On Iterated-Subspace Minimization Methods for Nonlinear Optimization

by A.R. Conn[1], Nick Gould[2],
A. Sartenaer[3] and Ph.L. Toint[3]

Report 94/13 May 18, 1994

[1] IBM T.J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, USA
Email : arconn@watson.ibm.com

[2] Rutherford Appleton Laboratory, Chilton, Oxfordshire, England
Email : nimg@directory.rl.ac.uk
Current reports available by anonymous ftp from the directory
"pub/reports" on camelot.cc.rl.ac.uk (internet 130.246.8.61)

[3] Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles,
B-5000 Namur, Belgium, EC
Email : as@math.fundp.ac.be and pht@math.fundp.ac.be
Current reports available by anonymous ftp from the directory
"pub/reports" on thales.math.fundp.ac.be (internet 138.48.4.14)

Keywords : Unconstrained optimization, large-scale computation, convergence theory.

Mathematics Subject Classifications: 65K05, 90C30

On Iterated-Subspace Minimization Methods for Nonlinear Optimization

A. R. Conn, Nick Gould, A. Sartenaer and Ph. L. Toint

May 18, 1994

Abstract

We consider a class of Iterated-Subspace Minimization (ISM) methods for solving large-scale unconstrained minimization problems. At each major iteration of such a method, a low-dimensional manifold, the iterated subspace, is constructed and an approximate minimizer of the objective function in this manifold then determined. The iterated subspace is chosen to contain vectors which ensure global convergence of the overall scheme and may also contain vectors which encourage fast asymptotic convergence. We demonstrate the efficacy of this approach on a collection of large problems and indicate a number of avenues of future research.

1 Introduction

In this paper, we consider finding a local solution of the unconstrained minimization problem,

$$\underset{\boldsymbol{x}\in\Re^n}{\text{minimize}} \quad f(\boldsymbol{x}), \tag{1.1}$$

where we assume, for simplicity, that the objective function $f \in C^2$. We are particularly interested in the case where n is sufficiently large that methods appropriate for small problems — such as those which might maintain a dense factorization of a suitable approximation of the Hessian matrix, see, for example, Gill *et al.* (1981), Dennis and Schnabel (1983) and Fletcher (1987) — are impractical.

We are primarily concerned with the commonly occurring case in which the cost of evaluating the value of the objective function and its derivatives, at a given point $\boldsymbol{x}$, is less significant than the cost of solving, for instance, the Newton equations. Our experience with the large-scale nonlinear optimization package LANCELOT (see Conn *et al.*, 1992b) has been that it is the linear-algebra cost which tends to dominate when solving a significant number of widely differing application

[0]The research of Conn and Toint was supported in part by the Advanced Research Projects Agency of the Departement of Defense and was monitored by the Air Force Office of Scientific Research under Contract No F49620-91-C-0079.

problems (see, Conn *et al.*, 1992c and Conn *et al.*, 1992a). Thus, it would appear desirable in these cases to attempt to reduce the linear-algebra costs, even if this results in an increase in the number of objective function evaluations.

The most common methods for unconstrained minimization either determine a search direction followed by a linesearch or use the trust-region approach (see, for example, Dennis and Schnabel, 1983). In the former case, a simple model of the underlying objective function is constructed in order to determine the search direction. By contrast, in the latter case, an approximate minimizer of the model within a restricted domain (the trust region) is determined. This model minimizer is then used as a prediction of the actual minimizer of the true objective. In a trust-region method, success of this process is measured by comparing the model and true function values at the predicted minimizer. In linesearch methods, the true function is used to establish a step size. Thus, both of these approaches may be considered to perform their multi-dimensional work with respect to a model whilst probing the true function uni-dimensionally. Of course, the model does make use of the true function and perhaps its derivatives — maybe at more than a single point.

In this paper, we take the view that the above schemes are quite wasteful, given the amount of information that may have been accrued during the (approximate) minimization of the model. In particular, the model may have been sampled in a number of potentially interesting directions, of which only the aggregate direction is normally considered to be of significance.

We also believe that, provided function and derivative values are inexpensive to compute relative to the linear-algebra costs, an (approximate) low-dimensional minimization is a trivial calculation. Indeed, we feel that, quite generally, the small-scale unconstrained minimization problem has effectively been solved in that there is high-quality, robust, general-purpose software easily available for such problems, and that such software is normally capable of solving problems of modest dimensions - say up to 100 variable problems - extremely fast on current workstations *provided that* function evaluation is cheap. Of course there are, and will continue to be, small-scale problems which are challenging, because they are so nonlinear that algorithms implemented in fixed, finite precision arithmetic are unsuccessful, but in our experience such examples occur rarely in practice.

Thus, in this paper, we propose methods which aim to investigate the *true* objective function in a space larger than the one-dimensional space which is normally associated with linesearch or trust-region methods. We do this knowing that, so long as the space is relatively modest, the approximate multi-dimensional minimization will still be a manageable calculation. Moreover, by carefully choosing the space that we investigate, we hope to reduce significantly the linear-algebra costs while still maintaining global, and fast asymptotic, convergence.

A particular form of this idea has been given by Saad (1990) for the solution of nonlinear systems of equations. Here, a sequence of iterates are generated as least-squares solutions to the equations in suitable Krylov subspaces. The principal difference is that, in Saad's proposal, the entire Krylov subspace generated is used, while, as we shall see, this is in general quite unnecessary.

Given an initial estimate of the solution to (1.1), $\boldsymbol{x}^{(0)}$, and an iteration count, k, set initially to zero, a prototype algorithm might be as follows:

1. Stop with the solution estimate $\boldsymbol{x}^{(k)}$ if convergence tests are satisfied.
2. Determine a full-rank subspace matrix $\boldsymbol{S}^{(k)} \in \Re^{n \times s^{(k)}}$, where $s^{(k)} \ll n$.
3. Approximately solve the $s^{(k)}$-dimensional minimization problem

$$\underset{\boldsymbol{y} \in \Re^{s^{(k)}}}{\text{minimize}} \quad f(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)} \boldsymbol{y}), \tag{1.2}$$

   set

$$\boldsymbol{x}^{(k+1)} = \text{(approx)} \underset{\boldsymbol{y} \in \Re^{s^{(k)}}}{\arg\min} \quad f(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)} \boldsymbol{y}), \tag{1.3}$$

   replace k by $k+1$ and return to step 1.

We refer to such a method as *Iterated-Subspace Minimization* or ISM for short. This is, of course, a multi-dimensional subspace analog of the unidimensional-subspace linesearch method. We are interested in the following issues.

- What is a good choice for $s^{(k)}$?
- How do we determine the *Iterated-Subspace* matrix $\boldsymbol{S}^{(k)}$?
- What do we mean by "approximate" in the problem (1.3)?
- Are there methods which are particularly appropriate for solving (1.2)?
- What can we say about the convergence of such a method?
- If we can establish convergence, what can we say about its asymptotic rate?

In this paper, we make preliminary attempts to answer all of these questions.

We will use the following notation. Bold lower and upper case roman letters indicate vectors and matrices, respectively, while greek and normal roman letters denote scalars. Script style letters are index sets. A superscript (k) indicates a quantity which occurs at the k-th iteration or which is evaluated at $\boldsymbol{x}^{(k)}$.

We let $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{H}(\boldsymbol{x})$, respectively, indicate the gradient, $\nabla_x f(\boldsymbol{x})$, and Hessian matrix, $\nabla_{xx} f(\boldsymbol{x})$, of the objective function. We define

$$f_s^{(k)}(\boldsymbol{y}) \stackrel{\text{def}}{=} f(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)} \boldsymbol{y}), \tag{1.4}$$

$\boldsymbol{g}_s^{(k)}(\boldsymbol{y}) \stackrel{\text{def}}{=} \nabla_y f_s^{(k)}(\boldsymbol{y})$ and $\boldsymbol{H}_s^{(k)}(\boldsymbol{y}) \stackrel{\text{def}}{=} \nabla_{yy} f_s^{(k)}(\boldsymbol{y})$, and will make use of the derivative identities

$$\boldsymbol{g}_s^{(k)}(\boldsymbol{y}) = \boldsymbol{S}^{(k)T} \boldsymbol{g}(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)} \boldsymbol{y}) \tag{1.5}$$

and

$$\boldsymbol{H}_s^{(k)}(\boldsymbol{y}) = \boldsymbol{S}^{(k)T}\boldsymbol{H}(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y})\boldsymbol{S}^{(k)}. \tag{1.6}$$

The paper is organised as follows. In Section 2, we analyse the convergence of the algorithm given in the introduction. We discuss a number of computationally attractive ISM methods in Section 3, and we report on some preliminary numerical experience when solving some relatively large test examples, from the CUTE test suite (see Bongartz *et al.*, 1993), in Section 4. Possible extensions, to the cases where there are linear or nonlinear constraints present, are given in Section 5. We conclude, in Section 6, by offering our perspectives of this and future work.

2 Convergence analysis of a general algorithm

Global convergence of the above scheme can be guaranteed under fairly general assumptions. Suppose that we are able to pick consecutive iterates $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y}^{(k)}$ for which the Goldstein (1964) conditions

$$f^{(k)} + \beta\boldsymbol{g}^{(k)T}\boldsymbol{S}^{(k)}\boldsymbol{y}^{(k)} \leq f^{(k+1)} \leq f^{(k)} + \alpha\boldsymbol{g}^{(k)T}\boldsymbol{S}^{(k)}\boldsymbol{y}^{(k)}, \tag{2.1}$$

for some $0 < \alpha \leq \beta < 1$, are satisfied, where $\boldsymbol{y}^{(k)}$ is the approximate solution of (1.2). Suppose, furthermore, that

$$\frac{-\boldsymbol{g}^{(k)T}\boldsymbol{S}^{(k)}\boldsymbol{y}^{(k)}}{\|\boldsymbol{S}^{(k)}\boldsymbol{y}^{(k)}\|_2} \geq \epsilon, \tag{2.2}$$

for some $\epsilon > 0$. Then the ISM algorithm from Section 1 is globally convergent to a stationary point for the problem (1.1) from any starting point so long as f is bounded from below and has a Lipschitz-continuous gradient (see, for example, Dennis and Schnabel, 1983, Theorem 6.3.3). We may rewrite (2.1) as

$$f_s^{(k)}(\boldsymbol{0}) + \beta\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}^{(k)} \leq f_s^{(k)}(\boldsymbol{y}^{(k)}) \leq f_s^{(k)}(\boldsymbol{0}) + \alpha\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}^{(k)} \tag{2.3}$$

and ensure that (2.2) is satisfied by requiring that

$$\frac{-\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}^{(k)}}{\|\boldsymbol{y}^{(k)}\|_2} \geq \epsilon\|\boldsymbol{S}^{(k)}\|_2. \tag{2.4}$$

This is relevant as now the global convergence conditions may be verified in terms of the inner-minimization function f_s and its gradient. Similar global convergence results can be obtained if we replace condition (2.1) by the Armijo (1966) backtracking strategy (see Bertsekas, 1982, Section 1.3). It may, however, be difficult to design general algorithms which ensure that conditions (2.3) and (2.4) are satisfied on exit from the inner minimization. Thus, it may be preferable to impose extra conditions on the iterates generated during the inner minimization to ensure overall global convergence. With this in mind, suppose that we can find any point $\boldsymbol{y}_s^{(k)}$ for which

$$f_s^{(k)}(\boldsymbol{0}) + \beta\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}_s^{(k)} \leq f_s^{(k)}(\boldsymbol{y}_s^{(k)}) \leq f_s^{(k)}(\boldsymbol{0}) + \alpha\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}_s^{(k)} \tag{2.5}$$

and

$$\frac{-\boldsymbol{g}_s^{(k)}(\boldsymbol{0})^T\boldsymbol{y}_s^{(k)}}{\|\boldsymbol{y}_s^{(k)}\|_2} \geq \epsilon\|\boldsymbol{S}^{(k)}\|_2 \tag{2.6}$$

are satisfied. Suppose, furthermore, that we terminate the inner minimization at a point $\boldsymbol{y}^{(k)}$ for which

$$f_s^{(k)}(\mathbf{0}) - f_s^{(k)}(\boldsymbol{y}^{(k)}) \geq \tau(f_s^{(k)}(\mathbf{0}) - f_s^{(k)}(\boldsymbol{y}_s^{(k)})), \tag{2.7}$$

where $\tau > 0$. Then it is easy to show that this scheme is globally convergent under the same conditions as stated above. The advantage here is that the tests (2.5) and (2.6) need only be satisfied at an intermediate point to ensure convergence. Typically, the first inner-iterate provides such a point for carefully chosen subspaces and minimizers. For example, if the subspace contains the steepest-descent direction and the inner minimization starts by performing a linesearch in this direction, the resulting first inner-iterate satisfies (2.5) and (2.6). Similarly, if the subspace contains a (modified) truncated-Newton direction and the inner minimization starts by performing a linesearch in this direction, the same conclusion is true.

3 Computational Variants

We consider it important from a practical point of view to require that $\boldsymbol{S}^{(k)}$ contains at least two components,

- a gradient-related direction, such as $-\boldsymbol{g}^{(k)}$, to encourage global convergence, and
- a Newton-related direction, such as might be computed by a truncated-Newton method, to encourage fast asymptotic convergence, with safeguards to account for indefiniteness.

Of course, these components play a key role in dog-leg trust-region methods (see, for example, Powell, 1970). Additional components have the advantage of enlarging the subspace searched, but the disadvantage of increasing the overheads in solving the $s^{(k)}$-dimensional subspace minimization problem. In this section, we consider various possible ways of choosing the iterated subspace.

3.1 Conjugate gradients

An appealing choice of $\boldsymbol{S}^{(k)}$ may be obtained by picking the columns of $\boldsymbol{S}^{(k)}$ as a set of $\boldsymbol{H}^{(k)}$-conjugate directions, especially if these directions are generated by a (preconditioned) conjugate-gradient (CG) method.

Suppose that $\boldsymbol{H}^{(k)}$ is positive definite. Let $\phi^{(k)}(\boldsymbol{x}^{(k)} + \boldsymbol{p})$ be the quadratic model,

$$\phi^{(k)}(\boldsymbol{x}^{(k)} + \boldsymbol{p}) = f^{(k)} + \boldsymbol{p}^T\boldsymbol{g}^{(k)} + \tfrac{1}{2}\boldsymbol{p}^T\boldsymbol{H}^{(k)}\boldsymbol{p}, \tag{3.1}$$

of $f(\boldsymbol{x}^{(k)} + \boldsymbol{p})$ about $\boldsymbol{x}^{(k)}$. The preconditioned conjugate-gradient method (see, for example, Hestenes and Stiefel, 1952 and Golub and Loan, 1989, Section 10.3) is an iterative method which may be used to calculate the smallest value of (3.1). It is well known that the solution, $\boldsymbol{p}_n$, to this problem is the Newton direction.

A *preconditioner* $\boldsymbol{P}^{(k)}$ is usually an easily invertible approximation to $\boldsymbol{H}^{(k)}$. We shall insist that $\boldsymbol{P}^{(k)}$ has a uniformly bounded condition number. The j-th step of the preconditioned conjugate-gradient method determines the smallest value of (3.1) in the Krylov subspace spanned by the vectors $\{-(\boldsymbol{P}^{(k)-1}\boldsymbol{H}^{(k)})^i \boldsymbol{P}^{(k)-1}\boldsymbol{g}^{(k)}\}_{i=0}^{j}$. Conjugacy properties ensure that each successive step may be accomplished by a univariate minimization of (3.1) in the direction $\boldsymbol{s}_j$; the vectors $\{\boldsymbol{s}_j\}$ are conjugate and are recurred from step to step. Significantly from our point of view, the first such vector, $\boldsymbol{s}_0 = -\boldsymbol{P}^{(k)-1}\boldsymbol{g}^{(k)}$. In exact arithmetic the method would terminate with the Newton direction, $\boldsymbol{p}_n$, after at most n steps, but numerical rounding errors ensure that the method behaves more like an infinite iteration (see Reid, 1971). Moreover, for the large-scale case, we would be unwilling to consider anywhere close to n iterations. Nonetheless, the method is an effective technique for calculating approximations to the Newton direction, especially if a good preconditioner is used or if low accuracy solutions may be tolerated (see Toint, 1981, Dembo *et al.*, 1982, and Dembo and Steihaug, 1983).

In truncated-Newton methods, (see Dembo *et al.*, 1982), the method of conjugate gradients is used to generate approximations to the Newton direction. The resulting search direction, $\boldsymbol{p}_{tn}$, is employed within a linesearch framework for solving unconstrained optimization problems. Significantly, highly accurate approximations to the Newton correction are only needed to accelerate the convergence of the iteration in the neighbourhood of a limit point, and crude improvements upon the steepest-descent direction suffice elsewhere. Furthermore, by monitoring the gradient of the model at each step of the conjugate-gradient method, we can decide when to terminate the iteration.

While such an approach has undoubtedly proved successful in practice, we note that a considerable amount of work is invested, in such a scheme, in calculating an "average" direction and that much of the information gleaned on the way is subsequently ignored. We take the point of view that directions generated by the conjugate-gradient method are of interest for the quadratic model, but might also be locally of interest for the true objective function. We thus propose to construct our iterated subspace from the subspace investigated by the conjugate gradient method.

We intend to include some or all of the following:

- The preconditioned steepest-descent direction, $\boldsymbol{s}_0 = -\boldsymbol{P}^{(k)-1}\boldsymbol{g}^{(k)}$;
- A number of other conjugate directions, $\boldsymbol{s}_j$, determined by the preconditioned conjugate-gradient method; and
- The overall truncated-Newton direction, $\boldsymbol{p}_{tn}$.

We note that the first of these components is designed to encourage global convergence, while the last will ensure that convergence occurs at a fast asymptotic rate.

3.2 Choice of conjugate directions

Suppose that, in addition to the (preconditioned) steepest-descent and (truncated) Newton directions, we wish to include q $\boldsymbol{H}^{(k)}$-conjugate directions in the subspace. The simplest choice is just to take the first q generated (excluding, of course, the steepest-descent direction). However, these may not be those which were most judicious for the quadratic model and another choice may be to take the q which gave the largest decrease in the model. Experiments suggest that this is rarely more successful than the simpler scheme.

Another possibility is to consider including approximations from the extreme eigenspaces, that is the set of eigenvectors which correspond to the smallest and largest eigenvalues (recall $\boldsymbol{H}^{(k)}$ is assumed positive definite). Eigenvectors corresponding to large eigenvalues may be useful as the objective function and CG model differ most significantly in these directions. Those associated with small eigenvalues reflect the space in which the Newton direction is likely to be sensitive and contributions from this space are necessary if rapid progress is to be made. Thus both sets of vectors are reasonable candidates for subspace directions.

Clearly, the calculation of these spaces is generally prohibitively expensive, but they may be approximated by directions generated during the CG process (see, eg, Parlett, 1980, Chapter 13). We might monitor the Rayleigh quotients

$$\frac{\boldsymbol{s}_j^T \boldsymbol{H}^{(k)} \boldsymbol{s}_j}{\boldsymbol{s}_j^T \boldsymbol{s}_j} \tag{3.2}$$

and include the $\boldsymbol{s}_j$ which give rise to the most extreme Rayleigh quotients. Of course, these vectors are not eigenvectors of $\boldsymbol{H}^{(k)}$, but they normally contain significant contributions in the extreme eigenspaces.

3.3 Choice of subspace dimension

The choice of subspace dimension is clearly important. The simplest choice is to fix an upper bound s on this dimension before the computation proceeds (perhaps $s = 10$, see Section 4), and to select $s^{(k)}$ to be the smaller of s and the total number of CG directions sampled during the k-th CG iteration — recall that the CG process may be truncated and thus fewer than s directions may have been computed.

A more sophisticated approach is to try to dynamically select the size of subspace based upon the needs of the k-th iteration. For instance, if the Hessian is relatively well-conditioned, it is likely that a subspace made up from the steepest-descent and (truncated) Newton directions will suffice. If, on the other hand, the Hessian is ill-conditioned, further subspace directions are likely to prove beneficial.

A simple heuristic would be to monitor the Rayleigh quotient as the CG iteration proceeds. Typically the first search direction, $\boldsymbol{s}_0 = -\boldsymbol{P}^{(k)-1}\boldsymbol{g}^{(k)}$, for the CG iteration will contain components of all eigenvectors and hence some components of those corresponding to the largest eigenvalues. The influence of the eigenvectors corresponding to the large eigenvalues is reduced in the subsequent directions $\boldsymbol{s}_1, \ldots$, and this is reflected in a reduction in the Rayleigh quotient during these iterations. This effect is reversed after a number of iterations, when the influence of the larger eigenvalues reappears. It would thus seem sensible to record the iteration number, $i^{(k)}$, at which the Rayleigh quotient first starts to increase after its initial sequence of decreases. As we know that we then have sampled eigenvectors in both "large" and "small" eigenspaces, it is appropriate to set $s^{(k)} = i^{(k)}$.

3.4 The inner minimization

As we have stated, we believe that there are a number of highly effective algorithms for the unconstrained minimization of a function of several variables. Indeed, it would not be unreasonable to say that the problem has effectively been solved so long as derivatives are available. Among the most successful methods are the Newton-like second-derivative methods and the finite-difference and secant methods which require only gradients (see, for example, Dennis and Schnabel, 1983, Gill *et al.*, 1981 or Fletcher, 1987).

When considering the minimization of (1.4), we note that the calculation of derivatives of $f_s^{(k)}$ requires those of f. We see that the calculation of the second derivatives (1.6) requires significantly more products involving $\boldsymbol{S}^{(k)}$ than do the first derivatives (1.5). Thus, we would prefer to use methods which either only require relatively few Hessian-vector products, such as (preconditioned and truncated) conjugate-gradient methods, or secant methods, which build up approximations to the second derivatives from gradients in the $s^{(k)}$-dimensional subspace as they proceed.

The most widely used secant methods are those in the convex Broyden class of positive-definite approximations, of which the BFGS method has the best reputation (again see, for example, Dennis and Schnabel, 1983). While there is currently some controversy as to whether there are better nonconvex secant updates (see for example, Conn *et al.*, 1991, Khalfan *et al.*, 1993, and Byrd *et al.*, 1993), we feel that convex secant methods are most natural in a linesearch, as opposed to a trust-region, context. Such methods start with a positive-definite second-derivative approximation and generate a sequence of matrices which mimic the curvature in the space of directions searched. Traditionally, the Cholesky factors of the sequence are updated as the iteration proceeds. We now show that building a good starting matrix in our case is easy.

Firstly, suppose that $\boldsymbol{S}^{(k)}$ is made up purely of $\boldsymbol{H}^{(k)}$-conjugate directions. Then the exact second-derivative matrix $\boldsymbol{H}_s^{(k)}$ is diagonal because of the conjugacy and moreover its diagonal entries will have been calculated during the conjugate-gradient process. This matrix, then, is a good starting approximation; its Cholesky factors are trivial to determine. Furthermore, this choice ensures that the first Quasi-Newton search direction is identical to that generated by minimizing the quadratic model (3.1) in the manifold $\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y}$ (see, Gill *et al.*, 1981, section 4.8.3.1).

Now suppose that $\boldsymbol{S}^{(k)}$ is made by augmenting a set of $\boldsymbol{H}^{(k)}$-conjugate directions by the overall truncated-Newton direction $\boldsymbol{p}_{tn}$. Then, the exact second-derivative matrix has an arrowhead structure with the leading $s^{(k)} - 1$ by $s^{(k)} - 1$ submatrix being diagonal and the remaining row and column easy to obtain. To be precise, if we denote the residual $\boldsymbol{H}^{(k)}\boldsymbol{p}_{tn} + \boldsymbol{g}^{(k)}$ following the truncated conjugate-gradient process by $\boldsymbol{r}^{(k)}$, the last column of the required second-derivative matrix is $\boldsymbol{S}^{(k)T}(\boldsymbol{r}^{(k)} - \boldsymbol{g}^{(k)})$. Thus, once again this matrix provides a good starting approximation in that its Cholesky factors are extremely cheap to compute. Moreover, as before, the first Quasi-Newton direction gives the minimum of the model (3.1) in the manifold $\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y}$. Significantly, as the truncated-Newton direction is in the subspace, this first Quasi-Newton direction is thus the same as the truncated-Newton direction.

4 Numerical Experiments

We start this section by investigating, from a numerical point of view, the impact of different subspace sizes on the convergence of the method. As the problem FMINSURF[1] is particularly efficiently solved using ISM in comparison with the default version of LANCELOT, we examined this problem in detail. We ran ISM, without preconditioner, but constructing the subspace from a combination of the steepest-descent, truncated-Newton and extreme directions as described above, using a variety of subspace dimensions and illustrate, in Figure 4.1, the effects of the choice of this dimension on the CPU time[2] required to solve the problem.

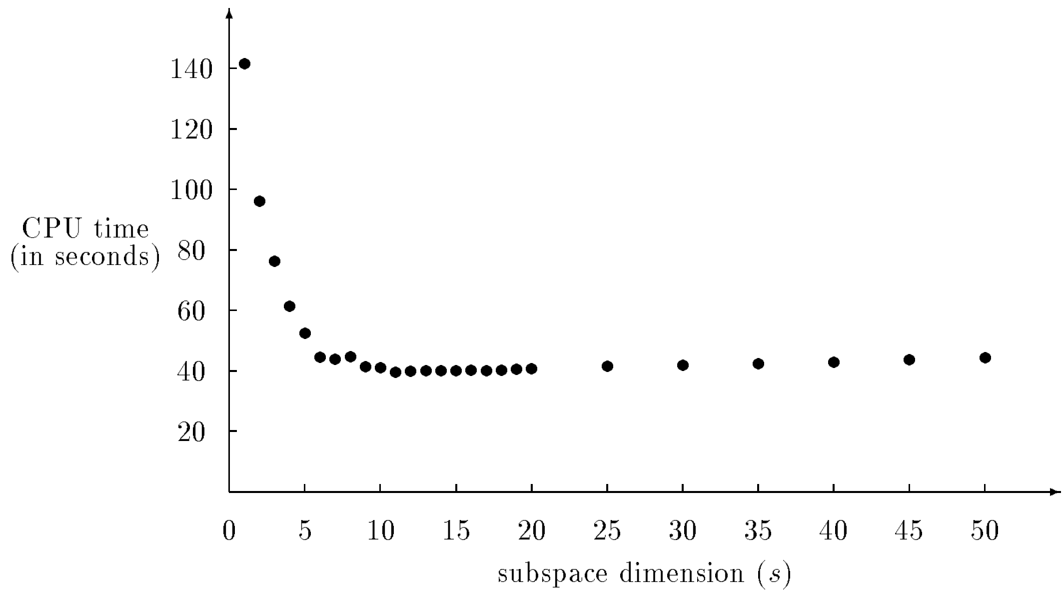

Figure 4.1: The impact of varying the subspace dimension on FMINSURF (using an unpreconditioned ISM in which the subspace is chosen from the the steepest-descent, truncated-Newton and extreme CG directions).

We observe that the CPU time for small subspace dimension is large, but that, for subspaces of dimension between 9 and 40, the time is relatively constant, being within ten percent of the least ($s = 11$) time. Thus, it appears that for this problem, adding information above the steepest-descent and truncated-Newton directions is beneficial but there is little extra payoff from using more than 11 directions. When we monitored the Rayleigh quotients for this problem, we observed that the quotient decreases for on average 10 CG iterations before increasing for the first time. Thus, unless we insist on at least 10 CG iterations, it is possible that we may not have

[1] From the CUTE collection, see Bongartz *et al.* (1993).

[2] This experiment was performed on an IBM RISC/6000 320h workstation, using optimized (-O) Fortran 77 code and IBM-supplied BLAS.

sampled the complete eigenspace. Similar runs on different problems indicate that this behaviour is quite typical.

We next report the results of running a few variants of our Iterated-Subspace Minimization code on some large or difficult test problems. The simplest variant has the following features.

- The iteration is deemed to have converged when $\|\boldsymbol{g}^{(k)}\|_2$ is smaller than 10^{-5}.
- The subspace is constructed from the first $s^{(k)} - 1$ conjugate directions plus the truncated-Newton direction. The truncation is performed when the residual (gradient) of the model is smaller than $\|\boldsymbol{g}^{(k)}\|_2 \min(0.1, \|\boldsymbol{g}^{(k)}\|_2^{0.1})$ or when more than n conjugate-gradient iterations were performed.
- $s^{(k)}$ is the smaller of 10 (as suggested by the above example) and the number of inner iterations required to determine the truncated-Newton direction.
- The model is modified, if necessary, to ensure that it is strictly convex. The modification is carried out as the conjugate-gradient iteration proceeds using the method of Arioli *et al.* (1993).
- A BFGS linesearch method is used to solve the inner-minimization problem. An Armijo backtracking linesearch is used, starting with a unit step and dividing the step by two until the Armijo sufficient decrease condition is satisfied. If a step of one proves acceptable, but the model has been modified to ensure that it is strictly convex, the step is doubled until an unacceptable stepsize is determined, when the last-found acceptable step is chosen. A maximum of $2\, s^{(k)}$ BFGS iterations are permitted and the iteration is stopped if $\|\boldsymbol{g}_s^{(k)}\|_2$ is smaller than 10^{-6}.

Note that one Hessian evaluation is made for each minimization and one gradient evaluation for each inner iteration. We denote this method by the symbol ISM(n,f,f), where n means that *no* preconditioning is used, the first f that the subspace is constructed from the *first* CG directions, and the second f that the maximal subspace dimension s is *fixed* (to 10).

We also consider the ISM(p,f,f) method, which is identical to ISM(n,f,f), except that an 11-band modified Cholesky factorization *preconditioner* is used in the conjugate-gradient calculation. The factorization takes the elements of $\boldsymbol{H}^{(k)}$ within a band of semi-bandwidth 5 of the diagonal, replacing any other elements by zeros. The resulting band matrix is factorized, modifications being made according to the recipe of Schnabel and Eskow (1991) to ensure that the preconditioner is positive definite with bounded condition number.

We define the methods ISM(n,e,f) and ISM(p,e,f) as the modifications of ISM(n,f,f) and ISM(p,f,f), respectively, where we construct the subspace from a set of *extreme* $s^{(k)} - 2$ conjugate directions plus the steepest-descent and truncated-Newton directions. We pick half of the extreme directions to be those whose Rayleigh quotient is largest, while the remainder correspond to the smallest Rayleigh quotients.

We next consider the possibility of generating the subspace dimension *automatically*, as discussed in Section 3.3. Once the subspace dimension has been determined, we construct the subspace from the set of first or extreme $s^{(k)} - 2$ conjugate directions plus the steepest-descent and truncated-Newton direction, just as before. This results in four additional variants, namely ISM(n,f,a), ISM(p,f,a), ISM(n,e,a) and ISM(p,e,a).

As a yard-stick, we compare the above methods with variants on two other algorithms. The first is the default version of the LANCELOT A nonlinear optimization package (see Conn *et al.*, 1992b) (denoted LAN(p)) in which an 11-band preconditioner is used, together with the same unpreconditioned algorithm (denoted LAN(n)). LANCELOT is a trust-region method in which a gradient and Hessian evaluation are made on every successful iteration. The second algorithm included in our comparison is a truncated-Newton method (see Dembo and Steihaug (1983)). The truncated-Newton search direction is obtained by an inexact minimization of the Newton model using unpreconditioned or preconditioned conjugate gradients. We denote the resulting methods by TN(n) and TN(p), respectively. These variants are obtained from our ISM algorithms by restricting the subspace minimization to a single linesearch along the truncated-Newton direction.

We note that all the algorithms considered in this comparison use exact first and second derivatives.

We selected our 34 test examples as the majority of large and/or difficult unconstrained test examples in the CUTE (see Bongartz *et al.*, 1993) test set. Only problems which took excessive CPU time (more than 30 minutes), which had multiple local optima, or which were variations on the reported problems, were excluded. All experiments were made on a DEC 3000/400 workstation, using optimized (-O) Fortran 77 code and DEC-supplied BLAS.

method	details	#f	#cg	time
LAN(n)	Table A.3	5552	33923	488
TN(n)	Table A.5	17048	41050	614
ISM(n,f,f)	Table A.7	21988	31919	809
ISM(n,e,f)	Table A.9	21243	28793	801
ISM(n,f,a)	Table A.11	15547	32235	611
ISM(n,e,a)	Table A.13	15263	32302	605
LAN(p)	Table A.4	4144	5577	497
TN(p)	Table A.6	7452	10280	489
ISM(p,f,f)	Table A.8	12103	7165	516
ISM(p,e,f)	Table A.10	12333	7604	527
ISM(p,f,a)	Table A.12	12159	8073	539
ISM(p,e,a)	Table A.14	12075	7741	524

Table 4.1: Cumulative statistics on the performance of all methods

Table 4.1 first reports cumulative statistics on the performance of the considered algorithms, which all succeeded in solving the 34 problems. In this table and the following ones, *#f* indicates the total number of function evaluations, *#cg* the total number of CG iterations and *time* the total CPU time (in seconds). The second column indicates which of the tables in the appendix gives the complete and detailed results for the considered method.

This table shows some interesting results. In particular, it indicates that the automatic choice of the subspace dimension is advantageous on average when preconditioning is not used. A possible explanation is that the automatic subspace selection is more useful if the eigenvalues of the Hessian are not well clustered: the CG procedure might then need more iterations to include the contributions of all relevant extreme eigenvalues. On the other hand, one sees that, on average, the preconditioned ISM variants are all better in CPU time than their unpreconditioned counterparts. Also, there is little difference between the average performance of ISM methods using the first CG directions to define the subspace and those using the extreme ones. It is however misleading in that it seems to suggest that LANCELOT (and, to some extent, truncated-Newton) dominate all ISM methods. Although obviously true on average, this conclusion is exaggerated because of the aggregate nature of the total sums presented in the table. A more disaggregate analysis of what happens problem by problem is thus needed to reveal more specific trends. We thus present in Table 4.2 the number of times that each of the considered method ranks first, second, third, etc. for the different criteria used above. In these rankings, two CPU times are reputed identical if they differ by less than five percent or by less than half a second.

We see in this table that the supremacy of LANCELOT and truncated-Newton in CPU time is not so clear, especially for the preconditioned variants. We also note that the amount of requested CG iterations by the ISM variants is comparable on average to that for LANCELOT, and quite often smaller, but the resulting gain in time is then sometimes lost due to the larger number of objective function evaluations. One should also bear in mind, at this point, that LANCELOT is a much more sophisticated code than our ISM variants, because it is designed for solving generally constrained in addition to unconstrained problems and contains a number of safeguards that are not included in the simpler ISM codes. Moreover, there are overheads associated with the interfaces with the SIF input in the ISM codes that are not present in LANCELOT. However, the streamlined character of the ISM variants may be considered to be one of their advantages.

Another interesting observation is that the truncated-Newton methods appear to require more CG iterations than the other methods, and yet their requirements in CPU time is not excessive: this can be explained by the fact that they do not require too many function evaluations and contain little other computational effort.

Finally, we notice that there is little difference in performance between ISM variants that build the subspace from the first CG directions and variants that use the extreme ones. As the former is easier to implement, one might prefer it in practice.

method	Unpreconditioned (n)						Preconditioned (p)					
	1st	2nd	3rd	4th	5th	6th	1st	2nd	3rd	4th	5th	6th
	Function evaluations (#f)											
LAN(.)	31	2	0	1	0	0	33	0	0	0	0	0
TN(.)	1	19	7	5	2	0	0	23	9	0	0	0
ISM(.,f,f)	1	13	9	5	5	1	0	14	14	4	1	1
ISM(.,e,f)	2	12	8	3	5	4	0	14	14	1	2	3
ISM(.,f,a)	1	15	13	3	1	1	0	15	15	3	1	0
ISM(.,e,a)	1	14	14	5	0	0	1	14	13	4	2	0
	CG iterations (#cg)											
LAN(.)	14	4	9	1	4	2	14	10	7	2	0	1
TN(.)	1	10	9	6	4	4	8	8	12	2	1	3
ISM(.,f,f)	16	11	2	2	2	1	21	10	2	0	1	0
ISM(.,e,f)	13	16	3	2	0	0	20	9	2	2	0	1
ISM(.,f,a)	13	12	4	3	2	0	16	9	4	2	2	1
ISM(.,e,a)	11	13	5	4	1	0	16	10	4	1	3	0
	Time											
LAN(.)	23	11	2	0	1	1	19	11	2	0	1	1
TN(.)	18	9	4	2	0	1	21	6	3	4	0	0
ISM(.,f,f)	20	5	5	3	0	1	23	7	3	1	0	0
ISM(.,e,f)	20	4	6	3	1	0	21	9	3	1	0	0
ISM(.,f,a)	22	6	5	0	1	0	19	8	5	1	1	0
ISM(.,e,a)	21	8	4	1	0	0	20	5	8	1	0	0

Table 4.2: Rankings

5 Linear and nonlinear constraints

It is straightforward to extend the iterated-subspace minimization philosophy to treat linearly constrained optimization problems of the form

$$\underset{\boldsymbol{x}\in\Re^n}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{l} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{u}, \tag{5.1}$$

where $\boldsymbol{A}$ is an m by n matrix and $\boldsymbol{l}$ and $\boldsymbol{u}$ are m-vectors. For suppose $\boldsymbol{x}^{(k)}$ satisfies $\boldsymbol{l} \leq \boldsymbol{A}\boldsymbol{x}^{(k)} \leq \boldsymbol{u}$. Then we may obtain an improved estimate $\boldsymbol{x}^{(k+1)}$ by applying the following linearly-constrained ISM algorithm:

1. Stop with the solution estimate $\boldsymbol{x}^{(k)}$ if convergence tests are satisfied.
2. Determine a full-rank subspace matrix $\boldsymbol{S}^{(k)} \in \Re^{n\times s^{(k)}}$, where $s^{(k)} \ll n$.
3. Approximately solve the $s^{(k)}$-dimensional minimization problem

$$\underset{\boldsymbol{y}\in\Re^{s^{(k)}}}{\text{minimize}} \quad f(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y}) \quad \text{subject to} \quad \boldsymbol{l}^{(k)} \leq \boldsymbol{A}^{(k)}\boldsymbol{y} \leq \boldsymbol{u}^{(k)}, \tag{5.2}$$

where $\boldsymbol{A}^{(k)} = \boldsymbol{A}\boldsymbol{S}^{(k)}$, $\boldsymbol{l}^{(k)} = \boldsymbol{l} - \boldsymbol{A}\boldsymbol{x}^{(k)}$ and $\boldsymbol{u}^{(k)} = \boldsymbol{u} - \boldsymbol{A}\boldsymbol{x}^{(k)}$, and set

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{y}\in\Re^{s^{(k)}}}{\text{(approx) arg min}} \quad f(\boldsymbol{x}^{(k)} + \boldsymbol{S}^{(k)}\boldsymbol{y}) \quad \text{subject to} \quad \boldsymbol{l}^{(k)} \leq \boldsymbol{A}^{(k)}\boldsymbol{y} \leq \boldsymbol{u}^{(k)}. \tag{5.3}$$

The central issues remain those discussed in Section 1. However, extra care must be exercised when picking the subspace matrix, as it is now desirable for a *constrained* steepest-descent and (truncated) Newton directions to lie in the subspace. We also now need to use efficient methods for solving *small linearly-constrained minimization* problems when determining $\boldsymbol{x}^{(k+1)}$, but fortunately the state-of-the-art here is as advanced as it is for unconstrained minimization.

The ISM philosophy does not obviously extend to handle nonlinearly-constrained minimization problems except that, of course, any unconstrained or linearly-constrained subproblems may be treated by existing ISM methods. This may be important for nonlinearly-constrained minimization methods which are based on the sequential minimization of penalty or barrier functions, or their augmented or shifted counterparts.

6 Perspectives and Conclusions

In this paper, we have shown that it is possible to solve large-scale nonlinear optimization problems using methods designed for small-scale problems. These methods may be regarded as a generalization of linesearch-type methods more usually used to solve unconstrained minimization problems. We have indicated that the convergence of our methods depends upon using robust algorithms for the small-dimensional subproblems, and have suggested a number of ways of selecting promising subspaces in which to search. Furthermore, our philosophy extends quite naturally to large-scale linearly-constrained optimization. None the less, we feel that there are a number of important areas for future investigation.

- In our investigations, we found it convenient to use a linesearch (BFGS) method to solve the inner-iteration subproblems. One might, of course, alternatively use a trust-region method to solve the subproblem. However, as the performance of such methods depends upon building a good model within an adequate trust region, and as our ISM method will solve a sequence of subproblems, it may be that a good trust-region radius for one subproblem is poor for the next, and inefficiencies may occur. Thus care may be needed in determining interactions between successive models.

  Another important issue is how to pick stopping rules, analogous to (2.5)–(2.7), which are appropriate for trust-region based methods. The main difficulty here is that the initial trust-region radius may interfere with a condition like (2.5).

- While we have suggested a number of methods for computing a good iterated subspace, more work clearly needs to be performed. We believe that we have identified some of the ingredients of a good subspace, but our understanding is far from complete.

- We have suggested that ISM methods are equally appropriate for linearly-constrained problems, but have not addressed the issues related to finding a good iterated subspace in this case. Work progresses in this direction, for both general linear constraints and for problems whose constraints arise from networks.

7 Acknowledgement

Nick Gould would like to thank CERFACS for the facilities which made some of this research possible.

References

[Arioli *et al.*, 1993] M. Arioli, T. F. Chan, I. S. Duff, N. I. M. Gould, and J. K. Reid. Computing a search direction for large-scale linearly constrained nonlinear optimization calculations. Technical Report TR/PA/93/34, CERFACS, Toulouse, France, 1993.

[Armijo, 1966] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[Bertsekas, 1982] D. P. Bertsekas. *Constrained Optimization and Lagrange Multipliers Methods*. Academic Press, London, 1982.

[Bongartz *et al.*, 1993] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. Technical Report TR/PA/93/10, CERFACS, Toulouse, France, 1993. to appear in ACM Transactions on Mathematical Software.

[Byrd *et al.*, 1993] R. H. Byrd, H. F. Khalfan, and R. B. Schnabel. Analysis of a symmetric rank-one trust region method. Technical Report CU-CS-657-93, Department of Computer Science, University of Colorado at Boulder, Boulder, USA, 1993.

[Conn *et al.*, 1991] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(2):177–196, 1991.

[Conn *et al.*, 1992a] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Intensive numerical tests with LANCELOT (Release A): the complete results. Technical Report 92/15, Department of Mathematics, FUNDP, Namur, Belgium, 1992.

[Conn *et al.*, 1992b] 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.

[Conn *et al.*, 1992c] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization. Technical Report 92-075, Rutherford Appleton Laboratory, Chilton, England, 1992.

[Dembo and Steihaug, 1983] R. S. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.

[Dembo *et al.*, 1982] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact-Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[Dennis and Schnabel, 1983] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice-Hall, Englewood Cliffs, USA, 1983.

[Fletcher, 1987] R. Fletcher. *Practical Methods of Optimization.* J. Wiley and Sons, Chichester, second edition, 1987.

[Gill *et al.*, 1981] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London and New York, 1981.

[Goldstein, 1964] A. A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70:709–710, 1964.

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

[Hestenes and Stiefel, 1952] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. N. B. S.*, 49:409–436, 1952.

[Khalfan *et al.*, 1993] H. F. Khalfan, R. H. Byrd, and R. B. Schnabel. A theoretical and experimental study of the symmetric rank-one update. *SIAM Journal on Optimization*, 3(1):1–24, 1993.

[Parlett, 1980] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, USA, 1980.

[Powell, 1970] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Method for Nonlinear Algebraic Equations*, pages 87–114, London, 1970. Gordon and Breach.

[Reid, 1971] J. K. Reid. On the method of conjugate gradients for the solution of large sparse linear equations. In J. K. Reid, editor, *Large sparse sets of linear equations*, pages 231–254, London, 1971. Academic Press.

[Saad, 1990] Y. Saad. Krylov subspace methods: theory, algorithms and applications. In *Computational Methods in Applied Science and Engineering, INRIA 29 January - 2 February*, 1990.

[Schnabel and Eskow, 1991] R. B. Schnabel and E. Eskow. A new modified Cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, 11:1136–1158, 1991.

[Toint, 1981] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, London, 1981. Academic Press.

A Detailed numerical results

In this appendix, we give comprehensive details of the performance of each of the methods discussed in the main body of the paper on the complete set of test examples. We include these results so that others may, in future, compare new proposals with ours.

problem	n	#f	#suc	#cg	time	f
ARWHEAD	1000	5	6	2	0.65	1.6903D-10
BDQRTIC	1000	12	13	66	1.77	3.9838D+03
BROWNAL	100	2	3	3	0.13	1.1263D-13
BRYBND	1000	13	13	177	3.50	2.8797D-13
CRAGGLVY	1000	13	14	166	2.28	3.3642D+02
DIXMAANA	1500	5	6	9	0.84	1.0000D+00
DIXON3DQ	1000	5	6	1427	5.32	7.7032D-09
DQDRTIC	1000	2	3	0	0.43	1.6602D-23
DQRTIC	1000	35	36	223	2.63	3.7687D-06
EDENSCH	1000	14	15	35	1.24	6.0033D+03
EIGENALS	110	15	16	142	0.71	2.7564D-11
ENGVAL1	1000	7	8	20	0.76	1.1082D+03
FLETCHCR	1000	3528	2635	20619	217.64	5.0581D-14
FMINSURF	1024	189	166	527	16.36	1.0000D+00
FREUROTH	1000	20	10	40	1.48	1.2147D+05
GENROSE	1000	1213	927	6640	77.29	1.0000D+00
LIARWHD	1000	13	14	14	0.93	2.3011D-16
MANCINO	100	11	12	13	12.87	4.3367D-17
MOREBV	1000	2	3	265	1.91	2.0186D-09
NCB20B	1000	27	22	1274	102.76	1.6760D+03
NONDIA	1000	29	27	29	1.68	4.3483D-16
NONDQUAR	1000	91	75	853	6.00	9.5606D-06
PENALTY1	1000	55	51	44	9.44	9.6862D-03
POWELLSG	1000	15	16	60	1.01	1.1167D-06
POWER	1000	28	29	613	3.60	5.8399D-09
QUARTC	1000	35	36	223	2.67	3.7687D-06
SINQUAD	1000	69	61	126	4.98	6.3238D-05
SROSENBR	1000	10	10	20	0.71	6.6723D-12
TOINTGSS	1000	8	8	15	0.60	1.0010D+01
TQUARTIC	1000	13	13	14	0.92	1.0168D-16
TRIDIA	1000	4	5	91	0.81	8.5487D-14
VARDIM	1000	36	37	0	1.41	1.1203D-20
VAREIGVL	1000	12	13	124	2.12	5.8152D-08
WOODS	1000	16	17	49	1.11	5.0956D-15

Table A.3: Results for LAN(n) (unpreconditioned LANCELOT) on large or hard problems. Key: n = number of variables, $\#f$ = number of function evaluations, $\#suc$ = number of successful iterations, $\#cg$ = total number of CG iterations, *time* = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#suc	#cg	time	f
ARWHEAD	1000	5	6	1	0.79	1.6903D-10
BDQRTIC	1000	11	12	13	1.75	3.9838D+03
BROWNAL	100	3	4	40	1.10	6.3470D-11
BRYBND	1000	16	14	30	2.88	1.8775D-12
CRAGGLVY	1000	14	15	11	1.68	3.3642D+02
DIXMAANA	1500	5	6	10	1.04	1.0000D+00
DIXON3DQ	1000	2	3	2	0.45	0.0000D+00
DQDRTIC	1000	2	3	0	0.45	1.6602D-23
DQRTIC	1000	35	36	27	2.60	3.6952D-06
EDENSCH	1000	12	13	9	1.36	6.0033D+03
EIGENALS	110	17	17	48	0.95	5.2529D-11
ENGVAL1	1000	7	8	7	0.91	1.1082D+03
FLETCHCR	1000	2137	1820	2136	127.72	2.8160D-12
FMINSURF	1024	315	295	436	118.40	1.0000D+00
FREUROTH	1000	10	10	7	1.16	1.2147D+05
GENROSE	1000	1094	906	1158	68.07	1.0000D+00
LIARWHD	1000	14	15	21	1.39	2.0274D-20
MANCINO	100	15	16	8	20.12	9.2449D-18
MOREBV	1000	1	2	1	0.45	7.3289D-13
NCB20B	1000	22	20	568	54.61	1.6760D+03
NONDIA	1000	29	27	47	2.38	6.9910D-16
NONDQUAR	1000	17	18	19	1.34	1.3932D-09
PENALTY1	1000	63	57	432	27.41	9.6862D-03
POWELLSG	1000	15	16	15	1.13	2.2324D-06
POWER	1000	27	28	53	8.56	1.8888D-08
QUARTC	1000	35	36	27	2.58	3.6952D-06
SINQUAD	1000	131	112	341	15.55	9.0365D-07
SROSENBR	1000	10	10	10	0.91	5.8309D-13
TOINTGSS	1000	2	3	2	0.49	1.0000D+01
TQUARTIC	1000	12	12	24	1.30	2.9487D-11
TRIDIA	1000	2	3	1	0.54	1.3827D-32
VARDIM	1000	36	37	0	21.26	1.1203D-20
VAREIGVL	1000	12	13	58	5.12	5.6139D-08
WOODS	1000	16	17	15	1.34	1.9658D-14

Table A.4: Results for LAN(p) (default preconditioned LANCELOT) on large or hard problems. Key: n = number of variables, $\#f$ = number of function evaluations, $\#suc$ = number of successful iterations, $\#cg$ = total number of CG iterations, *time* = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	time	f
ARWHEAD	1000	1545	389	390	395	31.74	0.00D+00
BDQRTIC	1000	44	18	19	96	2.08	3.98D+03
BROWNAL	100	5	3	4	4	0.05	2.45D-08
BRYBND	1000	38	15	16	50	1.82	2.77D-13
CRAGGLVY	1000	45	16	17	111	2.27	3.36D+02
DIXMAANA	1500	17	9	10	13	0.99	1.00D+00
DIXON3DQ	1000	7	4	5	1748	4.16	1.25D-11
DQDRTIC	1000	10	5	6	12	0.28	1.91D-14
DQRTIC	1000	48	17	18	54	0.40	5.72D-08
EDENSCH	1000	46	17	18	38	1.56	6.00D+03
EIGENALS	110	69	23	24	209	1.27	6.15D-12
ENGVAL1	1000	31	13	14	26	0.97	1.11D+03
FLETCHCR	1000	3065	1178	1179	13208	121.19	7.36D-13
FMINSURF	1024	250	33	34	6648	61.67	1.00D+00
FREUROTH	1000	53	18	19	109	2.43	1.21D+05
GENROSE	1000	2933	542	543	9745	88.37	1.00D+00
LIARWHD	1000	78	28	29	32	1.67	1.43D-16
MANCINO	100	74	23	24	30	82.92	9.93D-19
MOREBV	1000	4	3	4	519	2.17	1.37D-09
NCB20B	1000	95	22	23	1126	99.57	1.68D+03
NONDIA	1000	156	55	56	58	4.17	5.93D-21
NONDQUAR	1000	163	63	64	2940	9.18	1.01D-06
PENALTY1	1000	173	60	61	84	2.44	9.69D-03
POWELLSG	1000	838	282	283	577	6.93	5.32D-08
POWER	1000	5636	1414	1415	1419	34.68	4.19D-09
QUARTC	1000	48	17	18	54	0.40	5.72D-08
SINQUAD	1000	271	114	115	198	10.44	4.92D-06
SROSENBR	1000	26	12	13	15	0.35	6.11D-20
TOINTGSS	1000	298	101	102	201	10.42	1.00D+01
TQUARTIC	1000	57	20	21	27	0.89	1.24D-13
TRIDIA	1000	17	10	11	643	1.66	4.80D-15
VARDIM	1000	60	15	16	15	0.41	2.48D-21
VAREIGVL	1000	546	183	184	367	19.89	8.04D-10
WOODS	1000	302	96	97	279	4.64	1.65D-19

Table A.5: Results for the unpreconditioned truncated-Newton method TN(n) on large or hard problems.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	time	f
ARWHEAD	1000	20	8	9	10	0.82	0.00D+00
BDQRTIC	1000	30	11	12	15	1.39	3.98D+03
BROWNAL	100	68	27	28	207	8.32	3.35D-12
BRYBND	1000	72	15	16	39	3.12	8.33D-14
CRAGGLVY	1000	122	42	43	42	5.86	3.36D+02
DIXMAANA	1500	25	11	12	14	1.66	1.00D+00
DIXON3DQ	1000	3	1	2	1	0.06	0.00D+00
DQDRTIC	1000	3	1	2	1	0.11	0.00D+00
DQRTIC	1000	43	15	16	15	0.53	7.86D-08
EDENSCH	1000	42	12	13	13	1.56	6.00D+03
EIGENALS	110	81	20	21	80	1.45	3.04D-13
ENGVAL1	1000	26	10	11	10	1.01	1.11D+03
FLETCHCR	1000	2949	1158	1159	1159	94.00	1.49D-16
FMINSURF	1024	276	33	34	1509	31.66	1.00D+00
FREUROTH	1000	44	8	9	1007	17.97	1.21D+05
GENROSE	1000	2257	726	727	1573	67.58	1.00D+00
LIARWHD	1000	48	16	17	27	1.49	3.51D-15
MANCINO	100	95	24	25	52	102.73	1.50D-18
MOREBV	1000	2	1	2	1	0.10	7.33D-13
NCB20B	1000	47	16	17	498	51.77	1.68D+03
NONDIA	1000	50	19	20	35	1.88	8.47D-20
NONDQUAR	1000	20	9	10	15	0.45	2.31D-09
PENALTY1	1000	113	34	35	169	11.35	9.69D-03
POWELLSG	1000	22	11	12	17	0.49	2.14D-08
POWER	1000	43	15	16	15	4.34	1.00D-10
QUARTC	1000	43	15	16	15	0.52	7.86D-08
SINQUAD	1000	663	257	258	710	37.87	1.99D-08
SROSENBR	1000	36	12	13	13	0.63	8.11D-25
TOINTGSS	1000	3	1	2	1	0.17	1.00D+01
TQUARTIC	1000	25	9	10	16	0.67	6.46D-21
TRIDIA	1000	3	1	2	1	0.07	1.97D-26
VARDIM	1000	76	28	29	2916	31.20	3.77D-13
VAREIGVL	1000	44	13	14	62	5.55	1.42D-09
WOODS	1000	58	21	22	22	1.34	2.55D-18

Table A.6: Results for the preconditioned truncated-Newton method TN(p) on large or hard problems.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	11	3	6	4	1.33	0.32	0.00D+00
BDQRTIC	1000	108	10	45	59	5.30	2.62	3.98D+03
BROWNAL	100	5	2	4	3	1.50	0.05	2.45D-08
BRYBND	1000	68	10	47	62	6.20	3.03	6.16D-14
CRAGGLVY	1000	72	10	56	95	7.30	3.72	3.36D+02
DIXMAANA	1500	19	7	14	9	1.29	1.06	1.00D+00
DIXON3DQ	1000	7	4	5	1748	9.00	4.25	1.25D-11
DQDRTIC	1000	10	5	6	12	2.40	0.28	1.91D-14
DQRTIC	1000	58	9	37	18	2.00	0.45	1.75D-08
EDENSCH	1000	55	9	29	30	3.33	1.70	6.00D+03
EIGENALS	110	88	11	69	99	7.55	1.47	5.29D-12
ENGVAL1	1000	32	8	21	22	2.75	1.02	1.11D+03
FLETCHCR	1000	13009	1046	10628	13164	9.93	371.21	2.83D-14
FMINSURF	1024	169	10	89	2214	10.00	23.28	1.00D+00
FREUROTH	1000	137	8	30	24	3.00	2.94	1.21D+05
GENROSE	1000	5968	418	4380	7038	9.55	163.81	1.00D+00
LIARWHD	1000	18	4	12	6	1.50	0.45	1.14D-13
MANCINO	100	51	12	21	14	1.17	55.05	4.44D-18
MOREBV	1000	4	3	4	519	10.00	2.21	1.37D-09
NCB20B	1000	213	18	98	1181	9.89	127.30	1.68D+03
NONDIA	1000	14	4	10	5	1.25	0.45	1.48D-15
NONDQUAR	1000	539	49	403	4331	9.14	16.97	1.32D-06
PENALTY1	1000	738	126	478	239	1.90	11.44	9.69D-03
POWELLSG	1000	42	5	35	17	3.40	0.44	4.38D-11
POWER	1000	95	11	71	150	6.82	1.30	6.72D-11
QUARTC	1000	58	9	37	18	2.00	0.46	1.75D-08
SINQUAD	1000	193	36	123	69	1.92	6.53	1.97D-07
SROSENBR	1000	16	5	11	7	1.40	0.23	7.59D-16
TOINTGSS	1000	10	5	7	7	1.40	0.49	1.00D+01
TQUARTIC	1000	32	6	17	9	1.50	0.49	1.10D-17
TRIDIA	1000	17	10	11	643	9.80	1.77	7.20D-15
VARDIM	1000	64	13	27	13	1.00	0.49	1.33D-21
VAREIGVL	1000	35	7	29	76	6.86	2.07	1.02D-10
WOODS	1000	33	5	25	14	2.80	0.58	5.80D-16

Table A.7: Results for ISM(n,f,f) on large or hard problems, in which no preconditioning is used and the subspace is chosen from the first 10 CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, *time* = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	21	5	12	6	1.20	0.72	0.00D+00
BDQRTIC	1000	31	7	18	9	1.29	1.19	3.98D+03
BROWNAL	100	71	10	57	125	7.10	3.48	3.13D-12
BRYBND	1000	74	10	34	25	2.50	2.97	9.63D-13
CRAGGLVY	1000	45	12	24	12	1.00	2.12	3.36D+02
DIXMAANA	1500	18	6	13	8	1.33	1.20	1.00D+00
DIXON3DQ	1000	3	1	2	1	1.00	0.07	0.00D+00
DQDRTIC	1000	3	1	2	1	1.00	0.12	1.24D-21
DQRTIC	1000	49	12	25	12	1.00	0.53	2.68D-08
EDENSCH	1000	62	10	24	12	1.20	1.93	6.00D+03
EIGENALS	110	130	14	77	55	3.79	1.98	1.30D-11
ENGVAL1	1000	26	7	14	7	1.00	0.91	1.11D+03
FLETCHCR	1000	6376	1797	3594	1797	1.00	189.01	9.70D-16
FMINSURF	1024	249	9	87	468	9.44	12.92	1.00D+00
FREUROTH	1000	102	8	18	9	1.13	2.31	1.21D+05
GENROSE	1000	3659	656	2021	1011	1.54	93.51	1.00D+00
LIARWHD	1000	25	5	17	12	2.40	0.79	9.91D-16
MANCINO	100	48	14	26	16	1.14	64.06	1.07D-18
MOREBV	1000	2	1	2	1	1.00	0.10	7.33D-13
NCB20B	1000	110	15	69	611	6.67	75.08	1.68D+03
NONDIA	1000	156	25	113	58	2.32	5.26	3.05D-14
NONDQUAR	1000	39	7	29	15	2.14	0.61	2.77D-09
PENALTY1	1000	166	17	102	102	3.24	7.37	9.69D-03
POWELLSG	1000	50	13	30	15	1.15	0.71	5.05D-09
POWER	1000	47	11	23	11	1.00	3.30	4.44D-10
QUARTC	1000	49	12	25	12	1.00	0.53	2.68D-08
SINQUAD	1000	263	38	182	106	2.79	10.81	3.82D-09
SROSENBR	1000	22	8	15	8	1.00	0.48	4.99D-28
TOINTGSS	1000	3	1	2	1	1.00	0.18	1.00D+01
TQUARTIC	1000	21	5	16	8	1.60	0.55	2.45D-14
TRIDIA	1000	3	1	2	1	1.00	0.07	1.01D-25
VARDIM	1000	86	22	49	2585	5.00	26.40	2.09D-12
VAREIGVL	1000	43	7	39	31	4.43	4.13	1.86D-12
WOODS	1000	51	13	28	14	1.08	1.11	1.83D-16

Table A.8: Results for ISM(p,f,f) on large or hard problems, in which preconditioning is used and the subspace is chosen from the first 10 CG directions.

Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	11	3	6	4	1.33	0.32	0.00D+00
BDQRTIC	1000	189	11	51	73	5.82	3.58	3.98D+03
BROWNAL	100	5	2	4	3	1.50	0.05	2.45D-08
BRYBND	1000	68	10	47	62	6.20	3.03	6.16D-14
CRAGGLVY	1000	96	11	58	104	7.55	4.00	3.36D+02
DIXMAANA	1500	19	7	14	9	1.29	1.06	1.00D+00
DIXON3DQ	1000	7	4	5	1748	9.00	4.36	1.25D-11
DQDRTIC	1000	10	5	6	12	2.40	0.29	1.91D-14
DQRTIC	1000	58	9	37	18	2.00	0.46	1.75D-08
EDENSCH	1000	55	9	29	30	3.33	1.70	6.00D+03
EIGENALS	110	102	12	83	110	7.50	1.71	9.57D-11
ENGVAL1	1000	32	8	21	22	2.75	1.04	1.11D+03
FLETCHCR	1000	11901	1079	9821	13262	9.93	352.99	3.73D-16
FMINSURF	1024	158	9	79	1879	10.00	20.05	1.00D+00
FREUROTH	1000	137	8	30	24	3.00	2.91	1.21D+05
GENROSE	1000	5983	424	4346	6825	9.57	162.70	1.00D+00
LIARWHD	1000	18	4	12	6	1.50	0.45	1.14D-13
MANCINO	100	51	12	21	14	1.17	55.04	4.44D-18
MOREBV	1000	4	3	4	519	10.00	2.26	1.37D-09
NCB20B	1000	368	17	214	962	9.65	144.83	1.68D+03
NONDIA	1000	14	4	10	5	1.25	0.45	1.48D-15
NONDQUAR	1000	624	37	542	1843	8.62	11.97	2.77D-06
PENALTY1	1000	738	126	478	239	1.90	11.07	9.69D-03
POWELLSG	1000	42	5	35	17	3.40	0.44	4.38D-11
POWER	1000	95	11	71	150	6.82	1.33	6.72D-11
QUARTC	1000	58	9	37	18	2.00	0.45	1.75D-08
SINQUAD	1000	193	36	123	69	1.92	6.51	1.97D-07
SROSENBR	1000	16	5	11	7	1.40	0.24	7.59D-16
TOINTGSS	1000	10	5	7	7	1.40	0.49	1.00D+01
TQUARTIC	1000	32	6	17	9	1.50	0.49	1.10D-17
TRIDIA	1000	17	10	11	642	9.80	1.79	5.12D-15
VARDIM	1000	64	13	27	13	1.00	0.48	1.33D-21
VAREIGVL	1000	35	7	29	74	6.86	2.02	1.39D-10
WOODS	1000	33	5	25	14	2.80	0.57	5.80D-16

Table A.9: Results for ISM(n,e,f) on large or hard problems, in which no preconditioning is used and the subspace is chosen from 10 extreme CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, *time* = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	21	5	12	6	1.20	0.72	0.00D+00
BDQRTIC	1000	31	7	18	9	1.29	1.21	3.98D+03
BROWNAL	100	92	11	75	134	7.27	3.86	2.40D-13
BRYBND	1000	74	10	34	25	2.50	2.92	9.63D-13
CRAGGLVY	1000	45	12	24	12	1.00	2.12	3.36D+02
DIXMAANA	1500	18	6	13	8	1.33	1.20	1.00D+00
DIXON3DQ	1000	3	1	2	1	1.00	0.07	0.00D+00
DQDRTIC	1000	3	1	2	1	1.00	0.11	1.24D-21
DQRTIC	1000	49	12	25	12	1.00	0.54	2.68D-08
EDENSCH	1000	62	10	24	12	1.20	1.91	6.00D+03
EIGENALS	110	125	11	74	51	4.45	1.81	1.06D-11
ENGVAL1	1000	26	7	14	7	1.00	0.89	1.11D+03
FLETCHCR	1000	6376	1797	3594	1797	1.00	191.23	9.70D-16
FMINSURF	1024	227	11	104	501	10.00	14.20	1.00D+00
FREUROTH	1000	102	8	18	9	1.13	2.31	1.21D+05
GENROSE	1000	3659	656	2021	1011	1.54	93.05	1.00D+00
LIARWHD	1000	25	5	17	12	2.40	0.79	9.91D-16
MANCINO	100	48	14	26	16	1.14	64.15	1.07D-18
MOREBV	1000	2	1	2	1	1.00	0.11	7.33D-13
NCB20B	1000	165	15	73	523	6.67	73.48	1.68D+03
NONDIA	1000	156	25	113	58	2.32	5.24	3.05D-14
NONDQUAR	1000	39	7	29	15	2.14	0.61	2.77D-09
PENALTY1	1000	252	19	130	177	3.63	9.16	9.69D-03
POWELLSG	1000	50	13	30	15	1.15	0.71	5.05D-09
POWER	1000	47	11	23	11	1.00	3.32	4.44D-10
QUARTC	1000	49	12	25	12	1.00	0.54	2.68D-08
SINQUAD	1000	263	38	182	106	2.79	10.78	3.82D-09
SROSENBR	1000	22	8	15	8	1.00	0.48	4.99D-28
TOINTGSS	1000	3	1	2	1	1.00	0.17	1.00D+01
TQUARTIC	1000	21	5	16	8	1.60	0.54	2.45D-14
TRIDIA	1000	3	1	2	1	1.00	0.07	1.01D-25
VARDIM	1000	181	30	136	2999	5.53	34.14	4.48D-13
VAREIGVL	1000	43	7	39	31	4.43	4.15	1.86D-12
WOODS	1000	51	13	28	14	1.08	1.10	1.83D-16

Table A.10: Results for ISM(p,e,f) on large or hard problems, in which preconditioning is used and the subspace is chosen from the 10 extreme CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	11	3	6	4	2.00	0.32	0.00D+00
BDQRTIC	1000	50	11	33	63	2.45	2.05	3.98D+03
BROWNAL	100	5	2	4	3	2.00	0.05	2.45D-08
BRYBND	1000	61	11	40	85	2.64	3.16	1.26D-13
CRAGGLVY	1000	113	12	56	121	3.83	4.27	3.36D+02
DIXMAANA	1500	19	7	14	9	2.14	1.05	1.00D+00
DIXON3DQ	1000	7	4	5	1748	23.00	4.21	1.25D-11
DQDRTIC	1000	10	5	6	12	2.40	0.30	1.91D-14
DQRTIC	1000	58	9	37	18	2.00	0.44	1.75D-08
EDENSCH	1000	54	8	28	26	3.00	1.62	6.00D+03
EIGENALS	110	96	15	71	140	3.93	1.66	1.23D-11
ENGVAL1	1000	30	8	19	22	2.75	0.97	1.11D+03
FLETCHCR	1000	7718	1261	5049	14667	2.03	232.51	2.52D-13
FMINSURF	1024	189	11	101	2301	10.82	24.82	1.00D+00
FREUROTH	1000	59	9	29	25	2.44	1.96	1.21D+05
GENROSE	1000	5233	483	3469	7745	5.28	145.79	1.00D+00
LIARWHD	1000	18	4	12	6	2.00	0.44	1.14D-13
MANCINO	100	30	12	19	14	2.00	43.03	1.47D-18
MOREBV	1000	4	3	4	519	11.00	2.25	1.37D-09
NCB20B	1000	139	20	74	1021	2.60	105.74	1.68D+03
NONDIA	1000	14	4	10	5	2.00	0.45	1.48D-15
NONDQUAR	1000	316	49	218	2483	2.69	9.70	1.55D-06
PENALTY1	1000	738	126	478	239	2.00	11.27	9.69D-03
POWELLSG	1000	58	11	44	32	2.27	0.61	5.24D-11
POWER	1000	83	10	59	131	4.10	1.09	2.79D-09
QUARTC	1000	58	9	37	18	2.00	0.44	1.75D-08
SINQUAD	1000	169	33	105	64	2.18	5.78	4.97D-06
SROSENBR	1000	16	5	11	7	2.00	0.23	7.59D-16
TOINTGSS	1000	10	5	7	7	2.00	0.50	1.00D+01
TQUARTIC	1000	32	6	17	9	2.00	0.49	1.10D-17
TRIDIA	1000	17	10	11	591	9.50	1.66	7.26D-14
VARDIM	1000	64	13	27	13	2.00	0.47	1.33D-21
VAREIGVL	1000	34	7	28	71	3.14	1.98	2.21D-10
WOODS	1000	34	6	26	16	2.50	0.62	1.53D-17

Table A.11: Results for ISM(n,f,a) on large or hard problems, in which no preconditioning is used and the subspace dimension is chosen automatically from the first CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	21	5	12	6	2.00	0.71	0.00D+00
BDQRTIC	1000	31	7	18	9	2.00	1.18	3.98D+03
BROWNAL	100	57	11	43	128	2.82	3.67	7.56D-12
BRYBND	1000	72	10	32	25	2.50	2.90	8.59D-13
CRAGGLVY	1000	45	12	24	12	2.00	2.12	3.36D+02
DIXMAANA	1500	18	6	13	8	2.17	1.23	1.00D+00
DIXON3DQ	1000	3	1	2	1	2.00	0.07	0.00D+00
DQDRTIC	1000	3	1	2	1	2.00	0.11	1.24D-21
DQRTIC	1000	49	12	25	12	2.00	0.54	2.68D-08
EDENSCH	1000	62	10	24	12	2.00	1.92	6.00D+03
EIGENALS	110	107	16	62	58	2.44	1.82	2.51D-12
ENGVAL1	1000	26	7	14	7	2.00	0.90	1.11D+03
FLETCHCR	1000	6376	1797	3594	1797	2.00	189.09	9.70D-16
FMINSURF	1024	199	12	76	540	7.08	14.05	1.00D+00
FREUROTH	1000	102	8	18	9	2.00	2.33	1.21D+05
GENROSE	1000	3659	656	2021	1011	2.00	93.35	1.00D+00
LIARWHD	1000	52	9	28	19	2.11	1.39	3.01D-16
MANCINO	100	48	14	26	16	2.00	64.05	1.07D-18
MOREBV	1000	2	1	2	1	2.00	0.10	7.33D-13
NCB20B	1000	98	15	65	680	5.20	78.37	1.68D+03
NONDIA	1000	156	25	113	58	2.48	5.25	3.05D-14
NONDQUAR	1000	37	8	27	16	2.00	0.62	2.28D-09
PENALTY1	1000	225	38	130	347	2.24	15.17	9.69D-03
POWELLSG	1000	50	13	30	15	2.08	0.72	5.05D-09
POWER	1000	47	11	23	11	2.00	3.32	4.44D-10
QUARTC	1000	49	12	25	12	2.00	0.53	2.68D-08
SINQUAD	1000	325	71	216	225	2.24	16.05	1.20D-08
SROSENBR	1000	22	8	15	8	2.00	0.48	4.99D-28
TOINTGSS	1000	3	1	2	1	2.00	0.18	1.00D+01
TQUARTIC	1000	21	5	16	8	2.00	0.55	2.45D-14
TRIDIA	1000	3	1	2	1	2.00	0.07	1.01D-25
VARDIM	1000	82	25	46	2943	2.32	29.97	2.95D-13
VAREIGVL	1000	58	10	49	62	3.40	5.84	2.43D-11
WOODS	1000	51	13	28	14	2.00	1.10	1.83D-16

Table A.12: Results for ISM(p,f,a) on large or hard problems, in which preconditioning is used and the subspace dimension is chosen automatically from the first CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	11	3	6	4	2.00	0.31	0.00D+00
BDQRTIC	1000	50	11	33	63	2.45	2.02	3.98D+03
BROWNAL	100	5	2	4	3	2.00	0.05	2.45D-08
BRYBND	1000	65	11	42	87	2.73	3.25	3.89D-14
CRAGGLVY	1000	64	12	49	115	3.42	3.39	3.36D+02
DIXMAANA	1500	19	7	14	9	2.14	1.04	1.00D+00
DIXON3DQ	1000	7	4	5	1748	23.00	4.50	1.25D-11
DQDRTIC	1000	10	5	6	12	2.40	0.29	1.91D-14
DQRTIC	1000	58	9	37	18	2.00	0.45	1.75D-08
EDENSCH	1000	54	8	28	26	3.00	1.63	6.00D+03
EIGENALS	110	91	15	68	125	4.67	1.59	1.87D-10
ENGVAL1	1000	30	8	19	22	2.75	0.96	1.11D+03
FLETCHCR	1000	7699	1259	5054	14667	2.04	232.52	4.61D-15
FMINSURF	1024	184	10	91	1953	10.00	21.31	1.00D+00
FREUROTH	1000	90	8	29	25	2.50	2.29	1.21D+05
GENROSE	1000	4975	498	3332	7760	4.83	141.48	1.00D+00
LIARWHD	1000	18	4	12	6	2.00	0.44	1.14D-13
MANCINO	100	30	12	19	14	2.00	43.20	1.47D-18
MOREBV	1000	4	3	4	519	11.00	2.29	1.37D-09
NCB20B	1000	143	20	80	1004	2.50	105.97	1.68D+03
NONDIA	1000	14	4	10	5	2.00	0.46	1.48D-15
NONDQUAR	1000	318	51	219	2869	2.33	10.79	1.40D-06
PENALTY1	1000	738	126	478	239	2.00	11.15	9.69D-03
POWELLSG	1000	58	11	44	32	2.27	0.62	5.24D-11
POWER	1000	83	10	59	131	4.10	1.08	2.79D-09
QUARTC	1000	58	9	37	18	2.00	0.45	1.75D-08
SINQUAD	1000	180	36	112	72	2.17	6.19	1.98D-06
SROSENBR	1000	16	5	11	7	2.00	0.23	7.59D-16
TOINTGSS	1000	10	5	7	7	2.00	0.49	1.00D+01
TQUARTIC	1000	32	6	17	9	2.00	0.50	1.10D-17
TRIDIA	1000	17	10	11	631	8.90	1.76	1.53D-14
VARDIM	1000	64	13	27	13	2.00	0.47	1.33D-21
VAREIGVL	1000	34	7	28	73	3.00	1.99	2.12D-10
WOODS	1000	34	6	26	16	2.50	0.63	1.53D-17

Table A.13: Results for ISM(n,e,a) on large or hard problems, in which no preconditioning is used and the subspace dimension is chosen automatically from extreme CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.

problem	n	#f	#min	#its	#cg	$s^{(k)}$	time	f
ARWHEAD	1000	21	5	12	6	2.00	0.72	0.00D+00
BDQRTIC	1000	31	7	18	9	2.00	1.18	3.98D+03
BROWNAL	100	68	12	55	141	2.92	4.05	1.98D-13
BRYBND	1000	72	10	32	25	2.50	2.88	8.59D-13
CRAGGLVY	1000	45	12	24	12	2.00	2.14	3.36D+02
DIXMAANA	1500	18	6	13	8	2.17	1.21	1.00D+00
DIXON3DQ	1000	3	1	2	1	2.00	0.07	0.00D+00
DQDRTIC	1000	3	1	2	1	2.00	0.11	1.24D-21
DQRTIC	1000	49	12	25	12	2.00	0.53	2.68D-08
EDENSCH	1000	62	10	24	12	2.00	1.90	6.00D+03
EIGENALS	110	112	18	65	65	2.28	1.98	1.77D-14
ENGVAL1	1000	26	7	14	7	2.00	0.90	1.11D+03
FLETCHCR	1000	6376	1797	3594	1797	2.00	188.31	9.70D-16
FMINSURF	1024	196	14	80	690	6.36	16.49	1.00D+00
FREUROTH	1000	102	8	18	9	2.00	2.30	1.21D+05
GENROSE	1000	3659	656	2021	1011	2.00	92.87	1.00D+00
LIARWHD	1000	52	9	28	19	2.11	1.39	3.01D-16
MANCINO	100	48	14	26	16	2.00	64.08	1.07D-18
MOREBV	1000	2	1	2	1	2.00	0.10	7.33D-13
NCB20B	1000	108	15	67	565	4.73	70.89	1.68D+03
NONDIA	1000	156	25	113	58	2.48	5.23	3.05D-14
NONDQUAR	1000	37	8	27	16	2.00	0.62	2.28D-09
PENALTY1	1000	197	25	127	175	3.04	10.40	9.69D-03
POWELLSG	1000	50	13	30	15	2.08	0.71	5.05D-09
POWER	1000	47	11	23	11	2.00	3.30	4.44D-10
QUARTC	1000	49	12	25	12	2.00	0.53	2.68D-08
SINQUAD	1000	210	37	146	123	2.68	9.66	6.03D-08
SROSENBR	1000	22	8	15	8	2.00	0.47	4.99D-28
TOINTGSS	1000	3	1	2	1	2.00	0.17	1.00D+01
TQUARTIC	1000	21	5	16	8	2.00	0.54	2.45D-14
TRIDIA	1000	3	1	2	1	2.00	0.07	1.01D-25
VARDIM	1000	108	27	69	2801	2.30	30.63	2.63D-13
VAREIGVL	1000	68	11	58	91	3.27	6.80	1.19D-11
WOODS	1000	51	13	28	14	2.00	1.10	1.83D-16

Table A.14: Results for ISM(p,e,a) on large or hard problems, in which preconditioning is used and the subspace dimension is chosen automatically from extreme CG directions.
Key: n = number of variables, $\#f$ = number of function evaluations, $\#min$ = number of minimizations, $\#its$ = total number of iterations, $\#cg$ = total number of CG iterations, $s^{(k)}$ = average subspace dimension, $time$ = total CPU time in seconds, f = smallest function value obtained.