Computing useful sparse Hessian approximations satisfying componentwise secant equations I: using a known sparsity pattern

Working note RAL-NA-2013–1 — Nicholas I. M. Gould

29th April 2013

1 Introduction

Suppose that we are given a differentiable function $f(x)$ of n variables x, whose gradient $g(x) \stackrel{\text{def}}{=} \nabla_x f(x)$ is known. Our aim is to build estimates B_k of its Hessian matrix $H(x) \stackrel{\text{def}}{=} \nabla_{xx} f(x)$ at a sequence of given points x_k. Such a requirement lies at the heart of many Newton-like methods for minimizing f or methods that try to assess the stability of its gradient. We shall be particularly interested in the case for which

$\boxed{\textbf{A1}}$ $H(x)$ is sparse and its sparsity pattern is known;

the *sparsity pattern* of $H(x)$, $\mathcal{S}(H) \stackrel{\text{def}}{=} \{(i,j) : H_{ij}(x) = 0 \text{ for all } x\}$, and we shall say that the Hessian has an *entry* in row i and column j if $H_{ij}(x) \neq 0$ for some x.

If we are extremely fortunate, we might have an analytic expression for H or we might be able to obtain H by automatic differentiation [14]. If not, we might resort to a finite-difference approximation in which we obtain one column of B_k at a time from, for instance

$$B_k e_i \approx \delta_i^{-1}[g(x_k + \delta_i e_i) - g(x_k)],$$

where e_i is the i-th unit vector and δ_i is an appropriate small scalar [9]; more expensive but accurate central differences might also be used [6, §5.6]. Note that as it stands, $n+1$ gradient evaluations will be required to find B_k. If, however, $H(x)$ satisfies A1, it is often possible to partition $\mathcal{N} = \{1, \ldots, n\}$ so that $\mathcal{N} = \bigcup_{j=1}^m \mathcal{I}_j$, where $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ for $1 \leq i < j \leq m$, the rows indexed in each set $\mathcal{I}_j$ are orthogonal, and $m \ll n$. In this case

$$\sum_{i \in \mathcal{I}_j} B_k e_i \approx \delta_i^{-1}[g(x_k + \delta_i \sum_{i \in \mathcal{I}_j} e_i) - g(x_k)],$$

and only $m+1$ gradients are required; taking account of symmetry may reduce this count further [2, 18].

The other common way to approximate $H(x_k)$ is to use previous gradients $g(x_j)$, $j < k$ and to require that B_k satisfies the secant equation

$$B_k(x_k - x_{k-1}) = g_k - g_{k-1}.$$

where $g_k \stackrel{\rm def}{=} g(x_k)$. Traditionally, B_k is obtained from the previous estimate B_{k-1} by imposing the secant equation as well as the requirement that $B_k - B_{k-1}$ be of low rank [6, 17]—usually rank one or two—rather than sparse. Thus there is little reason to believe that B_k will be sparse even if its predecessor was. Although there have been attempts to derive sparse updates [7, 21, 22], there are good reasons to be concerned about their stability [20].

Such secant methods start from an initial Hessian estimate B_0 and build up the approximation as new points are added. Thus, after k steps, a rank k or $2k$ update will have been applied to B_0. A related limited-memory approach is to use dense low-rank updates, but rather than applying them all to B_0, apply the last m updates as if they had been applied to an re-initialized B_{k-m}. Indeed, rather than computing B_k from B_{k-1} directly, the sequence of previous m steps $s_j \stackrel{\rm def}{=} x_j - x_{j-1}$ and gradient differences $y_j \stackrel{\rm def}{=} g_j - g_{j-1}$ are recorded and the effect (product with, solve with) of using the relevant B_k computed when necessary [15, 16]. But as before, no attempt is made to enforce the structure $\mathcal{S}(H)$ on B_k.

At the other extreme, if f is partially separable [13], i.e., if

$$f(x) = \sum_{i=1}^{l} f_i(x),$$

where each "element" $f_i(x)$ has a large invariant subspace, it is then possible to obtain the approximation

$$B_k = \sum_{i=1}^{l} B_{ik}$$

where each element Hessian estimate B_{ik} satisfies its own secant equation

$$B_{ik}(x_k - x_{k-1}) = g_i(x_k) - g_i(x_{k-1})$$

and where $g_i(x) \stackrel{\rm def}{=} \nabla_x f_i(x)$. The invariant subspace assumption implies that g_i and B_{ik} are structured. In particular, any differentiable f with a sparse Hessian is partially separable [12], and in this case, the element secant equations each only involves a few variables, leading to an excellent sparse approximation. This and its generalization to group-partial separability [3], forms the basis of the approximations used in LANCELOT [4], but does not appear otherwise to have been widely adopted, primarily because users seem unable or unwilling to provide the necessary separability structure.

From the user's perspective, a more appealing approach, and the one we advocate in this paper, is to use the accumulated data $\{s_l\}_{l=k-m+1}^{k}$ and $\{y_l\}_{l=k-m+1}^{k}$ and the sparsity of H to estimate B_k directly. The only development we are aware of in this direction is that due to Fletcher, Grothey and Leyffer [8]. Their idea is to build an approximation B_k that satisfies as best as possible the multiple secant conditions

$$B_k s_l = y_l \;\text{ for }\; l = k-m+1, \ldots, k \tag{1.1}$$

the symmetry condition

$$B_k = B_k^T$$

and the sparsity condition

$$\mathcal{S}(B_k) = \mathcal{S}(H).$$

Since (1.1) may be inconsistent, a reasonable compromise is to solve instead the convex quadratic program

$$B_k = \arg\min_{B} \sum_{l=k-m+1}^{k} \|B_k s_l - y_l\|_2^2 \text{ subject to } B = B^T \text{ and } \mathcal{S}(B) = \mathcal{S}(H).$$

This solution may be found by solving a linear system of order, n_e, the number of entries in the upper triangle of $H(x)$. Since n_e may in general be large, estimates of the solution B_k may better be found using an iterative scheme such as conjugate gradients [8].

2 Estimation

The approach we advocate is somewhat different. Rather than imposing the full secant conditions (1.1) for the previous m steps, we aim to satisfy as many *componentwise* equations

$$e_i^T B_k s_l = e_i^T y_l \tag{2.1}$$

or equivalently

$$\sum_{j \in \mathcal{I}_i} (B_k)_{ij} s_{jl} = y_{il}, \text{ where } \mathcal{I}_i = \{j : H_{ij} \neq 0\} \tag{2.2}$$

for $l = k, k-1, \ldots$ as are necessary to define the ith row of B_k for each $1 \leq i \leq n$. Naively (and neglecting any inconsistencies or redundancies in dependencies), for row i, we need as many equations (2.2) as there are entries in row i, and the entries may be calculated in any order (and in parallel) as follows:

Algorithm 2.1: Sparse Hessian approximation (unsymmetric version)

For $i = 1, \ldots, n$:
Compute the unknown entries in the row using (2.2) for $l = k - |\mathcal{I}_i| + 1, \ldots, k$.

Note that any off diagonal entry will be estimated twice in Algorithm 2.1; if we are relying on a symmetric approximation, we may replace both by a suitable weighted average. Alternatively, we may simply take the off-diagonal values in the order they are calculated, overwriting the first estimate by the second; in practice in this case we access the rows in order of decreasing row counts, since rows with smaller counts require fewer differences and may therefore be more accurate.

But of course this does not truly account for symmetry. In particular (2.2) may be rewritten as

$$\sum_{j \in \mathcal{I}_i^-} (B_k)_{ij} s_{jl} = y_{il} - \sum_{j \in \mathcal{I}_i^+} (B_k)_{ij} s_{jl} \tag{2.3}$$

where

$$\mathcal{I}_i^+ = \{j : j \in \mathcal{I}_i \text{ and } (B_k)_{ji} \text{ is already known}\} \;\text{ and }\; \mathcal{I}_i^- = \mathcal{I}_i \setminus \mathcal{I}_i^+.$$

Thus for row i (and again ignoring inconsistencies and the like), the data from $|\mathcal{I}_i^+|$ previous steps is required.

It is clear from the above that the order in which the rows are addressed is crucial. For example, consider two Hessian matrices with arrow-head structures

$$\begin{pmatrix} * & & & & * \\ & * & & & * \\ & & \cdot & & \vdots \\ & & & * & * \\ * & * & \cdots & * & * \end{pmatrix} \text{ and } \begin{pmatrix} * & * & \cdots & * & * \\ * & * & & & \\ \vdots & & \cdot & & \\ * & & & * & \\ * & & & & * \end{pmatrix}$$

For the former, the first $n-1$ rows each require that two entries—those on the diagonal and those in column n—be computed. The last row requires n entries, but by symmetry, all of the off-diagonal entries have already been computed. Thus in this row, only the diagonal entry is unknown, and hence all of the data may be computed using two data pairs (s_k, y_k) and (s_{k-1}, y_{k-1}). Contrast this with the second example, where the first row contains n (at this stage) unknown entries that must be computed, and thus n data pairs $(s_l, y_l), l = k-n+1, \ldots k$ will be required. Of course the two examples are structurally symmetric permutations of one another.

Our algorithm is based on the connectivity graph $\mathcal{C}$ of $H(x)$. Recall, this is a graph with n vertices for which there is an edge between vertices i and j if and only if $H_{ij}(x) \neq 0$ for some x. The number of entries in row i is the degree of vertex i plus one. Our ordering and approximation strategy is as follows:

Algorithm 2.2: Sparse Hessian approximation (symmetric version)

Compute the connectivity graph $\mathcal{C}$ of $H(x)$, and record the degrees of each vertex.
For $i = 1, \ldots, n$:

  Find a vertex of minimum degree, and assign the corresponding row as row i.
  Compute the remaining unknown entries in the row using (2.3) for $l = k - |\mathcal{I}_i^+| + 1, \ldots, k$.
  Remove the chosen vertex, and update the remaining degrees.

The aim of the algorithm is clearly to keep the number of required data pairs per step small.

At each step, a vertex of smallest degree is required. The vertices can be initially ordered using a counting [19, §2.4.6] or bucket [5] sort in $O(n)$ operations and storage locations.

Thereafter, since at each stage each degree can decrease by at most one, the degrees and their order may be updated in $O(n_e)$ operations. The cost of finding the unknown entries in the ith ordered row is $O(|\mathcal{I}_i^+|^3)$ floating-point operations using Gaussian elimination.

While Algorithm 2.2 will almost always require fewer floating-point operations than its predecessor, it has two, related, disadvantages. The first is that the steps in Algorithm 2.1 may be performed in parallel (aside from any value averaging for symmetry), while Algorithm 2.2 is to a large degree sequential—in practice, vertices i and j of minimum degree with $\mathcal{I}_i^+ \cap \mathcal{I}_j^+ = \emptyset$ may be processed in parallel. The second, and more serious, defect is that inaccurate estimates from earlier steps in Algorithm 2.2 can be magnified when solving (2.3), leading to error growth even for constant Hessians ($H(x) = H$); this is similar in effect to that when attempting Gaussian elimination without pivoting. Algorithm 2.1, by contrast, is immune since each row's values are computed independently. Observed error growth is usually gradual but relentless; the more times an inaccurate early value occurs in later rows, the worse the effect, and this is particularly pernicious for large matrices.

An obvious way around this predicament is to combine the two (unsymmetric and symmetric) approaches. To do so, we presume that the vast majority of the rows of the Hessian we seek to estimate are very sparse, while the remaining few are relatively dense. Thus, we may estimate the entries in the sparse rows independently and (presumably) accurately leaving a few to estimate using the symmetric scheme. That is, after a implicit symmetric permutation, we estimate the Hessian

$$\begin{pmatrix} H_{11} & H_{12} \\ H_{12}^T & H_{22} \end{pmatrix} \tag{2.4}$$

by obtaining the entries $(H_{11} \quad H_{12})$ independently row-by-row, and then finding the unknowns H_{22} in the few remaining rows by applying Algorithm 2.2 to H_{22}. Formally,

Algorithm 2.3: Sparse Hessian approximation (combined version)

For sparse rows $i = 1, \ldots, n$:

Compute the unknown entries in the row using (2.2) for $l = k - |\mathcal{I}_i| + 1, \ldots, k$.

Compute the connectivity graph $\mathcal{C}$ of the remaining $H_{22}(x)$, and record the degrees of each vertex.

For remaining dense rows $i = 1, \ldots, n$:

Find a vertex of minimum degree, and assign the corresponding row as row i.

Compute the remaining unknown entries in the row using (2.3) for $l = k - |\mathcal{I}_i^+| + 1, \ldots, k$.

Remove the chosen vertex, and update the remaining degrees.

In principal it is possible to apply Algorithm 2.3 recursively to the block H_{22} in (2.4), but this may also lead to large error growth if the recursion is too deep. In our experience, fortunately, recursion is unnecessary (see §4).

An alternative is to order the vertices using Algorithm 2.2, but only to use (2.3) to compute the unknown entries in row i if $|\mathcal{I}_i|$ is deemed too large to use (2.2) to compute all the entries in the row afresh. We shall refer to this approach as Algorithm 2.2/3 (and use 50 entries as our default switch threshold).

Advantages and disadvantages c.f. [8] -

- More susceptible to bad early values
- possibly few entire secant equations
- much cheaper to obtain

3 Enhancements

The ith componentwise secant equation (2.1) may be written compactly as

$$S_i^T b_i = y_i^{\mathrm{R}}, \tag{3.1}$$

where b_i are the $|\mathcal{I}_i|$ entries in the ith row of B, S_i is the matrix made up of the components of the rows indexed by $\mathcal{I}_i$ of the data vectors $\{s_j\}$, $j = k, k-1, ...,$ and y_i^{R} is the (transpose of the) ith row of the matrix whose columns are the vectors $\{y_j\}$, $j = k, k-1, ...$ The corresponding equation (2.3) may be expressed similarly as

$$(S_i^-)^T b_i^- = y_i^{\mathrm{R}} - (S_i^+)^T b_i^+, \tag{3.2}$$

where the $^+$ and $^-$ indicate the known and unknown components of b_i and the corresponding sub-matrices of S_i. Thus generically, for each row we need to satisfy equations

$$S^T b = y, \tag{3.3}$$

to determine the ℓ components of b as best as possible. Notice that we have not yet specified how many (or which) data vectors $\{(s_j, y_j)\}$, $j = k, \ldots, 1$ are required.

Ideally we would have sufficient data that $k \geq l$ and the square matrix S made up from the ℓ most recent s_j is non singular. But clearly this may not be the case. Firstly, in the early stages of estimation, there may simply not be enough data; this will certainly be the case if $k < l$. Secondly, S formed as above may be singular (or close to singular) and in this case either again there will not be enough data to determine b uniquely or the data y may be inconsistent (if f is not quadratic). One may of course choose to substitute existing inconsistent data with that obtained from earlier iterations.

When there is insufficient data, one option is simply to assign certain components of b to zero, and solve for the remainder. One might, for example, remove elements far from the diagonal. However, since this seems relatively arbitrary, our preferred strategy is instead to find the smallest b consistent with the data by solving the constrained least-squares problem

$$\underset{b\in\mathbb{R}^l}{\text{minimize}} \;\; \|b\|_2 \;\text{ subject to }\; S^Tb = y$$

using, for example, a singular-value decomposition of S. When the latest data is inconsistent, rather than trying to find earlier data to exchange, one might instead put additional earlier data into B and y and to solve the weighted least-squares problem

$$\underset{b\in\mathbb{R}^k}{\text{minimize}} \;\; \|W(S^Tb - y)\|_2$$

where the diagonal weights W favour the latest data. Once again, a singular-value decomposition of WS^T is suitable. Our preference is however simply to find the least-squares solution to $S^Tb = y$ of minimum ℓ_2-norm from the singular-value decomposition of S^T.

To be specific, in both the under- and over-determined cases, we compute the "compact" singular-value decomposition $S^T = U\Sigma V^T \in \mathbb{R}^{m_s\times n_s}$, where the columns of $U \in \mathbb{R}^{m_s\times r_s}$ and $V \in \mathbb{R}^{n_s\times r_s}$ are orthogonal, $\Sigma \in \mathbb{R}^{r_s\times r_s}$ is non-singular and diagonal, and r_s is the rank of S^T. We then find the required solution $b = V\Sigma^{-1}U^Ty$ using, for example, `gelss` or `gelsd` from LAPACK [1]. Optionally we also provide a faster but potentially less stable variant based on a QR factorization of S^T with interchanges using LAPACK's `gelsy`, as well as a faster-still LU-based approach using `getrf/s` when S is square and non-singular.

4 Numerical experiments

We now consider how the methods we have suggested perform in practice. To do so, we consider Hessian matrices for all of the sparse examples from the CUTEr [10] collection whose dimensions either exceed 999 or are variable (in which case the default dimension is used). This leads to 447 problems in total, 127 of which are unconstrained.

Our experiments are performed on a single processor of a system comprising 16 Intel Xeon E5620 CPUs clocked at 2.4GHz, with 23.5 GiB of RAM, running the 64-bit Ubuntu 12.04.2 LTS (precise) operating system. The algorithms from § 2 have been implemented in the fortran 2003 package `SHA` as part of the GALAHAD library [11]. All codes are compiled in double precision using gfortran 4.6 with -O3 optimization.

In our first set of experiments, we simply wish to investigate the accuracy attained by the algorithms we have proposed under ideal circumstances. In particular, we evaluate the Hessian $H(x)$ of each of our problems at a "typical" value[1], we generate pseudo random vectors s_i, $i = 1, \ldots, m$, with entries in $[-1, 1]$, and we evaluate $y_i = H(x)s_i$. We then use our algorithms to see how well we can reproduce $H(x)$ from the given data pairs (s_i, y_i), $i = 1, \ldots, m$; at most $m = 100$ pairs are permitted. The full results of our experiments are given in Table A.1 in Appendix A.

[1] for constrained problems, we evaluate the Hessian of the Lagrangian function with "typical" Lagrange multipliers.

Close scrutiny of Table A.1 reveals the following trends. When the maximum number of entries per row is reasonable, Algorithm 2.1 finds their values fast and accurately. However, there are a significant number of examples in the test set that have one or more relatively dense rows, and for these the unsophisticated approach is inappropriate. Algorithm 2.2 generally produces unknown entry counts that are better than those for its predecessor, often markedly so, and as a consequence often runs faster. In some cases, the accuracy is good, but in many cases the error growth through using existing estimates when computing new ones is large, often catastrophically so. Thus, we really cannot recommend this approach. The combined versions fare better. Algorithm 2.2/3 generally limits the growth very well, only in a few cases are rows consistently too full to prohibit independent evaluation, and for these there is rarely fatal growth; large growth occured for `JIMACK`, `MSQRTA`, `MSQRTB`, `ORTHREGE`, `SCURLY30`, `SPARSINE`, `SPARSQUR`, `STC/NQP1/2` and `TWIRIMD1`, while there were insufficient data pairs to evaluate `FERRISDC` and `TWIRIBG1`. Algorithm 2.3 is even more successful. Only `ORTHREGE`, exhibited significant growth, although again `FERRISDC`, `TWIRIBG1` and additionally `TWIRIMD1` needed more data than was available. In most cases, the actual number of data pairs required is extremely modest, and this leads us to believe that Algorithm 2.3 has high potential for sparse estimation.

5 Conclusions

We have presented a number of new methods for computing Hessian approximations when the sparisty structure is known in advance. The methods are promising in many cases, but numerical issues with stability of the approximated Hessian entries gives some cause for concern. We view this as similar to those that may happen in Gaussian Elimination when solving linear systems, and anticipate that more complicated pivoting strategies will be needed to overcome this defficiency. This is the subject of ongoing research.

Acknowledgment

I am very grateful to Coralia Cartis, Jared Tanner and Philippe Toint for stimulating discussions on the work presented here.

Appendix A

In Table A.1, we report the complete results when applying Algorithms 2.1, 2.2 and the two variants of 2.3 on the sparse examples from the CUTEr [10] test set. For each problem, we categorise its type (ty) as Unconstrained or Constrained, its dimension n, and the maximum number of entries in any row of its Hessian (deg). For each algorithm we record the maximum dimension of any linear system that needs to be solved (sys), $\log_{10}$ of the relative componentwise error

$$\max_{(i,j)\in\mathcal{S}(H)} \frac{|H^{\text{EST}}_{ij} - H^{\text{EXACT}}_{ij}|}{\max(1, |H^{\text{EXACT}}_{ij}|)}$$

of the computed approximation H^{EST} of the exact Hessian H^{EXACT} (err, with -inf meaning the error is zero and inf indicating overflow) and the total CPU time taken to compute the approximation in seconds. We allowed a maximum linear-system dimension of 100, and any algorithm that required more than this is classified as a failure (indicated by a -).

Table A.1: Reproducing Hessians for quadratic examples: Complete results

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
A0ENDNDL	C	45006	3	3	-11	0.044	2	-14	0.041	3	-11	0.043	3	-11	0.045
A0ENINDL	C	45006	3	3	-11	0.045	2	-14	0.039	3	-11	0.042	3	-11	0.044
A0ENSNDL	C	45006	3	3	-11	0.046	2	-14	0.040	3	-11	0.045	3	-11	0.044
A0ESDNDL	C	45006	3	3	-11	0.047	2	-14	0.040	3	-11	0.047	3	-11	0.046
A0ESINDL	C	45006	3	3	-11	0.047	2	-14	0.040	3	-11	0.045	3	-11	0.043
A0ESSNDL	C	45006	3	3	-11	0.044	2	-14	0.041	3	-11	0.045	3	-11	0.041
A0NNDNDL	C	60012	3	3	-10	0.059	2	-13	0.054	3	-10	0.058	3	-11	0.055
A0NNDNIL	C	60012	3	3	-10	0.060	2	-13	0.052	3	-10	0.058	3	-11	0.055
A0NNDNSL	C	60012	3	3	-10	0.067	2	-13	0.053	3	-10	0.058	3	-11	0.055
A0NNSNSL	C	60012	3	3	-10	0.058	2	-13	0.052	3	-10	0.057	3	-11	0.055
A0NSDSDL	C	60012	3	3	-10	0.059	2	-13	0.057	3	-10	0.059	3	-11	0.056
A0NSDSDS	C	6012	3	3	-13	0.007	2	-14	0.007	3	-13	0.006	3	-13	0.007
A0NSDSIL	C	60012	3	3	-10	0.058	2	-13	0.053	3	-10	0.057	3	-11	0.057
A0NSDSSL	C	60012	3	3	-10	0.061	2	-13	0.054	3	-10	0.057	3	-11	0.058
A0NSSSSL	C	60012	3	3	-10	0.059	2	-13	0.056	3	-10	0.058	3	-11	0.055
A2ENDNDL	C	45006	3	3	-11	0.045	2	-14	0.039	3	-11	0.043	3	-11	0.042
A2ENINDL	C	45006	3	3	-11	0.044	2	-14	0.043	3	-11	0.045	3	-11	0.042
A2ENSNDL	C	45006	3	3	-11	0.047	2	-14	0.042	3	-11	0.043	3	-11	0.045
A2ESDNDL	C	45006	3	3	-11	0.044	2	-14	0.041	3	-11	0.043	3	-11	0.043
A2ESINDL	C	45006	3	3	-11	0.046	2	-14	0.040	3	-11	0.045	3	-11	0.043
A2ESSNDL	C	45006	3	3	-11	0.047	2	-14	0.039	3	-11	0.043	3	-11	0.042
A2NNDNDL	C	60012	3	3	-10	0.074	2	-13	0.052	3	-10	0.058	3	-11	0.057
A2NNDNIL	C	60012	3	3	-10	0.059	2	-13	0.054	3	-10	0.059	3	-11	0.057
A2NNDNSL	C	60012	3	3	-10	0.059	2	-13	0.053	3	-10	0.061	3	-11	0.056
A2NNSNSL	C	60012	3	3	-10	0.067	2	-13	0.054	3	-10	0.060	3	-11	0.057
A2NSDSDL	C	60012	3	3	-10	0.059	2	-13	0.055	3	-10	0.059	3	-11	0.055
A2NSDSIL	C	60012	3	3	-10	0.061	2	-13	0.056	3	-10	0.061	3	-11	0.055
A2NSDSSL	C	60012	3	3	-10	0.062	2	-13	0.055	3	-10	0.058	3	-11	0.055
A2NSSSSL	C	60012	3	3	-10	0.059	2	-13	0.073	3	-10	0.058	3	-11	0.057
A5ENDNDL	C	45006	3	3	-11	0.048	2	-14	0.043	3	-11	0.048	3	-11	0.042
A5ENINDL	C	45006	3	3	-11	0.045	2	-14	0.042	3	-11	0.043	3	-11	0.043
A5ENSNDL	C	45006	3	3	-11	0.049	2	-14	0.040	3	-11	0.045	3	-11	0.042
A5ESDNDL	C	45006	3	3	-11	0.044	2	-14	0.039	3	-11	0.044	3	-11	0.041
A5ESINDL	C	45006	3	3	-11	0.044	2	-14	0.042	3	-11	0.043	3	-11	0.046
A5ESSNDL	C	45006	3	3	-11	0.044	2	-14	0.039	3	-11	0.043	3	-11	0.044
A5NNDNDL	C	60012	3	3	-10	0.065	2	-13	0.053	3	-10	0.058	3	-11	0.055
A5NNDNIL	C	60012	3	3	-10	0.059	2	-13	0.052	3	-10	0.059	3	-11	0.060
A5NNDNSL	C	60012	3	3	-10	0.059	2	-13	0.053	3	-10	0.058	3	-11	0.060
A5NNSNSL	C	60012	3	3	-10	0.058	2	-13	0.057	3	-10	0.058	3	-11	0.056
A5NSDSDL	C	60012	3	3	-10	0.058	2	-13	0.053	3	-10	0.064	3	-11	0.057
A5NSDSDM	C	6012	3	3	-13	0.006	2	-14	0.006	3	-13	0.007	3	-13	0.006
A5NSDSIL	C	60012	3	3	-10	0.059	2	-13	0.053	3	-10	0.057	3	-11	0.055
A5NSDSSL	C	60012	3	3	-10	0.059	2	-13	0.053	3	-10	0.058	3	-11	0.056
A5NSSNSM	C	6012	3	3	-13	0.007	2	-14	0.005	3	-13	0.006	3	-13	0.007
A5NSSSSL	C	60012	3	3	-10	0.060	2	-13	0.055	3	-10	0.058	3	-11	0.055
ALJAZZAF	C	1000	1	1	-16	0.001	1	-16	0.002	1	-16	0.001	1	-16	0.001
ALLINQP	C	50000	3	3	-12	0.056	2	-8	0.049	3	-12	0.056	3	-11	0.057
ARGTRIG	C	200	1	1	-15	0.001	1	-15	0.001	1	-15	0.002	1	-15	0.001
ARTIF	C	5002	1	1	-16	0.004	1	-16	0.004	1	-16	0.004	1	-16	0.004
ARWHDNE	C	500	1	1	-16	0.001	1	-16	0.002	1	-16	0.001	1	-16	0.002
ARWHEAD	U	5000	5000	5000	-	-	2	-13	0.009	2	-13	0.009	2	-13	0.009
AUG2D	C	20200	1	1	-inf	0.015	1	-inf	0.015	1	-inf	0.015	1	-inf	0.015
AUG2DC	C	20200	1	1	-inf	0.014	1	-inf	0.015	1	-inf	0.015	1	-inf	0.014
AUG2DCQP	C	20200	1	1	-inf	0.016	1	-inf	0.013	1	-inf	0.014	1	-inf	0.014
AUG2DQP	C	20200	1	1	-inf	0.014	1	-inf	0.014	1	-inf	0.015	1	-inf	0.013
AUG3D	C	27543	1	1	-inf	0.016	1	-inf	0.016	1	-inf	0.018	1	-inf	0.017
AUG3DC	C	27543	1	1	-inf	0.020	1	-inf	0.019	1	-inf	0.020	1	-inf	0.019
AUG3DCQP	C	27543	1	1	-inf	0.019	1	-inf	0.019	1	-inf	0.020	1	-inf	0.020
AUG3DQP	C	27543	1	1	-inf	0.016	1	-inf	0.016	1	-inf	0.016	1	-inf	0.016
BDEXP	U	5000	5	5	-12	0.009	3	-11	0.007	5	-12	0.009	5	-13	0.009
BDQRTIC	U	5000	5000	5000	-	-	5	-9	0.014	8	-12	0.018	8	-12	0.017
BDVALUE	C	5002	1	1	-22	0.004	1	-22	0.003	1	-22	0.004	1	-22	0.004
BDVALUES	C	10002	1	1	-21	0.008	1	-21	0.006	1	-21	0.008	1	-21	0.006
BIGBANK	C	2230	1	1	-16	0.002	1	-16	0.002	1	-16	0.002	1	-16	0.002
BIGGSB1	U	5000	3	3	-13	0.006	2	-10	0.005	3	-13	0.006	3	-12	0.005
BLOCKQP1	C	10010	2	2	-13	0.010	2	-inf	0.010	2	-13	0.008	2	-13	0.010
BLOCKQP2	C	10010	2	2	-13	0.009	2	-inf	0.008	2	-13	0.010	2	-13	0.009
BLOCKQP3	C	10010	2	2	-13	0.010	2	-13	0.008	2	-13	0.010	2	-13	0.010
BLOCKQP4	C	10010	2	2	-13	0.009	2	-13	0.009	2	-13	0.013	2	-13	0.010
BLOCKQP5	C	10010	2	2	-13	0.010	2	-13	0.009	2	-13	0.009	2	-13	0.010
BLOWEYA	C	4002	4	4	-13	0.006	2	-10	0.004	4	-13	0.004	4	-13	0.004
BLOWEYB	C	4002	4	4	-13	0.004	2	-10	0.003	4	-13	0.005	4	-13	0.004
BLOWEYC	C	4002	4	4	-13	0.005	2	-10	0.004	4	-13	0.005	4	-13	0.004
BQPGAUSS	U	2003	552	552	-	-	14	06	0.004	40	-10	0.009	40	-11	0.009
BRAINPC0	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.009
BRAINPC1	C	6907	3452	3452	-	-	4	-11	0.010	5	-11	0.010	5	-11	0.010
BRAINPC2	C	13807	6902	6902	-	-	4	-9	0.021	5	-9	0.020	5	-8	0.020
BRAINPC3	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.010
BRAINPC4	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.011
BRAINPC5	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.011	5	-8	0.010
BRAINPC6	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.010
BRAINPC7	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.010

Table A.1: Reproducing Hessians for quadratic examples: Complete results (continued)

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
BRAINPC8	C	6907	3452	3452	-	-	4	-8	0.011	5	-8	0.010	5	-8	0.010
BRAINPC9	C	6907	3452	3452	-	-	4	-8	0.010	5	-8	0.010	5	-8	0.010
BRATU1D	U	5003	3	3	-12	0.006	2	-11	0.005	3	-12	0.006	3	-12	0.005
BRATU2D	C	5184	1	1	-19	0.005	1	-19	0.004	1	-19	0.003	1	-19	0.004
BRATU2DT	C	5184	1	1	-19	0.005	1	-19	0.004	1	-19	0.004	1	-19	0.004
BRATU3D	C	4913	1	1	-17	0.003	1	-17	0.003	1	-17	0.003	1	-17	0.003
BRIDGEND	C	2734	1	1	-20	0.002	1	-20	0.002	1	-20	0.003	1	-20	0.004
BROWNALE	C	200	10	10	-15	0.001	10	-14	0.000	10	-15	0.002	10	-15	0.000
BROYDN3D	C	5000	1	1	-16	0.006	1	-16	0.003	1	-16	0.005	1	-16	0.004
BROYDN7D	U	5000	6	6	-11	0.010	4	231	0.007	6	-11	0.011	6	-11	0.010
BROYDNBD	C	5000	1	1	-15	0.004	1	-15	0.004	1	-15	0.004	1	-15	0.004
BRYBND	U	5000	13	13	-10	0.028	7	-8	0.012	13	-10	0.028	13	-10	0.027
CAMSHAPE	C	800	5	5	-13	0.002	3	-11	0.001	5	-13	0.001	5	-13	0.002
CAR2	C	5999	5999	5999	-	-	11	235	0.026	17	-13	0.044	17	-12	0.043
CATENA	C	3003	3	3	-13	0.005	2	-11	0.004	3	-13	0.003	3	-13	0.005
CATENARY	C	3003	4	4	-13	0.003	2	-11	0.003	4	-13	0.004	4	-13	0.004
CATMIX	C	2403	5	5	-15	0.003	3	-12	0.003	5	-15	0.004	5	-15	0.004
CBRATU2D	C	3200	2	2	-15	0.004	2	-15	0.003	2	-15	0.004	2	-15	0.003
CBRATU3D	C	3456	2	2	-15	0.002	2	-14	0.002	2	-15	0.002	2	-15	0.003
CHAIN	C	802	2	2	-17	0.001	2	-16	0.001	2	-17	0.001	2	-17	0.001
CHAINWOO	U	4000	4	4	-11	0.004	2	-9	0.005	4	-11	0.004	4	-11	0.004
CHANNEL	C	9600	2	2	-12	0.006	2	-12	0.007	2	-12	0.009	2	-12	0.006
CHEMRCTA	C	5000	2	2	-9	0.004	2	-7	0.006	2	-9	0.004	2	-9	0.006
CHEMRCTB	C	5000	1	1	-16	0.004	1	-16	0.004	1	-16	0.005	1	-16	0.004
CHENHARK	U	5000	5	5	-12	0.009	3	-10	0.006	5	-12	0.009	5	-12	0.009
CHNROSNB	U	50	3	3	-14	0.001	2	-13	0.001	3	-14	0.001	3	-15	0.001
CHNRSBNE	C	50	1	1	-16	0.000	1	-16	0.001	1	-16	0.001	1	-16	0.001
CLNLBEAM	C	6003	1	1	-17	0.003	1	-17	0.004	1	-17	0.003	1	-17	0.003
CLPLATEA	U	5041	5	5	-10	0.009	3	19	0.007	5	-10	0.010	5	-9	0.007
CLPLATEB	U	5041	5	5	-10	0.009	3	19	0.007	5	-10	0.009	5	-9	0.009
CLPLATEC	U	5041	5	5	-9	0.009	3	19	0.006	5	-9	0.009	5	-9	0.010
CONT5-QP	C	40601	1	1	-19	0.002	1	-19	0.003	1	-19	0.003	1	-19	0.003
CONT6-QQ	C	20002	2	2	-17	0.018	2	-16	0.018	2	-17	0.018	2	-17	0.017
CORKSCRW	C	4506	3	3	-13	0.002	2	-13	0.002	3	-13	0.002	3	-13	0.003
COSHFUN	C	6001	2	2	-13	0.005	2	-13	0.005	2	-13	0.006	2	-13	0.006
COSINE	U	10000	3	3	-11	0.011	2	-9	0.009	3	-11	0.011	3	-11	0.011
CRAGGLVY	U	5000	3	3	-8	0.007	2	-6	0.006	3	-8	0.006	3	-8	0.006
CURLY10	U	10000	21	21	-11	0.124	11	-8	0.047	21	-11	0.122	21	-11	0.118
CURLY20	U	10000	41	41	-10	0.469	21	-8	0.131	41	-10	0.468	41	-10	0.461
CURLY30	U	10000	61	61	-9	1.207	31	-6	0.275	50	-6	0.276	61	-9	1.198
CVXBQP1	U	10000	9	9	-11	0.029	7	48	0.017	9	-11	0.028	9	-11	0.026
CVXQP1	C	10000	9	9	-11	0.027	7	48	0.018	9	-11	0.027	9	-11	0.027
CVXQP2	C	10000	9	9	-11	0.027	7	48	0.018	9	-11	0.028	9	-11	0.025
CVXQP3	C	10000	9	9	-11	0.027	7	48	0.017	9	-11	0.027	9	-11	0.026
DALLASL	C	906	1	1	-16	0.001	1	-16	0.001	1	-16	0.001	1	-16	0.001
DEGENQP	C	50	1	1	-16	0.002	1	-16	0.001	1	-16	0.000	1	-16	0.001
DITTERT	C	1133	127	127	-	-	10	-9	0.003	37	-12	0.004	37	-12	0.004
DIXCHLNV	C	1000	5	5	-7	0.002	3	-5	0.002	5	-7	0.003	5	-7	0.003
DIXMAANA	U	3000	5	5	-11	0.005	4	18	0.003	5	-11	0.006	5	-11	0.006
DIXMAANB	U	3000	5	5	-11	0.006	4	19	0.005	5	-11	0.006	5	-11	0.006
DIXMAANC	U	3000	5	5	-11	0.005	4	18	0.004	5	-11	0.006	5	-11	0.006
DIXMAAND	U	3000	5	5	-11	0.007	4	19	0.005	5	-11	0.005	5	-11	0.006
DIXMAANE	U	3000	5	5	-11	0.007	4	19	0.005	5	-11	0.006	5	-12	0.006
DIXMAANF	U	3000	5	5	-11	0.005	4	17	0.004	5	-11	0.006	5	-11	0.005
DIXMAANG	U	3000	5	5	-11	0.006	4	18	0.004	5	-11	0.005	5	-11	0.006
DIXMAANH	U	3000	5	5	-11	0.006	4	19	0.005	5	-11	0.005	5	-11	0.005
DIXMAANI	U	3000	5	5	-11	0.005	4	19	0.004	5	-11	0.006	5	-11	0.005
DIXMAANJ	U	3000	5	5	-11	0.006	4	18	0.004	5	-11	0.005	5	-11	0.006
DIXMAANK	U	3000	5	5	-11	0.006	5	-11	0.004	5	-11	0.005	5	-11	0.006
DIXMAANL	U	3000	5	5	-11	0.007	4	19	0.004	5	-11	0.005	5	-11	0.006
DIXON3DQ	U	10000	3	3	-12	0.012	2	-9	0.010	3	-12	0.013	3	-12	0.012
DQDRTIC	U	5000	1	1	-16	0.004	1	-16	0.004	1	-16	0.004	1	-16	0.004
DQRTIC	U	5000	1	1	-16	0.005	1	-16	0.005	1	-16	0.005	1	-16	0.003
DRCAV1LQ	U	4489	41	41	-8	0.197	21	115	0.059	41	-8	0.194	41	-7	0.189
DRCAV2LQ	U	4489	41	41	-8	0.196	21	115	0.058	41	-8	0.192	41	-7	0.189
DRCAV3LQ	U	4489	41	41	-8	0.196	21	115	0.060	41	-8	0.195	41	-7	0.190
DRCAVTY1	C	4489	41	41	-9	0.195	21	111	0.058	41	-9	0.201	41	-9	0.187
DRCAVTY2	C	4489	41	41	-9	0.193	21	111	0.059	41	-9	0.193	41	-9	0.186
DRCAVTY3	C	4489	41	41	-8	0.194	21	113	0.058	41	-8	0.194	41	-9	0.185
DRUGDIS	C	6004	6004	6004	-	-	4	-14	0.012	4	-14	0.013	4	-14	0.013
DRUGDISE	C	603	601	601	-	-	5	-7	0.003	7	-10	0.002	7	-8	0.002
DTOC1L	C	5998	1	1	-16	0.005	1	-16	0.005	1	-16	0.005	1	-16	0.005
DTOC1NA	C	5998	5	5	-12	0.008	3	-12	0.007	5	-12	0.009	5	-12	0.007
DTOC1NB	C	5998	5	5	-11	0.009	3	-11	0.007	5	-11	0.008	5	-12	0.008
DTOC1NC	C	5998	5	5	-11	0.008	3	-11	0.008	5	-11	0.008	5	-12	0.008
DTOC1ND	C	5998	5	5	-11	0.008	3	-11	0.006	5	-11	0.009	5	-12	0.009
DTOC2	C	5998	6	6	-12	0.012	6	-11	0.009	6	-12	0.012	6	-11	0.010
DTOC3	C	4499	1	1	-19	0.004	1	-19	0.003	1	-19	0.003	1	-19	0.003
DTOC4	C	4499	2	2	-15	0.004	2	-14	0.005	2	-15	0.004	2	-15	0.005
DTOC5	C	9999	1	1	-19	0.006	1	-19	0.008	1	-19	0.008	1	-19	0.007
DTOC6	C	10001	2	2	-13	0.008	2	-11	0.010	2	-13	0.009	2	-13	0.009
EDENSCH	U	2000	3	3	-13	0.003	2	-10	0.003	3	-13	0.002	3	-13	0.003

Table A.1: Reproducing Hessians for quadratic examples: Complete results (continued)

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
EG2	U	1000	999	999	-	-	2	-13	0.003	2	-13	0.002	2	-13	0.003
EG3	C	10001	10001	10001	-	-	4	-12	0.020	4	-12	0.019	4	-12	0.019
EIGENA	C	2550	51	51	-12	0.214	51	-9	0.079	51	-12	0.210	51	-12	0.210
EIGENA2	C	2550	51	51	-10	0.213	51	-8	0.077	51	-10	0.213	51	-10	0.207
EIGENAU	C	2550	51	51	-12	0.213	51	-9	0.079	51	-12	0.210	51	-12	0.209
EIGENB	C	2550	51	51	-12	0.213	51	-9	0.078	51	-12	0.210	51	-12	0.207
EIGENB2	C	2550	51	51	-12	0.210	51	-10	0.077	51	-12	0.212	51	-12	0.206
EIGENC	C	2652	52	52	-11	0.214	52	-9	0.082	52	-11	0.214	52	-11	0.213
EIGENC2	C	2652	52	52	-9	0.215	52	-8	0.082	52	-9	0.215	52	-9	0.210
ENGVAL1	U	5000	3	3	-12	0.006	2	-10	0.006	3	-12	0.007	3	-12	0.007
ERRINROS	U	50	3	3	-13	0.001	2	-12	0.000	3	-13	0.000	3	-13	0.001
EXPLIN	U	1200	3	3	-16	0.001	2	-15	0.000	3	-16	0.001	3	-16	0.001
EXPLIN2	U	1200	3	3	-15	0.001	2	-15	0.001	3	-15	0.000	3	-16	0.000
EXPQUAD	U	1200	1100	1100	-	-	2	-9	0.003	3	-12	0.002	3	-12	0.001
EXTROSNB	U	1000	3	3	-13	0.003	2	-11	0.001	3	-13	0.001	3	-13	0.002
FERRISDC	C	2200	200	200	-	-	200	-	-	200	-	-	200	-	-
FLETCBV2	U	5000	3	3	-12	0.006	2	-10	0.006	3	-12	0.007	3	-12	0.005
FLETCBV3	U	5000	3	3	-13	0.007	2	-12	0.006	3	-13	0.006	3	-13	0.006
FLETCHBV	U	5000	3	3	-6	0.006	2	-3	0.005	3	-6	0.005	3	-5	0.007
FLETCHCR	U	1000	3	3	-13	0.001	2	-11	0.002	3	-13	0.002	3	-12	0.001
FLOSP2HH	C	2883	9	9	-10	0.006	5	3	0.004	9	-10	0.006	9	-11	0.006
FLOSP2HL	C	2883	9	9	-10	0.007	5	3	0.004	9	-10	0.006	9	-11	0.006
FLOSP2HM	C	2883	9	9	-10	0.007	5	3	0.005	9	-10	0.007	9	-11	0.006
FLOSP2TH	C	2883	9	9	-10	0.007	5	3	0.003	9	-10	0.007	9	-11	0.005
FLOSP2TL	C	2883	9	9	-10	0.007	5	3	0.004	9	-10	0.007	9	-11	0.007
FLOSP2TM	C	2883	9	9	-10	0.007	5	3	0.004	9	-10	0.007	9	-11	0.005
FMINSRF2	U	5625	9	9	-13	0.019	5	41	0.011	9	-13	0.018	9	-13	0.018
FREUROTH	U	5000	3	3	-11	0.005	2	-9	0.005	3	-11	0.007	3	-11	0.007
GASOIL	C	10403	1602	1602	-	-	5	-12	0.006	5	-12	0.004	5	-12	0.006
GAUSSELM	C	22140	79	79	-11	0.045	41	3	0.018	49	-11	0.042	49	-11	0.041
GENHUMPS	U	5000	3	3	-12	0.005	2	-9	0.004	3	-12	0.007	3	-12	0.005
GENROSE	U	500	3	3	-13	0.001	2	-12	0.001	3	-13	0.002	3	-13	0.001
GILBERT	C	5000	1	1	-16	0.004	1	-16	0.004	1	-16	0.004	1	-16	0.003
GLIDER	C	5214	1605	1605	-	-	4	-9	0.006	6	-12	0.008	6	-12	0.009
GOULDQP2	C	19999	3	3	-12	0.013	2	-10	0.009	3	-12	0.012	3	-12	0.012
GOULDQP3	C	19999	4	4	-12	0.022	2	-10	0.019	4	-12	0.024	4	-11	0.023
GRIDGENA	U	6218	18	18	-2	0.055	10	85	0.026	18	-2	0.054	18	-2	0.052
GRIDNETA	C	7564	1	1	-16	0.006	1	-16	0.005	1	-16	0.006	1	-16	0.006
GRIDNETB	C	7564	1	1	-16	0.007	1	-16	0.006	1	-16	0.006	1	-16	0.007
GRIDNETC	C	7564	1	1	-16	0.006	1	-16	0.007	1	-16	0.006	1	-16	0.005
GRIDNETD	C	7564	3	3	-13	0.009	2	-11	0.007	3	-13	0.009	3	-13	0.008
GRIDNETE	C	7564	3	3	-14	0.009	2	-10	0.007	3	-14	0.009	3	-13	0.010
GRIDNETF	C	7564	3	3	-13	0.009	2	-10	0.008	3	-13	0.009	3	-13	0.008
GRIDNETG	C	7564	61	61	-13	0.016	60	-11	0.010	60	-13	0.015	60	-13	0.014
GRIDNETH	C	7564	61	61	-14	0.016	60	-10	0.010	60	-14	0.015	60	-13	0.016
GRIDNETI	C	7564	61	61	-13	0.016	60	-10	0.009	60	-13	0.016	60	-13	0.015
HADAMARD	C	401	20	20	-14	0.004	20	-11	0.003	20	-14	0.006	20	-14	0.005
HAGER1	C	5001	1	1	-20	0.002	1	-20	0.003	1	-20	0.002	1	-20	0.002
HAGER2	C	5001	3	3	-17	0.006	2	-14	0.006	3	-17	0.006	3	-17	0.004
HAGER3	C	5001	5	5	-16	0.008	3	-14	0.005	5	-16	0.007	5	-16	0.007
HAGER4	C	5001	3	3	-17	0.005	2	-15	0.004	3	-17	0.005	3	-17	0.005
HANGING	C	3600	5	5	-12	0.007	3	3	0.006	5	-12	0.007	5	-11	0.007
HELSBY	C	1408	1	1	-21	0.001	1	-21	0.001	1	-21	0.001	1	-21	0.002
HUES-MOD	C	5000	1	1	-20	0.003	1	-20	0.003	1	-20	0.005	1	-20	0.004
HUESTIS	C	5000	1	1	-inf	0.004	1	-inf	0.005	1	-inf	0.005	1	-inf	0.005
HVYCRASH	C	4004	3	3	-12	0.004	3	-12	0.004	3	-12	0.004	3	-13	0.003
HYDROELL	C	1009	3	3	-22	0.002	2	-20	0.002	3	-22	0.001	3	-22	0.002
HYDROELM	C	505	3	3	-22	0.002	2	-20	0.001	3	-22	0.001	3	-22	0.001
INDEF	U	5000	5000	5000	-	-	3	-10	0.010	3	-10	0.009	3	-10	0.010
INTEGREQ	C	502	1	1	-17	0.001	1	-17	0.001	1	-17	0.001	1	-17	0.001
JANNSON3	C	20000	3	3	-11	0.017	2	-12	0.018	3	-11	0.018	3	-11	0.018
JANNSON4	C	10000	2	2	-16	0.007	2	-16	0.007	2	-16	0.007	2	-16	0.007
JIMACK	U	3549	81	81	-9	0.636	39	128	0.123	39	123	0.127	81	-9	0.589
JNLBRNG1	U	10000	5	5	-12	0.017	3	29	0.013	5	-12	0.017	5	-12	0.017
JNLBRNG2	U	10000	5	5	-12	0.018	3	29	0.013	5	-12	0.017	5	-12	0.016
JNLBRNGA	U	10000	5	5	-12	0.017	3	28	0.012	5	-12	0.017	5	-12	0.017
JNLBRNGB	U	10000	5	5	-12	0.017	3	29	0.013	5	-12	0.016	5	-12	0.016
JUNKTURN	C	10010	7	7	-14	0.016	4	-13	0.011	7	-14	0.015	7	-14	0.014
KISSING	C	127	42	42	-14	0.006	42	-11	0.004	42	-14	0.007	42	-14	0.006
KISSING2	C	100	25	25	-13	0.003	25	-11	0.001	25	-13	0.003	25	-13	0.002
KTMODEL	C	726	3	3	-15	0.001	2	-10	0.000	3	-15	0.001	3	-10	0.001
LCH	C	3000	11	11	-8	0.012	9	154	0.007	11	-8	0.012	11	-8	0.010
LIARWHD	U	5000	5000	5000	-	-	2	-11	0.008	2	-11	0.009	2	-11	0.009
LINCONT	C	1257	3	3	-13	0.002	2	-15	0.002	3	-13	0.001	3	-13	0.003
LINVERSE	U	1999	9	9	-12	0.007	5	60	0.004	9	-12	0.007	9	-12	0.007
LISWET1	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002
LISWET10	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.003	1	-inf	0.002
LISWET11	C	2002	1	1	-inf	0.001	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002
LISWET12	C	2002	1	1	-inf	0.003	1	-inf	0.002	1	-inf	0.003	1	-inf	0.002
LISWET2	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002
LISWET3	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002
LISWET4	C	2002	1	1	-inf	0.002	1	-inf	0.001	1	-inf	0.002	1	-inf	0.001

Table A.1: Reproducing Hessians for quadratic examples: Complete results (continued)

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
LISWET5	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002	1	-inf	0.002
LISWET6	C	2002	1	1	-inf	0.001	1	-inf	0.001	1	-inf	0.002	1	-inf	0.002
LISWET7	C	2002	1	1	-inf	0.002	1	-inf	0.002	1	-inf	0.003	1	-inf	0.002
LISWET8	C	2002	1	1	-inf	0.001	1	-inf	0.002	1	-inf	0.002	1	-inf	0.001
LISWET9	C	2002	1	1	-inf	0.002	1	-inf	0.003	1	-inf	0.002	1	-inf	0.002
LMINSURF	U	5625	9	9	-13	0.019	5	43	0.011	9	-13	0.020	9	-13	0.018
LUBRIF	C	3751	6	6	-9	0.007	3	-9	0.004	6	-9	0.005	6	-11	0.005
LUBRIFC	C	3751	6	6	-6	0.005	3	-7	0.004	6	-6	0.006	6	-10	0.006
LUKVLE1	C	10000	3	3	-11	0.012	2	-9	0.010	3	-11	0.011	3	-11	0.011
LUKVLE10	C	10000	2	2	-13	0.010	2	-12	0.009	2	-13	0.010	2	-13	0.008
LUKVLE11	C	9998	4	4	-11	0.012	3	42	0.010	4	-11	0.012	4	-11	0.012
LUKVLE12	C	9997	2502	2502	-	-	3	-10	0.012	4	-10	0.014	4	-10	0.013
LUKVLE13	C	9998	3	3	-12	0.011	2	-12	0.010	3	-12	0.011	3	-12	0.010
LUKVLE14	C	9998	2	2	-8	0.009	2	-10	0.010	2	-8	0.009	2	-8	0.009
LUKVLE15	C	9997	3	3	-8	0.012	2	-7	0.010	3	-8	0.013	3	-10	0.011
LUKVLE16	C	9997	3	3	-12	0.010	2	-12	0.009	3	-12	0.010	3	-12	0.011
LUKVLE17	C	9997	3	3	-12	0.010	2	-12	0.009	3	-12	0.009	3	-12	0.009
LUKVLE18	C	9997	3	3	-12	0.010	2	-12	0.010	3	-12	0.009	3	-12	0.010
LUKVLE2	C	10000	4	4	-10	0.012	2	-9	0.009	4	-10	0.011	4	-11	0.012
LUKVLE3	C	10000	4	4	-11	0.014	3	32	0.012	4	-11	0.014	4	-10	0.013
LUKVLE4	C	10000	3	3	-4	0.013	2	1	0.011	3	-4	0.011	3	-4	0.011
LUKVLE5	C	10002	5	5	-10	0.018	3	-9	0.013	5	-10	0.017	5	-11	0.017
LUKVLE6	C	9999	13	13	-9	0.057	7	-7	0.026	13	-9	0.055	13	-9	0.055
LUKVLE7	C	10000	3	3	-12	0.007	2	-10	0.009	3	-12	0.008	3	-12	0.008
LUKVLE8	C	10000	5	5	-10	0.017	5	-9	0.013	5	-10	0.017	5	-9	0.015
LUKVLE9	C	10000	3	3	-12	0.008	2	-10	0.008	3	-12	0.009	3	-11	0.009
LUKVLI1	C	10000	3	3	-11	0.012	2	-9	0.011	3	-11	0.012	3	-11	0.011
LUKVLI10	C	10000	2	2	-13	0.010	2	-12	0.009	2	-13	0.010	2	-13	0.010
LUKVLI11	C	9998	4	4	-11	0.012	3	42	0.010	4	-11	0.012	4	-11	0.012
LUKVLI12	C	9997	2502	2502	-	-	3	-10	0.012	4	-10	0.014	4	-10	0.015
LUKVLI13	C	9998	3	3	-12	0.010	2	-12	0.010	3	-12	0.011	3	-12	0.010
LUKVLI14	C	9998	2	2	-8	0.010	2	-10	0.008	2	-8	0.008	2	-8	0.009
LUKVLI15	C	9997	3	3	-8	0.012	2	-7	0.013	3	-8	0.012	3	-10	0.011
LUKVLI16	C	9997	3	3	-12	0.010	2	-12	0.009	3	-12	0.009	3	-12	0.009
LUKVLI17	C	9997	3	3	-12	0.009	2	-12	0.009	3	-12	0.011	3	-12	0.009
LUKVLI18	C	9997	3	3	-12	0.009	2	-12	0.009	3	-12	0.010	3	-12	0.009
LUKVLI2	C	10000	4	4	-10	0.012	2	-9	0.010	4	-10	0.013	4	-11	0.013
LUKVLI3	C	10000	4	4	-11	0.014	3	32	0.011	4	-11	0.014	4	-10	0.013
LUKVLI4	C	10000	3	3	-4	0.012	2	1	0.011	3	-4	0.013	3	-4	0.012
LUKVLI5	C	10002	5	5	-10	0.017	3	-9	0.011	5	-10	0.018	5	-11	0.016
LUKVLI6	C	9999	13	13	-9	0.056	7	-7	0.027	13	-9	0.056	13	-9	0.055
LUKVLI7	C	10000	3	3	-12	0.007	2	-10	0.007	3	-12	0.007	3	-12	0.006
LUKVLI8	C	10000	5	5	-10	0.018	5	-9	0.014	5	-10	0.017	5	-9	0.016
LUKVLI9	C	10000	3	3	-12	0.009	2	-10	0.008	3	-12	0.010	3	-11	0.009
MADSSCHJ	C	201	1	1	-16	0.001	1	-16	0.000	1	-16	0.001	1	-16	0.001
MANNE	C	6000	1	1	-16	0.004	1	-16	0.003	1	-16	0.003	1	-16	0.003
MARINE	C	11215	401	401	-	-	3	-12	0.005	3	-12	0.005	3	-13	0.005
MCCORMCK	U	5000	3	3	-12	0.007	2	-10	0.005	3	-12	0.006	3	-12	0.006
METHANOL	C	12005	2405	2405	-	-	6	-12	0.008	7	-11	0.009	7	-12	0.008
MINC44	C	1113	127	127	-	-	10	-8	0.003	37	-12	0.004	37	-12	0.003
MINPERM	C	1113	127	127	-	-	10	-8	0.002	37	-12	0.003	37	-12	0.004
MINSURFO	U	5306	7	7	-13	0.013	4	24	0.009	7	-13	0.014	7	-13	0.012
MODBEALE	U	20000	3	3	-11	0.022	2	-9	0.018	3	-11	0.022	3	-11	0.021
MOREBV	U	5000	5	5	-12	0.009	3	-10	0.006	5	-12	0.009	5	-12	0.009
MOSARQP1	C	2500	10	10	-15	0.002	10	-14	0.002	10	-15	0.004	10	-14	0.002
MOSARQP2	C	2500	10	10	-15	0.002	10	-14	0.002	10	-15	0.002	10	-14	0.002
MSQRTA	C	1024	64	64	-12	0.139	63	176	0.044	63	171	0.066	64	-12	0.136
MSQRTB	C	1024	64	64	-12	0.139	63	176	0.044	63	171	0.047	64	-12	0.134
NCB20	U	5010	40	40	-11	0.204	20	-7	0.061	40	-11	0.203	40	-11	0.201
NCB20B	U	5000	39	39	-8	0.214	20	-7	0.060	39	-8	0.261	39	-8	0.208
NCVXBQP1	U	10000	9	9	-11	0.029	7	48	0.018	9	-11	0.028	9	-11	0.026
NCVXBQP2	U	10000	9	9	-11	0.027	7	48	0.017	9	-11	0.027	9	-11	0.026
NCVXBQP3	U	10000	9	9	-11	0.030	7	49	0.018	9	-11	0.026	9	-11	0.025
NCVXQP1	C	10000	9	9	-11	0.027	7	48	0.017	9	-11	0.027	9	-11	0.025
NCVXQP2	C	10000	9	9	-11	0.026	7	48	0.018	9	-11	0.028	9	-11	0.026
NCVXQP3	C	10000	9	9	-11	0.029	7	49	0.016	9	-11	0.027	9	-11	0.026
NCVXQP4	C	10000	9	9	-11	0.029	7	48	0.017	9	-11	0.028	9	-11	0.026
NCVXQP5	C	10000	9	9	-11	0.030	7	48	0.018	9	-11	0.027	9	-11	0.026
NCVXQP6	C	10000	9	9	-11	0.027	7	49	0.016	9	-11	0.028	9	-11	0.027
NCVXQP7	C	10000	9	9	-11	0.027	7	48	0.018	9	-11	0.027	9	-11	0.026
NCVXQP8	C	10000	9	9	-11	0.028	7	48	0.017	9	-11	0.028	9	-11	0.026
NCVXQP9	C	10000	9	9	-11	0.027	7	49	0.019	9	-11	0.028	9	-11	0.026
NET4	C	66816	13	13	-10	0.105	4	-10	0.075	13	-10	0.105	13	-12	0.096
NLMSURF	U	5625	9	9	-13	0.019	5	42	0.011	9	-13	0.019	9	-13	0.018
NOBNDTOR	U	5476	5	5	-12	0.010	3	18	0.008	5	-12	0.009	5	-12	0.009
NONCVXU2	U	5000	7	7	-11	0.014	6	33	0.010	7	-11	0.013	7	-12	0.012
NONCVXUN	U	5000	9	9	-12	0.014	7	49	0.009	9	-12	0.013	9	-12	0.013
NONDIA	U	5000	4999	4999	-	-	2	-12	0.008	2	-12	0.008	2	-12	0.008
NONDQUAR	U	5000	5000	5000	-	-	3	-9	0.010	4	-11	0.010	4	-11	0.011
NONMSQRT	U	4900	70	70	-11	0.840	70	-8	0.255	70	-11	0.939	70	-11	0.806
NONSCOMP	U	5000	3	3	-12	0.007	2	-10	0.005	3	-12	0.005	3	-11	0.005
NUFFIELD	C	940	25	25	-5	0.013	13	53	0.006	25	-5	0.014	25	-6	0.013

Table A.1: Reproducing Hessians for quadratic examples: Complete results (continued)

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
OBSTCLAE	U	10000	5	5	-13	0.018	3	27	0.013	5	-13	0.018	5	-12	0.016
OBSTCLAL	U	10000	5	5	-13	0.024	3	27	0.013	5	-13	0.016	5	-12	0.018
OBSTCLBL	U	10000	5	5	-13	0.017	3	27	0.013	5	-13	0.017	5	-12	0.016
OBSTCLBM	U	10000	5	5	-13	0.018	3	27	0.013	5	-13	0.018	5	-12	0.016
OBSTCLBU	U	10000	5	5	-13	0.016	3	27	0.013	5	-13	0.017	5	-12	0.017
ODC	U	5184	7	7	-11	0.012	4	24	0.008	7	-11	0.012	7	-11	0.013
OPTCDEG2	C	4502	1	1	-18	0.002	1	-18	0.002	1	-18	0.002	1	-18	0.002
OPTCDEG3	C	4502	1	1	-19	0.002	1	-19	0.002	1	-19	0.003	1	-19	0.002
OPTCTRL3	C	4502	1	1	-16	0.005	1	-16	0.005	1	-16	0.005	1	-16	0.003
OPTCTRL6	C	4502	1	1	-16	0.003	1	-16	0.004	1	-16	0.003	1	-16	0.005
OPTMASS	C	3010	1	1	-16	0.001	1	-16	0.002	1	-16	0.002	1	-16	0.001
ORBIT2	C	2698	2698	2698	-	-	16	87	0.018	25	-15	0.037	25	-15	0.038
ORTHRDM2	C	8003	8003	8003	-	-	5	-10	0.020	5	-11	0.021	5	-11	0.020
ORTHRDS2	C	5003	5003	5003	-	-	5	-11	0.011	5	-11	0.012	5	-11	0.014
ORTHREGA	C	8197	8193	8193	-	-	5	-11	0.020	5	-12	0.021	5	-11	0.019
ORTHREGC	C	5005	5001	5001	-	-	5	-10	0.013	5	-11	0.013	5	-11	0.012
ORTHREGD	C	5003	5003	5003	-	-	5	-11	0.012	5	-11	0.012	5	-11	0.012
ORTHREGE	C	7506	2504	2504	-	-	5	3	0.010	5	2	0.011	5	1	0.009
ORTHREGF	C	4805	3203	3203	-	-	5	-12	0.010	5	-13	0.010	5	-13	0.010
ORTHRGDM	C	10003	10003	10003	-	-	5	-10	0.024	5	-12	0.031	5	-12	0.024
ORTHRGDS	C	5003	5003	5003	-	-	5	-11	0.012	5	-11	0.012	5	-11	0.012
PENTDI	U	5000	5	5	-12	0.009	3	-9	0.007	5	-12	0.009	5	-12	0.007
PINENE	C	8805	602	602	-	-	4	-11	0.004	4	-12	0.003	4	-12	0.004
POROUS1	C	5184	1	1	-13	0.004	1	-13	0.004	1	-13	0.005	1	-13	0.004
POROUS2	C	5184	1	1	-12	0.003	1	-12	0.005	1	-12	0.005	1	-12	0.004
PORTSNQP	C	100000	1	1	-inf	0.100	1	-inf	0.067	1	-inf	0.068	1	-inf	0.087
PORTSQP	C	100000	1	1	-inf	0.065	1	-inf	0.079	1	-inf	0.071	1	-inf	0.067
POWELL20	C	5000	1	1	-inf	0.003	1	-inf	0.005	1	-inf	0.005	1	-inf	0.003
POWELLSG	U	5000	3	3	-13	0.006	3	-10	0.005	3	-13	0.007	3	-12	0.005
PRIMAL2	C	649	1	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001
PRIMAL3	C	745	1	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001
PRIMAL4	C	1489	1	1	-inf	0.001	1	-inf	0.002	1	-inf	0.001	1	-inf	0.001
PRIMALC8	C	520	1	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001	1	-inf	0.001
QPBAND	C	50000	3	3	-12	0.055	2	-8	0.050	3	-12	0.056	3	-11	0.055
QPNBAND	C	50000	3	3	-12	0.055	2	-8	0.049	3	-12	0.063	3	-11	0.055
QR3D	C	610	40	40	-14	0.013	21	-12	0.006	40	-14	0.013	40	-13	0.014
QR3DBD	C	457	23	23	-14	0.008	21	-12	0.004	23	-14	0.006	23	-14	0.010
QRTQUAD	U	5000	3900	3900	-	-	2	-6	0.008	3	-12	0.007	3	-12	0.008
QUARTC	U	5000	1	1	-16	0.005	1	-16	0.004	1	-16	0.003	1	-16	0.004
QUDLIN	U	5000	3	3	-13	0.004	2	-10	0.003	3	-13	0.004	3	-13	0.003
RAYBENDL	U	2050	6	6	-13	0.004	4	7	0.004	6	-13	0.004	6	-13	0.004
RAYBENDS	U	2050	14	14	-12	0.013	8	78	0.007	14	-12	0.012	14	-12	0.017
READING1	C	4002	2	2	-12	0.004	2	-12	0.003	2	-12	0.004	2	-12	0.004
READING3	C	4002	2	2	-12	0.005	2	-12	0.005	2	-12	0.004	2	-12	0.005
READING4	C	5001	3	3	-9	0.006	2	-1	0.006	3	-9	0.006	3	-10	0.006
READING5	C	5001	3	3	-9	0.006	2	-1	0.005	3	-9	0.006	3	-10	0.006
READING7	C	1002	2	2	-13	0.002	2	-13	0.001	2	-13	0.001	2	-13	0.002
READING8	C	2002	2	2	-11	0.002	2	-11	0.003	2	-11	0.003	2	-11	0.002
READING9	C	10002	2	2	-16	0.009	2	-15	0.009	2	-16	0.009	2	-17	0.009
ROBOTARM	C	4412	3209	3209	-	-	3	-13	0.007	3	-14	0.007	3	-13	0.008
ROCKET	C	2407	2006	2006	-	-	5	-6	0.005	7	-12	0.006	7	-11	0.005
ROTDISC	C	905	3	3	-14	0.001	2	-14	0.001	3	-14	0.001	3	-15	0.001
SARO	C	4754	11	11	-inf	0.015	11	-inf	0.010	11	-inf	0.022	11	-inf	0.015
SAROMM	C	5120	11	11	-inf	0.016	11	-inf	0.010	11	-inf	0.015	11	-inf	0.015
SAWPATH	C	583	1	1	-16	0.002	1	-16	0.001	1	-16	0.001	1	-16	0.001
SBRYBND	U	5000	13	13	-5	0.027	7	8	0.014	13	-5	0.028	13	-4	0.028
SCHMVETT	U	5000	5	5	-12	0.009	3	-9	0.006	5	-12	0.008	5	-12	0.010
SCOSINE	U	5000	3	3	-8	0.006	2	1	0.004	3	-8	0.007	3	-8	0.005
SCURLY10	U	10000	21	21	-11	0.121	11	-inf	0.047	21	-11	0.121	21	-11	0.122
SCURLY20	U	10000	41	41	-10	0.469	21	-inf	0.133	41	-10	0.470	41	-10	0.468
SCURLY30	U	10000	61	61	-10	1.200	31	1	0.277	50	2	0.277	61	-10	1.192
SEMICN2U	C	5002	1	1	-19	0.004	1	-19	0.004	1	-19	0.004	1	-19	0.004
SEMICON1	C	5002	1	1	-17	0.003	1	-17	0.003	1	-17	0.005	1	-17	0.004
SEMICON2	C	5002	1	1	-19	0.004	1	-19	0.005	1	-19	0.005	1	-19	0.003
SINEALI	U	1000	3	3	-12	0.001	2	-11	0.001	3	-12	0.003	3	-12	0.003
SINQUAD	U	5000	5000	5000	-	-	2	-11	0.009	2	-11	0.008	2	-11	0.009
SINROSNB	C	1000	3	3	-12	0.002	2	-10	0.001	3	-12	0.002	3	-11	0.002
SMMPSF	C	720	60	60	-13	0.024	40	-4	0.007	40	-3	0.007	60	-13	0.021
SOSQP1	C	5000	2	2	-14	0.005	2	-inf	0.005	2	-14	0.006	2	-14	0.005
SOSQP2	C	5000	2	2	-14	0.005	2	-inf	0.004	2	-14	0.005	2	-14	0.005
SPARSINE	U	5000	56	56	-8	0.161	26	inf	0.052	48	62	0.125	48	-8	0.132
SPARSQUR	U	10000	56	56	-8	0.353	26	300	0.108	48	93	0.269	50	-8	0.277
SPMSQRT	C	4999	6	6	-12	0.010	4	25	0.008	6	-12	0.010	6	-12	0.009
SPMSRTLS	U	4999	8	8	-12	0.014	5	10	0.009	8	-12	0.019	8	-12	0.013
SREADIN3	C	4002	2	2	-12	0.004	2	-12	0.005	2	-12	0.005	2	-12	0.004
SROSENBR	U	5000	2	2	-11	0.005	2	-11	0.004	2	-11	0.004	2	-11	0.005
SSC	U	5184	5	5	-11	0.009	3	16	0.006	5	-11	0.010	5	-11	0.009
SSNLBEAM	C	3003	1	1	-16	0.001	1	-16	0.002	1	-16	0.002	1	-16	0.002
STCQP1	C	8193	121	121	-	-	43	144	0.052	49	136	0.057	75	-10	0.177
STCQP2	C	8193	121	121	-	-	43	144	0.052	49	136	0.057	75	-10	0.168
STEENBRC	C	540	15	15	-12	0.005	15	-9	0.002	15	-12	0.005	15	-12	0.004
STEENBRE	C	540	15	15	-12	0.004	15	-9	0.002	15	-12	0.004	15	-12	0.004

Table A.1: Reproducing Hessians for quadratic examples: Complete results (continued)

				Algorithm 2.1			Algorithm 2.2			Algorithm 2.2/3			Algorithm 2.3		
name	ty	n	deg	sys	err	time	sys	err	time	sys	err	time	sys	err	time
STEENBRG	C	540	15	15	-12	0.004	15	-9	0.003	15	-12	0.004	15	-12	0.004
STEERING	C	2006	1204	1204	-	-	2	-11	0.003	2	-11	0.003	2	-11	0.003
STNQP1	C	8193	121	121	-	-	43	144	0.052	49	136	0.059	75	-10	0.169
STNQP2	C	8193	121	121	-	-	43	144	0.052	49	136	0.057	75	-10	0.168
SVANBERG	C	5000	1	1	-15	0.007	1	-15	0.005	1	-15	0.004	1	-15	0.003
TESTQUAD	U	5000	1	1	-16	0.003	1	-16	0.003	1	-16	0.005	1	-16	0.003
TOINTGSS	U	5000	5	5	-12	0.009	3	-10	0.007	5	-12	0.008	5	-12	0.009
TORSION1	U	5476	5	5	-12	0.010	3	18	0.007	5	-12	0.009	5	-12	0.010
TORSION2	U	5476	5	5	-12	0.010	3	18	0.007	5	-12	0.010	5	-12	0.009
TORSION3	U	5476	5	5	-12	0.009	3	18	0.007	5	-12	0.009	5	-12	0.009
TORSION4	U	5476	5	5	-12	0.010	3	18	0.007	5	-12	0.010	5	-12	0.009
TORSION5	U	5476	5	5	-12	0.009	3	18	0.007	5	-12	0.010	5	-12	0.010
TORSION6	U	5476	5	5	-12	0.013	3	18	0.008	5	-12	0.009	5	-12	0.009
TORSIONA	U	5476	5	5	-13	0.010	3	18	0.008	5	-13	0.010	5	-12	0.008
TORSIONB	U	5476	5	5	-13	0.010	3	18	0.008	5	-13	0.010	5	-12	0.009
TORSIONC	U	5476	5	5	-13	0.010	3	18	0.008	5	-13	0.010	5	-12	0.008
TORSIOND	U	5476	5	5	-13	0.009	3	18	0.007	5	-13	0.010	5	-12	0.009
TORSIONE	U	5476	5	5	-13	0.010	3	18	0.008	5	-13	0.010	5	-12	0.010
TORSIONF	U	5476	5	5	-13	0.010	3	18	0.007	5	-13	0.009	5	-12	0.009
TQUARTIC	U	5000	5000	5000	-	-	2	-13	0.008	2	-13	0.009	2	-13	0.008
TRAINF	C	4008	2	2	-16	0.002	2	-17	0.002	2	-16	0.002	2	-16	0.002
TRAINH	C	4008	2	2	-16	0.004	2	-16	0.003	2	-16	0.002	2	-16	0.004
TRIDIA	U	5000	3	3	-12	0.006	2	-10	0.006	3	-12	0.006	3	-12	0.007
TWIRIBG1	C	3127	1886	1886	-	-	105	-	-	105	-	-	1885	-	-
TWIRIMD1	C	1247	660	660	-	-	63	24	0.077	63	24	0.076	652	-	-
UBH1	C	9009	1	1	-inf	0.003	1	-inf	0.003	1	-inf	0.003	1	-inf	0.003
UBH5	C	5010	1	1	-16	0.001	1	-16	0.001	1	-16	0.001	1	-16	0.001
WALL10	U	1461	42	42	-10	0.016	12	23	0.007	42	-10	0.016	42	-10	0.016
WALL100	U	149624	42	42	-8	1.947	12	inf	0.811	42	-8	1.950	42	-7	1.743
WALL20	U	5924	42	42	-9	0.068	12	61	0.027	42	-9	0.068	42	-8	0.066
WALL50	U	37311	42	42	-8	0.456	12	144	0.176	42	-8	0.459	42	-8	0.433
WOODSNE	C	4000	1	1	-16	0.002	1	-16	0.001	1	-16	0.002	1	-16	0.002
YAO	C	2002	1	1	-inf	0.002	1	-inf	0.001	1	-inf	0.002	1	-inf	0.002
YATP1SQ	C	123200	352	352	-	-	3	-9	0.141	3	-9	0.140	3	-10	0.140
YATP2SQ	C	123200	352	352	-	-	3	-11	0.147	3	-11	0.154	3	-11	0.147
ZAMB2	C	3966	5	5	-14	0.004	3	-13	0.003	5	-14	0.004	5	-14	0.004
ZIGZAG	C	3004	1	1	-16	0.000	1	-16	0.001	1	-16	0.001	1	-16	0.001

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, third edition, 1999.

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

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. In R. Glowinski and A. Lichnewsky, editors, *Computing Methods in Applied Sciences and Engineering*, pages 42–51, Philadelphia, USA, 1990. SIAM.

[4] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization. *Mathematical Programming, Series A*, 73(1):73–110, 1996.

[5] E. Corwin and A. Logar. Sorting in linear time — variations on the bucket sort. *Journal of Computing Sciences in Colleges*, 20(1):197–202, 2004.

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

[7] R. Fletcher. An optimal positive definite update for sparse Hessian matrices. *SIAM Journal on Optimization*, 5(1):192–217, 1995.

[8] R. Fletcher, A. Grothey, and S. Leyffer. Computing sparse Hessian and Jacobian approximations with optimal hereditary properties. In A.R. Conn L.T. Biegler, T.F. Coleman and F.N. Santosa, editors, *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, volume 93 of *IMA Volumes in Mathematics and its Applications*, pages 37–52, Heidelberg, Berlin, New York, 1997. Springer Verlag.

[9] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Computing forward-difference intervals for numerical optimization. *SIAM Journal on Scientific and Statistical Computing*, 4:310–321, 1983.

[10] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[11] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2003.

[12] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312, London, 1982. Academic Press.

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

[14] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* SIAM, Philadelphia, second edition, 2008.

[15] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large-scale optimization. *Mathematical Programming, Series B*, 45(3):503–528, 1989.

[16] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[17] J. Nocedal and S. J. Wright. *Numerical Optimization.* Series in Operations Research. Springer Verlag, Heidelberg, Berlin, New York, second edition, 2006.

[18] 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.

[19] H. H. Seward. Information sorting in the application of electronic digital computers to business operations. Report R-232, Digital Computer Laboratory, Massachusetts Institute of Technology, USA, 1954.

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

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

[22] Ph. L. Toint. Some numerical result using a sparse matrix updating formula in unconstrained optimization. *Mathematics of Computation*, 32(143):839–851, 1978.