint (2008*c*) and Gratton *et al.* (2008*b*) for the case where a trust region is used to enforce global convergence. In all of these cases, these methods require the computation of the exact Hessian $\nabla_{xx} f(x_k)$ at an iterate $x_k$ or an approximation of this matrix. This part of the algorithms is costly both in processing time and (if structure is not preserved) in memory. The associated algorithmic choices are therefore crucial for the design of numerically efficient methods. The purpose of this paper is to review and compare strategies for computing such Hessian approximations while preserving the partially separable structure of the problem. This comparison is proposed in the context of the trust-region based multilevel variant by Gratton *et al.* (2008*b*).

Two main classes of numerical procedures are used to approximate second derivatives: finite-difference methods, which use gradient differences, and quasi-Newton methods, which are based on more specific secant equations. Finite-difference methods normally preserve the sparsity pattern of the Hessian. While the basic algorithm requires a number of gradient evaluations equal to the size of the problem (which makes this approach practically unaffordable), Powell and Toint (1979), inspired by Curtis, Powell and Reid (1974), propose to use the sparse structure present in the considered problems to drastically reduce this number of evaluations. We also refer to Coleman and Moré (1984) and Coleman, Garbow and Moré (1985). The use of these specialized finite-difference schemes is therefore attractive in our context and will therefore be included in our comparison.

In quasi-Newton methods, the Hessian is updated by some low-rank correction, rather than reevaluated, the update taking the curvature information collected between two successive iterates into account. More precisely, the current Hessian approximation $H$ is updated to $H^+$ to enforce the secant equation

$$H^+(x^+ - x) = \nabla_x f(x^+) - \nabla_x f(x), \tag{1.4}$$

where $x$ and $x^+$ are two successive iterates in the minimization process. However, standard updating procedures like the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, and Shanno, 1970), or the symmetric-rank-one (SR1) formula (see Davidon, 1959) usually destroy Hessian structure, which is not acceptable for the considered class of problems. This deficiency may be addressed in several ways.

The first is to construct a Hessian approximation $H^+$ subject to the secant equation (1.4) but preserving the sparsity pattern of $H$. This can be achieved by applying the sparse Powell-symmetric-Broyden (PSB) update derived by Marwil (1978) and Toint (1977), in which the new approximation is chosen to minimize the size of the correction $H^+ - H$ expressed in the Frobenius norm.

A second possibility is to consider the use of *partitioned updating* for partially separable problems, as proposed by Griewank and Toint (1982*a*). In this technique, the classical updating process is not applied on the Hessian of the objective function, but on that of each of its element functions. However this requires the knowledge of the element functions' gradients $\nabla_x f_i(x)$, which is not always realistic. Consider, for instance, a discretized problem arising from partial differential equations whose gradient is computed by solving an adjoint equation. In this case, the system solution provides the full gradient vector, but does not allow the distinction between gradients of the involved element functions, which makes the direct exploitation of the problem's partially separable structure impossible. One of the purposes of this paper is to show how the partially separable nature of the objective function can nevertheless be used, even if only the full gradient is available. Variational properties are used to recover the element gradients from the full gradient and the previously computed element Hessian matrices. Interestingly, this can be integrated in the construction of the complete Hessian approximation, the element gradients and Hessians remaining implicit in the calculation. The resulting algorithm, elaborated in Section 2.3, turns out to be very similar (up to the choice of the weighting norm) to the sparse PSB method. This connection is detailed in Appendix A.

However, neither the sparse PSB nor the new partially separable updating scheme maintain positive-definite Hessian approximations. This may be considered as a drawback, especially in the context of multilevel methods, whose efficiency is best on convex problems. We therefore complete our panel of Hessian approximation methods with a new attempt to obtain a positive-definite partitioned updating method (Section 2.4), whose performance is then included in our comparison.

The paper is organized as follows. We first review the contending Hessian approximation methods in Section 2. We then compare them on a battery of multilevel test problems in Section 3, and finally present some perspectives in Section 4.

## 2 Structure preserving Hessian approximation schemes

We first review the techniques we have considered for computing a Hessian approximation for our problem at the finest level.

### 2.1 Structured finite differences

Finite-differences can be used to approximate (second) derivatives of the objective function. Yet it would be inefficient to compute many zero values in a Hessian known to be sparse. This observation is at the basis of the CPR algorithm developed (for Jacobian matrices) by Curtis *et al.* (1974). In absence of sparsity, each column of the Jacobian is computed by taking a finite difference in the corresponding direction of the canonical basis of $\mathbb{R}^n$. Instead, their idea is to combine as many as possible of these directions into one, such that the differences can be untangled (the sparsity patterns of the assembled columns do not overlap). This procedure has been refined for (symmetric) Hessian matrices by Powell and Toint (1979). In the most economical variant (the substitution method), only the matrix lower or upper triangular part is computed. This method does not directly provide the values of the Hessian elements, but these are given as the solution of a triangular system of linear equations. The procedure computes the column grouping by applying the CPR algorithm on the lower triangular part of the Hessian.

**Algorithm 2.1: Lower triangular substitution (LTS) method**

**Step 1.** Apply the CPR algorithm to the lower triangular sparsity pattern of $H$.

**Step 2.** Estimate the corresponding gradient differences.

**Step 3.** Reconstruct the entries of the estimated $H$ by solving a triangular system of equations.

Even if counterexamples can be found, this method is often nearly optimal in terms of the necessary number of gradient evaluations per Hessian approximation. On the other hand, Goldfarb and Toint (1984) described optimal grouping based on computational molecules or stencils typically arising from the discretization of differential equations.

## 2.2 Sparse Powell-symmetric-Broyden updating

In the domain of quasi-Newton Hessian approximations, Marwil (1978) and Toint (1977) developed a sparse Hessian updating process for which a global and superlinear local convergence theory is established when combined with trust-region techniques (see Toint, 1979). In this method, the updated Hessian $H^+$ is required to satisfy a symmetry condition and a sparsity pattern defined by

$$H^+ = {H^+}^T \quad \text{and} \quad \mathcal{P}(H^+) = H^+, \tag{2.5}$$

where $\mathcal{P}$ is the *gangster* operator that zeroes all elements of a matrix that are prescribed to be zero by its sparsity structure; we also require the *secant equation*

$$H^+ s = y, \tag{2.6}$$

to be verified, where $s \overset{\text{def}}{=} x^+ - x$ is the iteration step and $y \overset{\text{def}}{=} g^+ - g$ is the corresponding variation in the gradient of the objective function (we use the notation $g = \nabla_x f(x)$ and $g^+ = \nabla_x f(x^+)$). As these conditions do not fully determine the updating process, a standard technique (see Dennis and Schnabel, 1983) consists in choosing the smallest correction matrix (in an appropriate norm) such that the updated Hessian satisfies (2.5) and (2.6). We consider here the weighted Frobenius norm defined by

$$\|A\|_\Omega^2 \overset{\text{def}}{=} \sum_{j,l=1}^{n} \Omega_{jl} A_{jl}^2,$$

for some $n \times n$ weight matrix $\Omega$ with positive elements. These considerations lead to Algorithm 2.2, in which $\bullet$ denotes the entrywise Hadamard product and $A^{\maltese}$ is the inverse of matrix $A$ for that product, *i.e.* $[A^{\maltese}]_{ij} = A_{ij}^{-1}$.

**Algorithm 2.2: Sparse PSB Update**

**Step 1.** Define $S = \mathcal{P}(ss^T) \bullet \Omega^{\maltese} + \text{diag}(\mathcal{P}(\Omega^{\maltese})(s \bullet s))$.

**Step 2.** Solve $S\lambda = y - Hs$ for $\lambda$.

**Step 3.** Compute $H^+ = H + \mathcal{P}(s\lambda^T + \lambda s^T) \bullet \Omega^{\maltese}$.

## 2.3 Partially separable PSB update

Assume now that the objective function is partially separable. Now denote by $I_i$ the $n \times n$ identity matrix with zeros at diagonal elements indexed by $\mathcal{I}_i^C = \{1, \ldots, n\} \setminus \mathcal{I}_i$, denote by $J$ the $n \times n$ matrix whose elements are all equal to 1 and define $J_i$ to be the $n \times n$ matrix with zero entries everywhere except in positions $\mathcal{I}_i \times \mathcal{I}_i$ where they are equal to 1. Observe that $J$ is the neutral element for the Hadamard product.

The partitioned updating technique by Griewank and Toint (1982*a*) then updates every element Hessian $H_i$ separately, using the element secant equation

$$H_i^+ s_i = y_i \stackrel{\rm def}{=} g_i^+ - g_i \tag{2.7}$$

where $H_i^+$ is the updated approximation of $\nabla_{xx} f_i(x^+)$, $s_i \in \mathbb{R}^n$ is the step vector $s$ whose components in $\mathcal{I}_i^C$ have been zeroed, $g_i = \nabla_x f_i(x)$ and $g_i^+ = \nabla_x f_i(x^+)$. However, as we indicated in the introduction, assuming the knowledge of the element gradients $g_i$ and $g_i^+$ and their difference $y_i$ may be unrealistic, and we are therefore interested in a technique that would avoid this requirement.

As in the previous section, we use variational properties to elaborate our new update and each matrix $H_i^+$ is chosen as close as possible to the current approximation $H_i$, under suitable conditions. More precisely, we propose to compute a correction to $H = \sum_{i=1}^m H_i$ and a set of element gradient differences $y_i$ which together minimize some (possibly) weighted sum (with weights $\omega_i > 0$) of the squared distances between $H_i^+$ and $H_i$ (in the Frobenius norm)

$$\tfrac{1}{2}\sum_{i=1}^m \omega_i \left\|H_i^+ - H_i\right\|_F^2, \tag{2.8}$$

under the structure constraints on the element gradients and Hessians

$$I_i y_i = y_i \quad \text{and} \quad J_i \bullet H_i^+ = H_i^+, \quad (i = 1, \dots, m), \tag{2.9}$$

the symmetry of the Hessian corrections $E_i \stackrel{\rm def}{=} H_i^+ - H_i$, that is

$$E_i = E_i^T, \quad (i = 1, \dots, m) \tag{2.10}$$

and the constraint that the (unknown) element gradients differences $y_i$ sum up to the full gradient difference $y$, that is

$$\sum_{i=1}^m y_i = y. \tag{2.11}$$

Obviously, we also require that the element secant equation (2.7) holds for all $i = 1, \dots, m$. We stress that no assumption is here made on the positive semi-definiteness of the element Hessians $H_i$.

The solution of this variational problem (2.7)–(2.11) is obtained by a Lagrangian technique. The Lagrangian function of this problem, which depends on multiplicators $\nu_i \in \mathbb{R}^n$ and $\chi_i \in \mathbb{R}^{n\times n}$ $(i = 1, \dots m)$ for constraints (2.9), $\mu \in \mathbb{R}^n$ for the constraint (2.11) and $2\lambda_i \in \mathbb{R}^n$ $(i = 1, \dots m)$ for constraints (2.7), can be written as

$$\begin{aligned} & L(E_1, \dots, E_m, y_1, \dots, y_m, \nu_1, \dots, \nu_m, \chi_1, \dots, \chi_m, \mu, \lambda_1, \dots, \lambda_m) \\ & = \sum_{i=1}^m \Bigg[ \tfrac{1}{2}\omega_i \left\|E_i\right\|_F^2 - \langle \nu_i, (I_i - I) y_i\rangle - \langle \chi_i, (J_i - J) \bullet (H_i + E_i)\rangle - \langle \mu, y_i\rangle \\ & \qquad - \left\langle 2\lambda_i, \left(H_i + \tfrac{E_i + E_i^T}{2}\right) s_i - y_i \right\rangle \Bigg] \end{aligned} \tag{2.12}$$

where $\langle A, B\rangle = \mathrm{tr}(A^T B)$ is the usual inner product for matrices $A, B \in \mathbb{R}^{n\times n}$. Note that there is no multiplicators for the symmetry constraints (2.10) on the correction $E_i$, because the contribution of secant equations to the Lagrangian is expressed in such a way that these constraints are automatically fulfilled.

At optimal solutions, the derivative of the Lagrangian with respect to each variable $y_i$ must be zero, that is

$$-(I_i - I)\nu_i - \mu + 2\lambda_i = 0, \quad (i = 1, \dots, m). \tag{2.13}$$

Focusing now on the components indexed by $\mathcal{I}_i$, we observe that the first term of (2.13) disappears, yielding to $2\tilde{\lambda}_i = \tilde{\mu}_i$, where $\tilde{\mu}_i, \tilde{\lambda}_i \in \mathbb{R}^{n_i}$ is the part of $\mu$ and $\lambda_i$, respectively, corresponding to the components indexed by $\mathcal{I}_i$. Clearly, these equations imply that every $\tilde{\lambda}_i$ is in fact a (maybe overlapping) piece of vector $\lambda \stackrel{\text{def}}{=} \mu/2$.

The derivatives of the Lagrangian with respect to variables $E_i$ also have to be zero, which means that the directional derivative of $L$ with respect to $E_i$ is zero in every (matrix) direction $K_i \in \mathbb{R}^{n\times n}$:

$$\begin{aligned} \mathrm{D}_{E_i}L \cdot K_i &= \tfrac{1}{2}\omega_i \mathrm{D}_{E_i}\mathrm{tr}(E_i^T E_i)\cdot K_i - \langle \chi_i, (J_i - J)\bullet K_i\rangle - \lambda_i(K + K^T)s_i \\ &= \omega_i \mathrm{tr} K_i^T E_i - \langle \chi_i, (J_i - J)\bullet K_i\rangle - \lambda_i(K_i + K_i^T)s_i = 0, \end{aligned} \tag{2.14}$$

where $\mathrm{D}_{E_i}L \cdot K_i$ is the derivative of $L$ with respect to $E_i$ in direction $K_i$. In particular, we may choose $K_i$ as any matrix with the same structure as $H_i$, that for which the only potentially nonzero entries are in the submatrix indexed by $\mathcal{I}_i \times \mathcal{I}_i$. We denote this $n_i \times n_i$ submatrix by $\tilde{K}_i$. For this choice of $K$, the second term of equation (2.14) disappears when considering only entries indexed by $\mathcal{I}_i \times \mathcal{I}_i$, and we obtain that

$$\omega_i \mathrm{tr}(\tilde{K}_i^T \tilde{E}_i) = \tilde{\lambda}_i(\tilde{K}_i + \tilde{K}_i^T)\tilde{s}_i,$$

where $\tilde{E}_i \in \mathbb{R}^{n_i\times n_i}$ is formed with the elements of $E_i$ indexed by $\mathcal{I}_i \times \mathcal{I}_i$. Developing this expression, we find that

$$\sum_{k,l=1}^{n_i} [\tilde{K}_i]_{jl}\left([\tilde{E}_i]_{jl} - \omega_i^{-1}\left([\tilde{\lambda}_i]_j[\tilde{s}_i]_l + [\tilde{\lambda}_i]_l[\tilde{s}_i]_j\right)\right) = 0, \quad \forall \tilde{K}_i \in \mathbb{R}^{n_i\times n_i},$$

which implies that

$$\tilde{E}_i = \omega_i^{-1}\left(\tilde{\lambda}_i\tilde{s}_i^T + \tilde{s}_i\tilde{\lambda}_i^T\right). \tag{2.15}$$

On the other hand, the element Hessian matrices $H_i$ have to fulfil the secant equation (2.7), *i.e.* $y_i = H_i s_i + E_i s_i$, yielding to $p_i \stackrel{\text{def}}{=} y_i - H_i s_i = E_i s_i$. Focusing again on the components indexed by $\mathcal{I}_i$, and using (2.15), we have that

$$\tilde{p}_i = \tilde{E}_i\tilde{s}_i = \omega_i^{-1}\left(\tilde{\lambda}_i\tilde{s}_i^T\tilde{s}_i + \tilde{s}_i\tilde{\lambda}_i^T\tilde{s}_i\right) = \omega_i^{-1}\big(\|\tilde{s}_i\|^2\,\tilde{I}_i + \tilde{s}_i\tilde{s}_i^T\big)\tilde{\lambda}_i \stackrel{\text{def}}{=} \omega_i^{-1}\tilde{S}_i\tilde{\lambda}_i,$$

where $\tilde{p}_i \in \mathbb{R}^{n_i}$ is the subvector of $p_i$ indexed by $\mathcal{I}_i$. From the small matrix $\tilde{S}_i \in \mathbb{R}^{n_i\times n_i}$, we construct the matrix $S_i \in \mathbb{R}^{n\times n}$ by adding zero rows and columns in such a way that $S_i$ has the same structure as $H_i$. Consequently, $p_i = \omega_i^{-1}S_i\lambda$ and, using the decomposition of the gradient difference $y$ into element gradient differences $y_i$, we obtain that

$$p \stackrel{\text{def}}{=} \sum_{i=1}^m p_i = y - Hs = \left(\sum_{i=1}^m \omega_i^{-1}S_i\right)\lambda \stackrel{\text{def}}{=} S\lambda.$$

Therefore, we just need to solve the system $p = S\lambda$, then substitute its solution $\lambda$ in (2.15) to find the corrections $E_i$ and finally, determine the updated element Hessians $H_i^+ = H_i + E_i$ and the full Hessian $H^+ = H + \sum_{i=1}^m E_i$.

Note that this updating procedure requires explicitly neither the element gradients $g_i^+$ or $g_i$, nor their differences $y_i$, despite that the latter appear as variables in our variational problem. There is thus no reason to compute them at each iteration, leading to Algorithm 2.3.

```
Algorithm 2.3: Partially Separable PSB Update

Step 1. Decompose vector $s$ into $\{s_i\}_{i=1}^m$ and define $S = \sum_{i=1}^m \omega_i^{-1}(\|s_i\|^2 I_i + s_i s_i^T)$.
Step 2. Solve system $S\lambda = y - Hs$ for $\lambda$ and decompose it into $\{\lambda_i\}_{i=1}^m$.
Step 3. Update Hessian: $H^+ = H + \sum_{i=1}^m \omega_i^{-1}(\lambda_i s_i^T + s_i \lambda_i^T)$.
```

This algorithm is very similar to Algorithm 2.2, but differ by the choice of weighted norm, the partially-separable version giving more weight to the parts of the Hessian belonging to overlapping elements. It may nevertheless be interpreted as a "PSB-type" update where one minimizes a (weighted) Frobenius norm of the matrix updates subject to linear conditions. The links between Algorithms 2.2 and 2.3 are explicited in Appendix A.

## 2.4 Positive-definite Hessian update

In the previous subsections, positive definiteness of the updated Hessian was not imposed. But as we aim to design Hessian updating procedure in the framework of multilevel methods, this property could be advantageous since multigrid methods are known to perform well on convex problems. However Sorensen (1981) exhibited a counterexample which shows that it may be impossible to require at the same time the preservation of a sparsity pattern for the updated Hessian, its positive definiteness and the secant equation (2.6) (Toint (1981) gave sufficient conditions for the existence of such updates).

In this paragraph we present an attempt to design such a positive definite Hessian updating process in the specific context of partially separable unconstrained optimization, but without assuming the knowledge of the individual element gradients. We start from the observation that, if these individual gradients were available, then we could apply the partitioned updating of Griewank and Toint (1982*a*) and perform the BFGS update on each block of the Hessian matrix. Hence, the idea is to re-create a collection of "element gradients" whose sum equals the full gradient. These vectors are then used in the partially separable BFGS update. However, as each elemental secant equation automatically holds for each elemental BFGS correction, the global secant equation then also holds, which we know is impossible in general. Our proposal is then to relax the summation condition (2.11) whenever necessary, in which case our element gradients may differ from their analytical values and the global secant equation no longer holds exactly.

In addition, the BFGS formula

$$H_i^+ = H_i - \frac{H_i s_i s_i^T H_i}{s_i^T H_i s_i} + \frac{y_i y_i^T}{s_i^T y_i}, \tag{2.16}$$

provides a positive definite update (if and) only if the inner product $\langle s_i, y_i \rangle$ is positive (and if $H_i$ is already positive definite). Splitting the gradient, and thus $y$, blindly could consequently be inappropriate. We thus aim to split the gradient variation $y$ into elemental gradient variations $y_i$ such that their sum is (not too different of) $y$ and that some elemental "curvature" is maintained in the sense that

$$\mu_i = \mu_i(y_i) \overset{\text{def}}{=} \frac{\langle s_i, y_i \rangle}{\langle s_i, H_i s_i \rangle} \geq \frac{\epsilon}{m} \sum_{j=1}^m \frac{\langle s_j, y_j \rangle}{\langle s_j, H_j s_j \rangle} \tag{2.17}$$

for some fixed $\epsilon > 0$. Note that we assume that each $H_i$ have been kept positive definite such that $\langle s_i, H_i s_i \rangle > 0$. Remark also that due to the structure of $y_i$ and $H_i$, the inner products $\langle s_i, y_i \rangle$ and $\langle s_i, H_i s_i \rangle$ can equivalently be written as $\langle s, y_i \rangle$ and $\langle s, H_i s \rangle$.

**Uniform splitting.** The simplest splitting procedure is to split the vector $y$ uniformly, in the following sense. Consider the $\ell$-th component of $y$ and consider the collection $\{y_i\}_{i\in\mathcal{K}(\ell)}$ where $\mathcal{K}(\ell) = \{i \mid \ell \in \mathcal{I}_i\}$ (this collection gathers the particular $y_i$ whose $\ell$-th component is possibly nonzero). Then the $\ell$-th component of each $y_i$ in this collection is defined as the $\ell$-th component of $y$ divided by $|\mathcal{K}(\ell)|$. Although this procedure may be judged simplistic, we still add to our algorithms test in the following form.

**Algorithm 2.4: Uniformly Partitioned BFGS Update**

**Step 1.** Decompose vector $s$ into $\{s_i\}_{i=1}^m$ and split uniformly $y$ into $\{y_i\}_{i=1}^m$.

**Step 2.** Perform the BFGS update (2.16) on each $H_i$ for which $\langle s_i, y_i\rangle \geq 10^{-6}\,\langle s_i, H_i si\rangle$.

**A curvature flow problem.** Unfortunately, the vectors $y_i$ resulting from the uniform splitting do not necessarily satisfy the curvature constraints (2.17). We then attempt to modify this initial split of $y$ by considering a *feasible flow problem.* We define a network whose nodes represent the element of the partially-separable structure, and such that each pair of nodes corresponding to elements sharing at least one variable is connected by an arc. Each of these arcs is directed to start at the node of the pair with largest value of the curvature measure $\mu_i$ and terminate at the node of the pair with the smallest $\mu_i$. Each node $i$ for which (2.17) is then considered as a source of (curvature) flow in the network and, symmetrically, each node for which (2.17) fails is considered as a sink.

Our aim is then to modify the values of the $y_i$, which then results in corresponding modifications of the $\mu_i$, such that (2.17) holds for all nodes. This is achieved by shifting curvature across the network from the sources to the sinks. The shift of curvature along the arc $i \to j$ is obtained by considering only the variables shared by elements $i$ and $j$, that is those in $\mathcal{I}_i \cap \mathcal{I}_j$, which is nonempty by construction of the network. In what follows, we use the superscript "$\cap$" to restrict vectors to the components indexed by this set $\mathcal{I}_i \cap \mathcal{I}_j$ ($i$ and $j$ remain implicit), and the superscript "$\setminus$" to restrict them to the complementary set of variables, which is $\mathcal{I}_i^C \cup \mathcal{I}_j^C$. To improve $\mu_j$, we add to $y_j^\cap$ some subvector $u$ such that $\langle s^\cap, u\rangle > 0$, which implies that

$$\langle s, y_j^+\rangle = \langle s^\setminus, y_j^\setminus\rangle + \langle s_j^\cap, y_j^\cap + u\rangle > \langle s^\setminus, y_j^\setminus\rangle + \langle s_j^\cap, y_j^\cap\rangle = \langle s, y_j\rangle\,,$$

such that $\mu_j^+$ is larger than $\mu_j$ after this modification. The same subvector $u$ is then symmetrically subtracted from $y_i^\cap$, with the effect that curvature is shifted from node $i$ to node $j$. The amount of shifted curvature is limited by the amount of positive curvature available in node $i$. We also impose another bound whose purpose is to avoid a too large increase in the norm of the vectors $y_i$ and $y_j$ (which in turn results in an increase in the norm and condition number of the Hessian approximation), and require that the norms of $y_i^\cap$ and $y_j^\cap$ remain bounded above by some constant $M$. This constraint defines the capacity of the arc $i \to j$, that is the amount of "curvature" that can be shifted along this arc.

Note that each arc capacity depends on the values of the elemental gradient variations corresponding to the extremities of the arc. The arc capacities are thus likely to vary along the shifting process. Our problem therefore departs from the classical flow problem in graph theory (where the capacities are fixed independently of demand). Moreover, the total flow $\sum \mu_i$ is not necessarily conserved after a modification of the flow (due to the definition (2.17)). As a consequence, an iterative process will be necessary, which involves updating the capacities and demand after one round of flow modifications, which in turn results in an implicit update of the total flow. The flow analogy is then used as a heuristic.

**Solving the feasible flow problem.** We next investigate methods designed to compute a maximal flow from a (possibly multiple) source to a (possibly multiple) sink on a network with fixed capacities (see Ahuja, Magnanti and Orlin, 1993, or Ahuja, Kodialam, Mishra and Orlin, 1997, for a survey). We decided to use the *push-relabel* algorithm introduced by Goldberg and Tarjan (1988), and more practically described by Cherkassky and Goldberg (1997). This method attributes to each node a (variable) distance label, which is, at the beginning, the distance to the sink node (that is the least number of nodes on a path from that node to the sink); the source has by assumption the maximal label. For each excess node (also called *active node*), the algorithm tries to reduce this excess, by sending it to adjacent nodes (this constitutes the *push* operations), given that transfers may occur only between nodes with consecutive distance label (from the highest to the lowest). But it may also happen that no more adjacent node has the required label while the excess is still not deleted. In this case, the label of the current node is updated (this is called a *relabel* operation) so that an adequate node appears, allowing again a push to reduce the excess. This procedure is repeated until every active node becomes inactive. There are several strategies to define in which order the algorithm deals with the active nodes. Here, we consider the *lowest-label* variant of this method, because it starts by considering the deficient nodes (that have the lowest labels), which are, in our case, the only problematic ones. Moreover, Ahuja *et al.* (1997) and Cherkassky and Goldberg (1997) seem to indicate that it could also give better results in practice.

**Performing a push and updating arcs capacities.** At each push step, $\delta$ units of flow (in our case, curvature) have to be transferred along the arc $i \to j$, with the aim of satisfying demand at node $j$ subject to capacity constraint on this arc. Consider the resulting modification of vectors $y_i^\cap$ and $y_j^\cap$ with respect to their mean $w = \frac{1}{2}(y_i^\cap + y_j^\cap)$. To preserve the summation condition (2.11), $y_i^{\cap+} + y_j^{\cap+}$ must be equal to $y_i^\cap + y_j^\cap$, and so $w^+ = w$, yielding that

$$y_i^{\cap+} = w - v \quad \text{and} \quad y_j^{\cap+} = w + v, \tag{2.18}$$

for some vector $v = \frac{1}{2}(y_j^\cap - y_i^\cap) + u$.

As indicated, we require that the norms of these two vectors are bounded above by the constant $M$. First consider the case where $s^\cap$ and $w$ are linearly independent. Given that

$$\|w \pm v\|^2 \le \|w\|^2 + \|v\|^2 + 2\,|\langle w, v\rangle|\,,$$

we choose $v$ to zero the term $\langle w, v\rangle$ to limit the values of these norms. In this situation, consider a fixed improvement in $\mu_j$. The value of $\langle s^\cap, v\rangle$ is then fixed, and the norm of $y_j^{\cap+}$ (which is now the same as that of $y_i^{\cap+}$) is minimal if the vector $v$ lies in the plane $\Pi$ spanned by $s^\cap$ and $w$. Indeed, consider the orthogonal decomposition of $v$ as

$$P_\Pi v + (I - P_\Pi)v,$$

where $P_\Pi$ is the orthogonal projection on $\Pi$. Then, the squared norm

$$\|y_j^{\cap+}\|^2 = \|w\|^2 + \|P_\Pi v\|^2 + \|(I - P_\Pi)v\|^2 \tag{2.19}$$

has to be minimal with $\langle s^\cap, v\rangle$ fixed and $\langle w, v\rangle$ equal to zero (due to the orthogonality assumption on $v$ and $w$). But, $\langle s^\cap, v\rangle = \langle s^\cap, P_\Pi v\rangle$ and $\langle w, v\rangle = \langle w, P_\Pi v\rangle$ because $s^\cap$ and $w$ lie in $\Pi$. So the two constraints determine only (but completely) the component of $v$ lying in the plane $\Pi$. The minimum of (2.19) is thus reached when $(I - P_\Pi)v$ is zero. We now give an explicit expression of vector $v$, knowing that it lies in plane $\Pi$ and is perpendicular to $w$. Observe that these conditions define a one-dimensional space and are fulfilled by $(I - P_w)s^\cap$. Therefore,

$$v = \beta \left( I - \frac{ww^T}{w^T w} \right) s^\cap \tag{2.20}$$

for some scalar $\beta$.

Now consider the other case where $s^{\cap}$ and $w$ are collinear. Choosing $v$ orthogonal to $w$ (and thus to $s^{\cap}$) prevents increasing $\mu_j$. Then, we just set

$$v = \beta s^{\cap}, \tag{2.21}$$

for some scalar $\beta$, enabling a direct increase of $\langle s^{\cap}, v\rangle$, and thus of $\mu_j$.

In both cases, the positive scalar $\beta$ is determined such that

$$\|y_i^{\cap +}\| \leq M \quad \text{and} \quad \|y_j^{\cap +}\| \leq M, \tag{2.22}$$

and that $\left\langle s^{\cap}, y_j^{\cap +}\right\rangle$ is bounded above by what is needed to get the desired increase $\delta$ in $\mu_j$:

$$\left\langle s^{\cap}, y_j^{\cap +}\right\rangle \leq \left\langle s^{\cap}, y_j^{\cap}\right\rangle + \delta \left\langle s, H_j s\right\rangle . \tag{2.23}$$

A lower bound is also imposed on $\left\langle s^{\cap}, y_i^{\cap +}\right\rangle$ to prevent it to become (too) negative:

$$\left\langle s^{\cap}, y_i^{\cap +}\right\rangle \geq -\left\langle s^{\setminus}, y_i^{\setminus}\right\rangle + \tau \left\langle s, H_i s\right\rangle , \tag{2.24}$$

where $\tau$ determines how negative $\mu_i^{+}$ may become. Obviously, we also impose that

$$\mu_j^{+} > \mu_j . \tag{2.25}$$

The updated vectors $y_i^{+}$ and $y_j^{+}$ are then given by (2.18), and (2.20) or (2.21) with the maximal value of $\beta$ satisfying the conditions (2.22) to (2.25) (if there are compatible).

Note that requiring $v$ to be orthogonal to $w$ can be a restrictive choice when $M$ is sufficiently large (more precisely, when the intersection of lines $w + \mathbb{R}v$ and $\mathbb{R}s^{\cap}$ lies inside the ball of radius $M$ centred at the origin). We then turn back to (2.21) even if $s^{\cap}$ and $w$ are not collinear. This completes the push operation description.

This process without condition (2.23) is also used to initialize the arcs capacities and to update every arc adjacent after a push.

**Balanced Partitioned BFGS Update.** Practically, the curvature balancing process can stop prematurely with nodes still deficient, because the constraints (2.22) to (2.25) become incompatible and thus the arc capacities reduce to zero. Therefore, some additional heuristics were tested to accept or refuse the generated set of element gradient differences. The first one was to keep the new version of vector $y_i$ only if the corresponding $\mu_i$ was initially positive or was not decreased. The second one was to monitor each element secant equation and to update the element gradient difference $y_i$ only if the corresponding $i$-th element secant equation was not deteriorated with respect to the initial set of vectors $y_i$. The best results were obtained by combining these heuristics: each $y_i$ was updated only if both rules were satisfied. This obviously generates a relaxation of the summation condition (2.11), which is equivalent to relaxing the full secant equation.

The last algorithm (Algorithm 2.5 on the following page) which we propose to test therefore consists in uniformly splitting the gradient difference $y$ and then applying the elemental BFGS updates on the basis of the element gradient differences $y_i$ balanced using the heuristics described above.

## 3 Numerical experiments

We performed our numerical experiments inside the *Recursive Multilevel Trust-Region* (RMTR) method of Gratton *et al.* (2008*c*). More precisely, we used the Fortran 95 code written by D. Tomanos

```
Algorithm 2.5: Balanced Partitioned BFGS Update

Step 1. Decompose vector $s$ into $\{s_i\}_{i=1}^m$ and split uniformly $y$ into $\{y_i\}_{i=1}^m$.
Step 2. Apply the push-relabel heuristic on the curvature flow network described above, in
        order to balance the vectors $y_i$.
Step 3. For each element $i$, check for initial positivity or improvement of $\mu_i$, and non-
        deterioration of the $i$-th element secant equation. If these conditions fail, use the initial
        vector $y_i$.
Step 4. Perform the BFGS update (2.16) on each $H_i$ for which $\langle s_i, y_i \rangle \geq 10^{-6} \langle s_i, H_i si \rangle$.
```

and implementing the infinite-norm version of that algorithm (see Gratton *et al.*, 2008*b*). All codes were written in Fortran 95 and experiments were conducted on a 3.40 GHz Intel dual-core processor computer with 2 Gbytes of RAM.

We used the RMTR code with the parameters values advised by Gratton, Mouffe, Sartenaer, Toint and Tomanos (2008*a*)). Our modifications to this code intent to deal with approximate Hessians. So, instead of reevaluating the Hessian when needed, it was updated using one of the compared updating procedure:

- the lower triangular substitution method (LTS, see Algorithm 2.1);
- the lower triangular substitution method (LTS-O) using the optimal columns groups from Goldfarb and Toint (1984);
- the sparse PSB update (S-PSB, see Algorithm 2.2);
- the partially separable PSB update (PS-PSB, see Algorithm 2.3);
- the partitioned BFGS update with the uniform splitting of the gradient differences (UP-BFGS, see Algorithm 2.4);
- the partitioned BFGS update with the balanced splitting of the gradient differences (BP-BFGS, see Algorithm 2.5).

Observe that the element gradients and Hessians need to be stored for the UP-BFGS and BP-BFGS choices, at variance with the PS-PSB option.

Except for the LTS and LTS-O methods, the initial Hessian at each level of the algorithm has to be estimated rather than evaluated. For the UP-BFGS and BP-BFGS options, we choose to set every element Hessian $H_i$ to the identity matrix, the full initial Hessian being thus defined as the sum of these element Hessians. The same matrix is chosen for S-PSB and PS-PSB (without the need to initialize the $H_i$ explicitely).

**Practicalities.** In the LTS-based algorithms, we fixed the increment for the finite difference computation at $10^{-8}$. For LTS-O, the covering molecules are described in Appendix B. The linear systems in the S-PSB and the PS-PSB algorithms are solved by a preconditioned conjugate gradient method. The chosen preconditioner is given by the diagonal term $\mathrm{diag}(\mathcal{P}(\Omega^{\maltese})(s \bullet s))$ for S-PSB, and $\sum_{i \in \mathcal{I}} \omega_i^{-1} \|s_i\|^2 I_i$ for PS-PSB. The required precision on the solution $\lambda$ was set to $10^{-6}$. In the BP-BFGS algorithm, our implementation of the push-relabel algorithm is mainly inspired of the *highest-label push-relabel* (HIPR) code of A. V. Goldberg. The bound $M$ on the norms was set to $\|y\|$, $\epsilon$ to 0.1 and $\tau$ to 0. Finally, except for the LTS and LTS-O algorithms, the $i$-th element Hessian $H_i$ was not updated if the norm of the corresponding element step $s_i$ was smaller than $10^{-6}$ times the norm of the step $s$. This is to prevent bad conditioning of the linear system for the PSB-like updates, and numerical errors in the partitioned BFGS updates. Besides, as nearly no information can be collected in the directions given by these $s_i$, it does not hurt much to forget them.

**Test problems.** We have considered minimization problems posed in infinite-dimensional spaces and involving differential operators. We refer to Gratton *et al.* (2008*a*) for a detailed description.

| Problem name | Sizes | Comment |
|---|---:|---|
| P2D | 261121, 1046529, 4190209 | 2-D, quadratic |
| P3D | 29791, 250047 | 3-D, quadratic |
| DEPT | 261121, 1046529, 4190209 | 2-D, quadratic, (Minpack 2) |
| DPJB | 261121, 1046529, 4190209 | 2-D, quadratic, with bound constraints (Minpack 2) |
| DODC | 261121, 1046529, 4190209 | 2-D, convex, (Minpack 2) |
| MINS-SB | 65025, 261121, 1046529 | 2-D, convex, smooth boundary conditions |
| MINS-OB | 65025, 261121, 1046529 | 2-D, convex, oscillatory boundary conditions |
| MINS-DMSA | 65025, 261121, 1046529 | 2-D, convex, (Minpack 2) |
| IGNISC | 261121, 1046529 | 2-D, convex |
| DSSC | 261121, 1046529, 4190209 | 2-D, convex, (Minpack 2) |
| BRATU | 65025, 261121, 1046529 | 2-D, convex |
| MEMBR | 261121, 1046529, 4190209 | 2-D, convex, free boundary, with bound constraints |
| NCCS | 7938, 32258, 130050 | 2-D, nonconvex, smooth boundary conditions |
| NCCO | 32258, 130050, 522242 | 2-D, nonconvex, oscillatory boundary conditions |
| MOREBV | 65025, 261121, 1046529 | 2-D, nonconvex |

Table 3.1: Test problems characteristics

**Results.** As function and gradient evaluations do not cost the same at each level, we defined an equivalent number of evaluations (see Gratton *et al.*, 2008*a*) given by

$$q = \sum_{\text{level } \ell} q_\ell \frac{n_\ell}{n}$$

where $q_\ell$ and $n_\ell$ are respectively the number of evaluations and the size of the problem at level $\ell$. The results of the numerical experience are given in Appendix C. For a better readability, we display these results as performance profiles (see Dolan and Moré, 2002). In Figure 1, we take the number of function evaluations plus five times the number of gradient evaluations as comparison criterion; this ratio seems appropriate in view of Griewank (1989). In Figures 2 and 3, we compare the CPU time and the equivalent number of Hessian updates, respectively.

The graphs shows that the two LTS-based methods and the two PSB-type updates are clearly more efficient and robust than the partitioned BFGS updating procedures.

We observe that if an optimal column grouping is available, it is worth exploiting it, because it allows an important reduction of the number of gradient evaluations. A reduction of the cpu time is also observed, but it corresponds mainly to the smaller number of gradient evaluations. Indeed, as both methods give a relatively accurate approximation of the Hessian, the number of iterations requiring an Hessian update are quite similar.

In term of robustness, the LTS-based methods and the PSB-type updates give similar results, with a small advantage for the LTS-based methods. Remember that the LTS-based methods require more gradient evaluations than the PSB-type updates at each iteration where the Hessian is

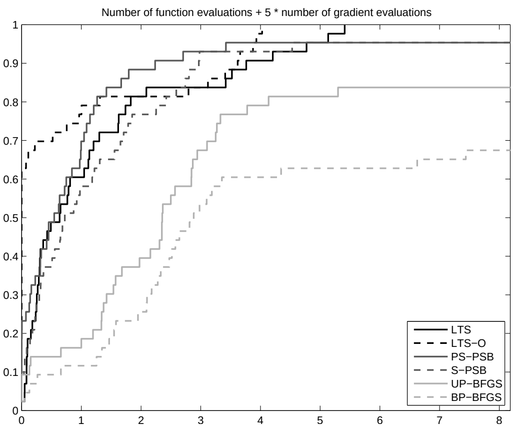

Figure 1: Performance profiles based on the number of function evaluations plus five times the number of gradient evaluations.

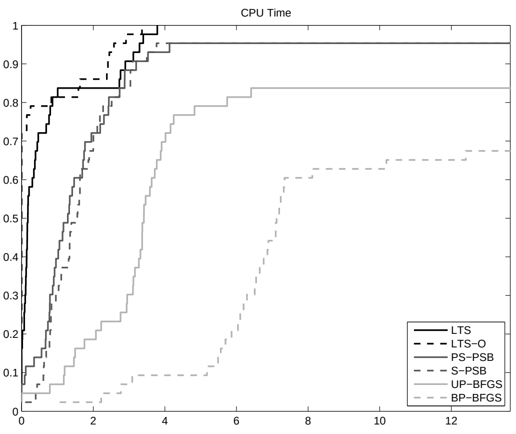

Figure 2: Performance profiles based on the CPU time.

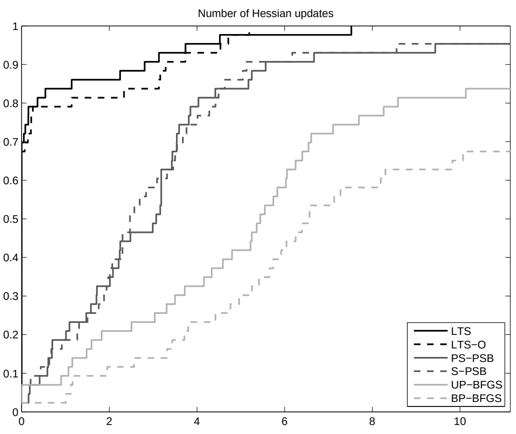

Figure 3: Performance profiles based on the number of Hessian updates.

(approximately) recomputed. This should be a disadvantage, but as shown in Figure 3, the better accuracy obtained with the LTS-based methods reduces the number of iterations where the Hessian needs to be recomputed in RMTR. We obtain similar results in term of function and gradient evaluations (and in fact, better results for the LTS-O variant as it requires less gradient evaluations). If we now consider Figure 2, we observe that the LTS-based methods are faster than the PSB-type updates. This seems to correspond to the additional iterations for the PSB-type updates and to the fact that the LTS-based methods solve a sparse *triangular* linear system instead of a sparse general linear system. A small advantage is also observed for the partially separable PSB update with respect to the sparse one.

Regarding the two partitioned BFGS updating process, we note that, surprisingly, the unbalanced version is a little more efficient and robust. Clearly, the balancing procedure is also too expensive to produce good results in CPU time. This initially interesting idea thus appears not to give the associated advantages.

## 4 Conclusion and perspectives

We have reviewed existing methods and presented new methods to update the Hessian of structured large-scale functions present in multilevel optimization. A numerical comparison has been reported and it indicates that the LTS methods is a good choice in term of effectiveness and robustness.

Our test problems set is still rather small and new problems are sought. In particular, problems with different structures than the current ones (e.g. denser) could allow a finer comparison of the methods.

## References

R. K. Ahuja, M. Kodialam, A. K. Mishra, and J. B. Orlin. Computational Investigation of Maximum Flow Algorithms. *European Journal on Operational Research*, **97**, 509–542, 1997.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows, Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and its Applications*, **6**, 76–90, 1970.

B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, **19**(4), 390–410, 1997.

T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, **28**, 243–270, 1984.

T. F. Coleman, B. Garbow, and J. J. Moré. Software for estimating sparse Hessian matrices. *ACM Transactions on Mathematical Software*, **11**, 363–378, 1985.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A).* Number 17 *in* 'Springer Series in Computational Mathematics'. Springer Verlag, Heidelberg, Berlin, New York, 1992.

A. Curtis, M. J. D. Powell, and J. Reid. On the estimation of sparse Jacobian matrices. *Journal of the Institute of Mathematics and its Applications*, **13**, 117–119, 1974.

W. C. Davidon. Variable metric method for minimization. Report ANL-5990(Rev.), Argonne National Laboratory, Research and Development, 1959. republished in the SIAM Journal on Optimization, vol. 1, pp. 1–17, 1991.

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.

E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213, 2002.

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

R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, **13**, 317–322, 1970.

A. R. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery*, **35**(4), 921–940, October 1988.

D. Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computation*, **24**, 23–26, 1970.

D. Goldfarb and Ph. L. Toint. Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations. *Mathematics of Computation*, **43**(167), 69–88, 1984.

S. Gratton, M. Mouffe, A. Sartenaer, Ph. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. Technical Report 08/10, Department of Mathematics, University of Namur, Namur, Belgium, june 2008*a*.

S. Gratton, M. Mouffe, Ph. L. Toint, and M. Weber-Mendonça. A recursive trust-region method in infinity norm for bound-constrained nonlinear optimization. *IMA Journal of Numerical Analysis*, **(to appear)**, 2008*b*.

S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive Trust-Region Methods for Multiscale Nonlinear Optimization. *SIAM Journal on Optimization*, **19**(1), 414–444, 2008*c*.

A. Griewank. On automatic differentiation. *in* M. Iri and K. Tanabe, eds, 'Mathematical Programming: Recent Developments and Applications', pp. 83–108, Dordrecht, The Netherlands, 1989. Kluwer Academic Publishers.

A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. *in* M. J. D. Powell, ed., 'Nonlinear Optimization 1981', pp. 301–312, London, 1982*a*. Academic Press.

A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, **39**, 119–137, 1982*b*.

E. Marwil. *Exploiting sparsity in Newton-like methods*. PhD thesis, Cornell University, Ithaca, USA, 1978.

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

J. Nocedal and S. J. Wright. *Numerical Optimization*. Series in Operations Research. Springer Verlag, Heidelberg, Berlin, New York, 1999.

S. Oh, A. Milstein, C. Bouman, and K. Webb. Multigrid algorithms for optimization and inverse problems. *in* C. Bouman and R. Stevenson, eds, 'Computational Imaging', Vol. 5016 of *Proceedings of the SPIE*, pp. 59–70. DDM, 2003.

M. J. D. Powell and Ph. L. Toint. On the estimation of sparse Hessian matrices. *SIAM Journal on Numerical Analysis*, **16**(6), 1060–1074, 1979.

D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, **24**, 647–657, 1970.

D. C. Sorensen. An example concerning quasi-Newton estimates of a sparse Hessian. *SIGNUM Newsletter*, **16**(2), 8–10, 1981.

Ph. L. Toint. On sparse and symmetric matrix updating subject to a linear equation. *Mathematics of Computation*, **31**(140), 954–961, 1977.

Ph. L. Toint. On the superlinear convergence of an algorithm for solving a sparse minimization problem. *SIAM Journal on Numerical Analysis*, **16**, 1036–1045, 1979.

Ph. L. Toint. A note on sparsity exploiting quasi-Newton methods. *Mathematical Programming*, **21**(2), 172–181, 1981.

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.

## A Link between sparse and partially separable PSB updates

Algorithm 2.2 and 2.3 turns out to differ only by the choice of the norm used in the variational problem. More precisely, they produce the same update if we defined the weigh matrix $\Omega$ in Algorithm 2.2 such that

$$\Omega^{\maltese} = \sum_{i=1}^{m} \omega_i^{-1} J_i. \tag{1.26}$$

Indeed, for the particular choice of $\Omega$, the matrix $S$ is identical for both considered algorithms:

$$\begin{aligned} S &= \mathcal{P}\left(ss^T \bullet \sum_{i=1}^{m} \omega_i^{-1} J_i\right) + \operatorname{diag}\left(\mathcal{P}\left(\sum_{i=1}^{m} \omega_i^{-1} J_i\right)(s \bullet s)\right) \\ &= \mathcal{P}\left(\sum_{i=1}^{m} \omega_i^{-1} (ss^T \bullet J_i)\right) + \sum_{i=1}^{m} \omega_i^{-1} \operatorname{diag}\left(J_i(s \bullet s)\right) \\ &= \sum_{i=1}^{m} \omega_i^{-1} s_i s_i^T + \sum_{i=1}^{m} \omega_i^{-1} \operatorname{diag}\left(\|s_i\|^2 e_i\right) = \sum_{i=1}^{m} \omega_i^{-1}\left(s_i s_i^T + \|s_i\|^2 I_i\right), \end{aligned}$$

where $e_i = \operatorname{diag}(I_i)$ is a vector with value 1 for components indexed by $\mathcal{I}_i$ and value 0 elsewhere. It now remains to show that the correction $E \stackrel{\text{def}}{=} H^+ - H$ is the same in both cases:

$$E = \mathcal{P}\left((s\lambda^T + \lambda s^T) \bullet \sum_{i=1}^{m} \omega_i^{-1} J_i\right) = \mathcal{P}\left(\sum_{i=1}^{m} \omega_i^{-1} (s\lambda^T + \lambda s^T) \bullet J_i\right) = \sum_{i=1}^{m} \omega_i^{-1}\left(s_i \lambda_i^T + \lambda_i s_i^T\right).$$

As a consequence of this result, the convergence theory presented by Toint (1979) (in the unweighted case) immediately adapt to provide global and local convergence with superlinear speed for Algorithm 2.3 embedded inside a trust-region method.

## B Covering molecules for LTS-O

The multi-diagonal structure of the Hessian in our problems can be represented with computational molecules, whose size could be reduced by taking into account the symmetry of the Hessian. Goldfarb and Toint (1984) define these reduced molecules (see details in their paper) and use them to cover the discretization grid space (reduced molecules are shaded in the following pictures); each (Hessian column) group then consists in all nodes of the grid covered by the same node of the molecule.

Problems P2D, BRATU et MEMBR arise from a 5-point finite difference Laplacian operator; this gives a 5-diagonal Hessian, whose cover is displayed in Figure 4 (left). The Hessian of the 3-dimensional Laplacian problem P3D consists in 7 diagonals and an horizontal layer of its cover is represented in Figure 4 (center); the molecules are indeed tetrahedrons. In problems DEPT, DPJB, DODC, MINS-SB, MINS-OB, MINS-DMSA and DSSC, the Hessian has also 7 diagonal and its cover is displayed in Figure 4 (right).

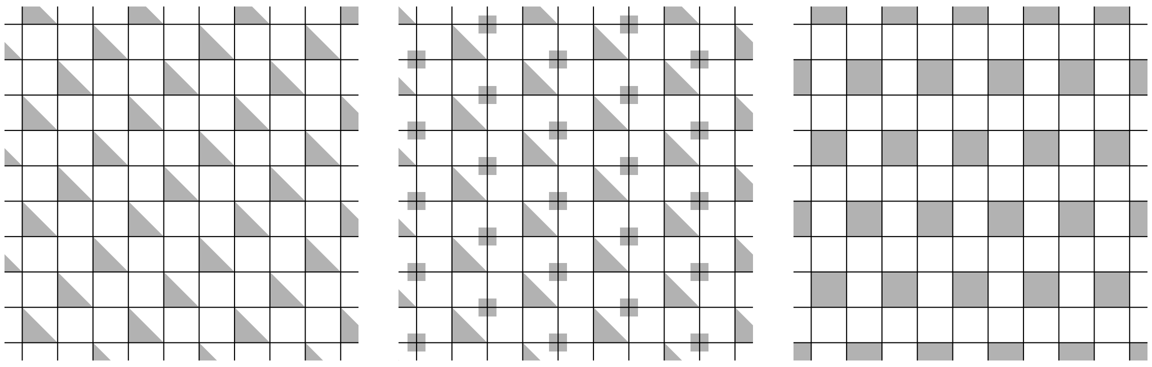
Figure 4: Covers for a 5-diagonal Hessian (left), a 3-dimensional 7-diagonal Hessian (horizontal layer, in the center) and a 7-diagonal Hessian (right).

For problems IGNISC and MOREBV, we have a 13-diagonal Hessian, whose cover is displayed in Figure 5 (left). Finally, while Goldfarb and Toint (1984) present no cover for the two problems NCCS and NCCO, we describe one in Figure 5. As these problems use two set of variables, the cover is also defined on two layers, the first one corresponding to that of a 13-diagonal Hessian.

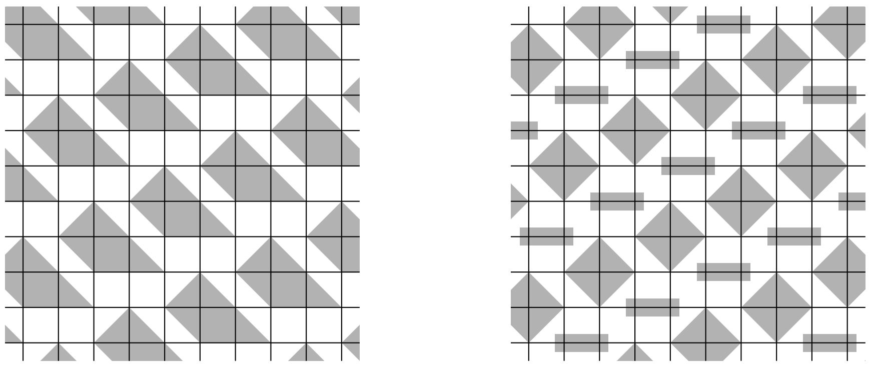
Figure 5: Covers for a 13-diagonal Hessian (left), and for NCCS/NNCO problems (first set of variables on the left, second one on the right).

## C Complete numerical results

In the next table, we report the CPU time (cpu), the equivalent number of function (nf), gradient (ng) and Hessian (nh) evaluations. The symbol "–" indicates that the iteration limit (fixed at 10,000) or the time limit (fixed at 1 hour) was exceeded.

| P2D (261121) | CPU | nf | ng | nh | DEPT (261121) | CPU | nf | ng | nh |
|---|---|---|---|---|---|---|---|---|---|
| LTS | 0.88 | 1.88 | 2.47 | 0.08 | LTS | 0.58 | 1.67 | 2.15 | 0.08 |
| LTS-O | 0.80 | 1.88 | 1.96 | 0.08 | LTS-O | 0.55 | 1.67 | 1.90 | 0.08 |
| S-PSB | 2.43 | 5.24 | 3.55 | 1.07 | S-PSB | 1.18 | 3.91 | 2.49 | 0.54 |
| P-PSB | 2.71 | 5.32 | 3.76 | 1.17 | P-PSB | 1.38 | 4.28 | 2.81 | 0.70 |
| UP-BFGS | 12.90 | 22.96 | 15.50 | 7.04 | UP-BFGS | 10.41 | 27.58 | 16.91 | 7.78 |
| BP-BFGS | 127.24 | 22.87 | 15.44 | 7.01 | BP-BFGS | 74.94 | 27.85 | 17.24 | 7.91 |
| P2D (1046529) | CPU | nf | ng | nh | DEPT (1046529) | CPU | nf | ng | nh |
| LTS | 3.69 | 1.70 | 2.35 | 0.08 | LTS | 2.07 | 1.47 | 1.57 | 0.02 |
| LTS-O | 3.32 | 1.70 | 1.84 | 0.08 | LTS-O | 2.04 | 1.47 | 1.51 | 0.02 |
| S-PSB | 5.83 | 2.48 | 2.09 | 0.34 | S-PSB | 3.55 | 2.20 | 1.81 | 0.23 |
| P-PSB | 6.28 | 2.60 | 2.19 | 0.39 | P-PSB | 3.73 | 2.19 | 1.81 | 0.23 |
| UP-BFGS | 34.17 | 12.46 | 8.67 | 3.63 | UP-BFGS | 27.96 | 11.06 | 7.07 | 2.86 |
| BP-BFGS | 243.75 | 12.29 | 8.52 | 3.55 | BP-BFGS | 134.64 | 12.65 | 7.79 | 3.22 |
| P2D (4190209) | CPU | nf | ng | nh | DEPT (4190209) | CPU | nf | ng | nh |
| LTS | 13.33 | 1.43 | 1.59 | 0.02 | LTS | 8.21 | 1.38 | 1.52 | 0.02 |
| LTS-O | 11.54 | 1.43 | 1.46 | 0.02 | LTS-O | 7.80 | 1.38 | 1.46 | 0.02 |
| S-PSB | 17.70 | 1.70 | 1.59 | 0.12 | S-PSB | 13.39 | 2.04 | 1.60 | 0.12 |
| P-PSB | 19.19 | 1.71 | 1.59 | 0.12 | P-PSB | 15.85 | 2.55 | 1.86 | 0.19 |
| UP-BFGS | – | – | – | – | UP-BFGS | 93.50 | 5.32 | 3.58 | 1.11 |
| BP-BFGS | – | – | – | – | BP-BFGS | – | – | – | – |
| DPJB (261121) | CPU | nf | ng | nh | DODC (261121) | CPU | nf | ng | nh |
| LTS | 0.69 | 1.87 | 2.48 | 0.08 | LTS | 2.41 | 4.54 | 13.68 | 1.13 |
| LTS-O | 0.62 | 1.87 | 2.05 | 0.08 | LTS-O | 14.39 | 42.78 | 90.04 | 15.01 |
| S-PSB | 2.47 | 6.77 | 4.65 | 1.62 | S-PSB | 4.68 | 13.63 | 6.91 | 2.71 |
| P-PSB | 2.38 | 6.17 | 4.13 | 1.36 | P-PSB | 4.68 | 14.11 | 6.46 | 2.40 |
| UP-BFGS | 6.56 | 13.99 | 9.22 | 3.90 | UP-BFGS | 68.43 | 165.18 | 94.23 | 46.30 |
| BP-BFGS | 84.27 | 15.74 | 10.31 | 4.45 | BP-BFGS | 668.77 | 255.94 | 137.65 | 68.17 |
| DPJB (1046529) | CPU | nf | ng | nh | DODC (1046529) | CPU | nf | ng | nh |
| LTS | 2.73 | 1.63 | 2.31 | 0.08 | LTS | 6.33 | 2.40 | 5.75 | 0.40 |
| LTS-O | 2.57 | 1.63 | 1.89 | 0.08 | LTS-O | 65.05 | 34.87 | 88.18 | 14.57 |
| S-PSB | 7.56 | 4.91 | 3.08 | 0.83 | S-PSB | 16.18 | 7.54 | 5.65 | 2.21 |
| P-PSB | 8.45 | 5.31 | 3.35 | 0.97 | P-PSB | 13.98 | 6.35 | 4.56 | 1.69 |
| UP-BFGS | 25.65 | 11.30 | 7.63 | 3.11 | UP-BFGS | 537.42 | 315.22 | 166.44 | 82.52 |
| BP-BFGS | 234.51 | 11.37 | 7.77 | 3.18 | BP-BFGS | – | – | – | – |
| DPJB (4190209) | CPU | nf | ng | nh | DODC (4190209) | CPU | nf | ng | nh |
| LTS | 11.28 | 1.46 | 1.61 | 0.02 | LTS | 24.72 | 2.23 | 4.45 | 0.29 |
| LTS-O | 10.11 | 1.46 | 1.50 | 0.02 | LTS-O | 186.52 | 18.71 | 40.99 | 6.65 |
| S-PSB | 31.38 | 2.86 | 2.25 | 0.45 | S-PSB | 38.18 | 4.16 | 2.81 | 0.81 |
| P-PSB | 22.73 | 2.52 | 1.95 | 0.30 | P-PSB | 42.76 | 4.46 | 2.77 | 0.80 |
| UP-BFGS | – | – | – | – | UP-BFGS | – | – | – | – |
| BP-BFGS | – | – | – | – | BP-BFGS | – | – | – | – |
| MINS-SB (65025) | CPU | nf | ng | nh | MINS-OB (65025) | CPU | nf | ng | nh |
| LTS | 10.70 | 86.05 | 295.36 | 24.28 | LTS | 3.96 | 29.22 | 111.59 | 10.10 |
| LTS-O | 9.43 | 99.05 | 202.89 | 28.28 | LTS-O | 2.93 | 29.22 | 61.04 | 10.10 |
| S-PSB | 45.65 | 914.76 | 518.48 | 45.93 | S-PSB | 13.33 | 141.54 | 82.13 | 39.73 |
| P-PSB | 31.61 | 555.84 | 362.48 | 39.45 | P-PSB | 15.85 | 173.29 | 98.77 | 48.06 |
| UP-BFGS | 84.65 | 653.59 | 582.71 | 199.22 | UP-BFGS | 22.37 | 178.39 | 117.47 | 57.41 |
| BP-BFGS | 947.50 | 840.55 | 781.92 | 338.50 | BP-BFGS | 347.34 | 187.87 | 122.08 | 59.71 |
| MINS-SB (261121) | CPU | nf | ng | nh | MINS-OB (261121) | CPU | nf | ng | nh |
| LTS | 53.04 | 99.15 | 364.73 | 28.70 | LTS | 51.16 | 90.12 | 398.27 | 33.65 |
| LTS-O | 41.01 | 104.52 | 205.17 | 27.70 | LTS-O | 28.13 | 60.37 | 156.01 | 26.15 |
| S-PSB | 234.72 | 987.57 | 617.88 | 68.79 | S-PSB | 57.59 | 154.79 | 86.25 | 41.79 |
| P-PSB | 201.77 | 894.33 | 539.26 | 55.88 | P-PSB | 58.44 | 151.90 | 87.12 | 42.23 |
| UP-BFGS | 2191.21 | 3359.43 | 3306.73 | 1579.73 | UP-BFGS | 79.25 | 149.86 | 100.29 | 48.81 |
| BP-BFGS | – | – | – | – | BP-BFGS | 1454.49 | 164.08 | 108.05 | 52.69 |
| MINS-SB (1046529) | CPU | nf | ng | nh | MINS-OB (1046529) | CPU | nf | ng | nh |
| LTS | 227.83 | 99.26 | 375.90 | 29.42 | LTS | 184.29 | 75.51 | 352.23 | 29.47 |
| LTS-O | 173.90 | 102.67 | 209.02 | 30.86 | LTS-O | 114.60 | 71.76 | 158.60 | 26.47 |
| S-PSB | 1429.00 | 1589.16 | 912.60 | 99.64 | S-PSB | 174.74 | 113.10 | 63.72 | 30.53 |
| P-PSB | 1279.08 | 1382.69 | 804.51 | 96.06 | P-PSB | 213.70 | 160.04 | 83.40 | 40.37 |
| UP-BFGS | 3108.23 | 1117.39 | 1074.15 | 525.82 | UP-BFGS | 482.94 | 276.23 | 162.65 | 79.99 |
| BP-BFGS | – | – | – | – | BP-BFGS | – | – | – | – |
| MINS-DMSA (65025) | CPU | nf | ng | nh | DSSC (261121) | CPU | nf | ng | nh |
| LTS | 3.98 | 27.09 | 98.37 | 8.91 | LTS | 0.80 | 1.68 | 2.33 | 0.08 |
| LTS-O | 21.19 | 195.10 | 522.04 | 87.02 | LTS-O | 0.71 | 1.68 | 1.90 | 0.08 |
| S-PSB | 11.79 | 103.27 | 70.95 | 34.15 | S-PSB | 1.77 | 3.79 | 2.84 | 0.71 |
| P-PSB | 10.60 | 97.58 | 61.29 | 29.31 | P-PSB | 1.81 | 3.49 | 2.74 | 0.66 |
| UP-BFGS | 34.71 | 277.40 | 179.19 | 88.27 | UP-BFGS | 10.51 | 19.99 | 12.32 | 5.45 |
| BP-BFGS | 463.76 | 279.24 | 180.39 | 88.86 | BP-BFGS | 77.21 | 20.38 | 12.42 | 5.50 |
| MINS-DMSA (261121) | CPU | nf | ng | nh | DSSC (1046529) | CPU | nf | ng | nh |
| LTS | 30.89 | 50.43 | 204.22 | 18.56 | LTS | 2.87 | 1.48 | 1.62 | 0.02 |
| LTS-O | 95.51 | 207.78 | 562.36 | 93.73 | LTS-O | 2.83 | 1.48 | 1.51 | 0.02 |
| S-PSB | 40.61 | 105.21 | 60.13 | 28.73 | S-PSB | 4.55 | 2.08 | 1.84 | 0.24 |
| P-PSB | 33.50 | 87.02 | 51.99 | 24.66 | P-PSB | 4.88 | 2.08 | 1.86 | 0.25 |
| UP-BFGS | 104.21 | 207.93 | 134.25 | 65.82 | UP-BFGS | 24.42 | 8.48 | 5.42 | 2.04 |
| BP-BFGS | 1460.76 | 227.11 | 146.39 | 71.86 | BP-BFGS | 126.81 | 8.38 | 5.32 | 1.98 |
| MINS-DMSA (1046529) | CPU | nf | ng | nh | DSSC (4190209) | CPU | nf | ng | nh |
| LTS | 119.48 | 49.11 | 191.30 | 17.39 | LTS | 11.78 | 1.39 | 1.56 | 0.02 |
| LTS-O | 652.20 | 334.14 | 934.21 | 155.70 | LTS-O | 10.85 | 1.39 | 1.46 | 0.02 |
| S-PSB | 128.06 | 83.01 | 48.96 | 23.52 | S-PSB | 14.65 | 1.66 | 1.51 | 0.08 |
| P-PSB | 126.76 | 92.12 | 54.96 | 26.15 | P-PSB | 15.95 | 1.72 | 1.55 | 0.10 |
| UP-BFGS | 329.05 | 157.77 | 99.53 | 48.62 | UP-BFGS | – | – | – | – |
| BP-BFGS | – | – | – | – | BP-BFGS | – | – | – | – |
| P3D (29791) | CPU | nf | ng | nh | IGNISC (261121) | CPU | nf | ng | nh |
| LTS | 1.86 | 6.19 | 21.96 | 1.14 | LTS | 5.57 | 17.10 | 49.60 | 4.47 |
| LTS-O | 1.10 | 6.19 | 9.45 | 1.14 | LTS-O | 0.82 | 1.97 | 4.22 | 0.34 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| S-PSB | 12.36 | 88.61 | 53.73 | 25.74 | S-PSB | 7.02 | 17.25 | 11.63 | 5.41 |
| P-PSB | 12.67 | 92.79 | 51.12 | 24.43 | P-PSB | 6.02 | 14.94 | 9.36 | 3.90 |
| UP-BFGS | 11.30 | 74.84 | 48.45 | 23.11 | UP-BFGS | 6.33 | 14.17 | 9.12 | 3.79 |
| BP-BFGS | 152.70 | 78.63 | 51.04 | 24.47 | BP-BFGS | 123.64 | 14.30 | 8.86 | 3.63 |
| P3D (250047) | CPU | nf | ng | nh | IGNISC (1046529) | CPU | nf | ng | nh |
| LTS | 18.31 | 6.03 | 21.87 | 1.14 | LTS | 35.55 | 22.88 | 80.71 | 7.67 |
| LTS-O | 10.52 | 6.03 | 9.31 | 1.14 | LTS-O | 3.63 | 1.99 | 4.22 | 0.33 |
| S-PSB | 143.50 | 119.03 | 59.15 | 28.44 | S-PSB | 29.84 | 21.04 | 11.69 | 4.53 |
| P-PSB | 184.30 | 139.67 | 84.77 | 41.24 | P-PSB | 24.18 | 14.48 | 8.19 | 3.59 |
| UP-BFGS | 130.35 | 90.27 | 57.34 | 27.53 | UP-BFGS | 35.11 | 17.43 | 10.26 | 4.40 |
| BP-BFGS | – | – | – | – | BP-BFGS | 588.85 | 17.33 | 10.35 | 4.38 |
| BRATU (65025) | CPU | nf | ng | nh | MEMBR (261121) | CPU | nf | ng | nh |
| LTS | 0.35 | 2.57 | 5.26 | 0.33 | LTS | 96.29 | 146.65 | 554.56 | 46.75 |
| LTS-O | 0.25 | 2.57 | 3.20 | 0.33 | LTS-O | 75.95 | 191.72 | 256.87 | 51.13 |
| S-PSB | 1.01 | 9.99 | 6.44 | 2.39 | S-PSB | 163.53 | 686.03 | 377.24 | 44.12 |
| P-PSB | 0.70 | 7.10 | 4.36 | 1.35 | P-PSB | 96.66 | 234.36 | 128.07 | 50.20 |
| UP-BFGS | 3.78 | 32.25 | 20.28 | 9.31 | UP-BFGS | 171.86 | 391.21 | 197.86 | 98.27 |
| BP-BFGS | 36.87 | 31.46 | 19.82 | 9.08 | BP-BFGS | 645.75 | 387.99 | 198.45 | 98.55 |
| BRATU (261121) | CPU | nf | ng | nh | MEMBR (1046529) | CPU | nf | ng | nh |
| LTS | 0.85 | 1.73 | 2.36 | 0.08 | LTS | 163.75 | 45.57 | 166.86 | 14.16 |
| LTS-O | 0.78 | 1.73 | 1.84 | 0.08 | LTS-O | 152.82 | 54.98 | 71.55 | 14.06 |
| S-PSB | 2.05 | 4.25 | 3.31 | 0.95 | S-PSB | 661.63 | 644.41 | 330.95 | 12.74 |
| P-PSB | 1.92 | 3.84 | 2.91 | 0.75 | P-PSB | 242.78 | 159.23 | 79.63 | 13.75 |
| UP-BFGS | 10.31 | 21.68 | 12.50 | 5.54 | UP-BFGS | 351.39 | 107.60 | 54.81 | 26.71 |
| BP-BFGS | 73.41 | 15.99 | 9.85 | 4.22 | BP-BFGS | 1045.92 | 110.47 | 56.86 | 27.73 |
| BRATU (1046529) | CPU | nf | ng | nh | MEMBR (4190209) | CPU | nf | ng | nh |
| LTS | 3.58 | 1.55 | 2.25 | 0.08 | LTS | 380.26 | 16.36 | 55.48 | 4.70 |
| LTS-O | 3.25 | 1.55 | 1.75 | 0.08 | LTS-O | 452.74 | 16.15 | 20.61 | 3.86 |
| S-PSB | 5.79 | 2.67 | 2.12 | 0.35 | S-PSB | 1405.36 | 262.67 | 134.00 | 3.23 |
| P-PSB | 5.20 | 2.48 | 1.91 | 0.25 | P-PSB | 651.50 | 80.32 | 39.02 | 3.61 |
| UP-BFGS | 35.82 | 13.49 | 7.77 | 3.18 | UP-BFGS | – | – | – | – |
| BP-BFGS | 189.15 | 11.00 | 6.62 | 2.60 | BP-BFGS | – | – | – | – |
| NCCS (7938) | CPU | nf | ng | nh | NCCO (32258) | CPU | nf | ng | nh |
| LTS | 0.29 | 11.52 | 60.82 | 1.34 | LTS | 0.05 | 1.34 | 1.37 | 0.00 |
| LTS-O | 0.13 | 11.52 | 26.88 | 1.34 | LTS-O | 0.04 | 1.34 | 1.37 | 0.00 |
| S-PSB | 0.14 | 6.28 | 4.12 | 3.00 | S-PSB | 0.16 | 2.57 | 2.03 | 0.75 |
| P-PSB | 0.24 | 10.33 | 5.94 | 5.54 | P-PSB | 0.40 | 3.58 | 2.49 | 1.39 |
| UP-BFGS | 0.04 | 2.15 | 1.88 | 0.61 | UP-BFGS | 0.08 | 1.58 | 1.47 | 0.15 |
| BP-BFGS | 51.26 | 309.39 | 166.18 | 85.42 | BP-BFGS | 0.09 | 1.58 | 1.47 | 0.15 |
| NCCS (32258) | CPU | nf | ng | nh | NCCO (130050) | CPU | nf | ng | nh |
| LTS | 0.80 | 8.47 | 61.00 | 1.34 | LTS | 11.51 | 9.33 | 289.34 | 7.00 |
| LTS-O | 0.40 | 8.47 | 24.52 | 1.34 | LTS-O | 0.83 | 3.33 | 16.34 | 1.00 |
| S-PSB | 0.19 | 2.57 | 2.03 | 0.75 | S-PSB | 2.57 | 22.64 | 2.51 | 0.19 |
| P-PSB | 0.35 | 3.58 | 2.49 | 1.39 | P-PSB | 2.69 | 22.90 | 2.62 | 0.35 |
| UP-BFGS | 0.08 | 1.58 | 1.47 | 0.15 | UP-BFGS | 3.88 | 22.40 | 2.37 | 0.04 |
| BP-BFGS | 411.80 | 415.97 | 226.73 | 138.35 | BP-BFGS | 3.88 | 22.40 | 2.37 | 0.04 |
| NCCS (130050) | CPU | nf | ng | nh | NCCO (522242) | CPU | nf | ng | nh |
| LTS | 11.37 | 12.02 | 264.70 | 6.33 | LTS | 12.03 | 3.33 | 73.34 | 1.75 |
| LTS-O | 1.53 | 7.02 | 23.58 | 1.33 | LTS-O | 1.38 | 1.83 | 5.09 | 0.25 |
| S-PSB | – | – | – | – | S-PSB | – | – | – | – |
| P-PSB | – | – | – | – | P-PSB | – | – | – | – |
| UP-BFGS | – | – | – | – | UP-BFGS | – | – | – | – |
| BP-BFGS | – | – | – | – | BP-BFGS | – | – | – | – |
| MOREBV (65025) | CPU | nf | ng | nh | MOREBV (1046529) | CPU | nf | ng | nh |
| LTS | 9.74 | 94.71 | 55.30 | 0.33 | LTS | 137.13 | 19.41 | 12.20 | 0.08 |
| LTS-O | 8.76 | 94.71 | 51.07 | 0.33 | LTS-O | 121.24 | 19.41 | 11.06 | 0.08 |
| S-PSB | 20.06 | 264.23 | 197.74 | 11.71 | S-PSB | 450.78 | 459.46 | 236.22 | 2.76 |
| P-PSB | 7.91 | 36.80 | 28.55 | 12.72 | P-PSB | 68.21 | 14.34 | 11.33 | 3.95 |
| UP-BFGS | 84.49 | 465.10 | 253.53 | 128.27 | UP-BFGS | 463.23 | 100.26 | 52.25 | 25.57 |
| BP-BFGS | 620.03 | 361.73 | 211.12 | 105.11 | BP-BFGS | 2468.63 | 94.53 | 50.04 | 24.37 |
| MOREBV (261121) | CPU | nf | ng | nh | | | | | |
| LTS | 29.62 | 43.44 | 24.25 | 0.08 | | | | | |
| LTS-O | 28.87 | 43.44 | 23.20 | 0.08 | | | | | |
| S-PSB | 46.17 | 145.85 | 101.13 | 6.03 | | | | | |
| P-PSB | 26.19 | 26.23 | 22.09 | 8.51 | | | | | |
| UP-BFGS | 271.05 | 362.46 | 187.63 | 93.71 | | | | | |
| BP-BFGS | 1802.63 | 342.01 | 180.34 | 89.62 | | | | | |