Preconditioning Newton–Krylov methods in nonconvex large scale optimization

GIOVANNI FASANO[1,2] and MASSIMO ROMA[2]

[1] Università Ca' Foscari di Venezia
Ca' Dolfin Dorsoduro, 3825/E
30123 Venezia, ITALY
E-mail: fasano@unive.it

[2] Dipartimento di Informatica e Sistemistica "A. Ruberti"
SAPIENZA, Università di Roma
via Ariosto, 25
00185 Roma, ITALY
E-mail: (fasano,roma)@dis.uniroma1.it

Abstract

We consider an iterative preconditioning technique for large scale optimization, where the objective function is possibly non-convex. First, we refer to the solution of a generic indefinite linear system by means of a Krylov subspace method, and describe the iterative construction of the preconditioner which does not involve matrices products or matrix storage. The set of directions generated by the Krylov subspace method is also used, as by product, to provide an approximate inverse of the system matrix. Then, we experience our method within Truncated Newton schemes for large scale unconstrained optimization, in order to speed up the solution of the Newton equation. Actually, we use a Krylov subspace method to approximately solve the Newton equation at current iterate (where the Hessian matrix is possibly indefinite) and to construct the preconditioner to be used at the current outer iteration. An extensive numerical experience show that the preconditioning strategy proposed leads to a significant reduction of the overall inner iterations on most of the test problems considered.

Keywords : Large Scale Optimization, Non–convex problems, Krylov subspace methods, Newton–Krylov methods, Preconditioning.

1 Introduction

In this paper we deal with a preconditioning strategy to be used for the efficient solution of the large scale linear systems, which arise very frequently in large scale nonlinear optimization. As well known, there are many and different contexts of nonlinear optimization in which the iterative solution of sequences of linear systems is required. Some examples are Truncated Newton methods in unconstrained optimization, equality and/or inequality constrained problems, KKT systems, interior point methods, PDE constrained optimization and many others.

Krylov subspace methods are usually used for iteratively solving such linear systems by means of "matrix–free" implementations. Some of the most commonly used are the Conjugate Gradient methods for symmetric positive definite systems, the Lanczos algorithm for symmetric systems, GMRES, Bi-CGSTAB, QMR methods for unsymmetric systems.

In this work we consider symmetric indefinite linear systems and focus on the possibility of constructing good preconditioners for Krylov subspace methods. In particular, our aim is to construct a preconditioner by using an iterative decomposition of the system matrix, commonly obtained as by product of the Krylov subspace methods (see, e.g. [22, 10]) and without requiring a significant additional computational effort. Firstly, we state the least requirements that a Krylov subspace method must satisfy to be suited to this aim. Then we define the preconditioner with reference to a generic Krylov subspace method and prove interesting properties of such a preconditioner. Finally, we focus on a particular Krylov subspace method, a version of the planar Conjugate Gradient (CG) method, which is an extension of the well known standard CG method. In particular, we refer to the planar CG algorithm FLR [6, 9]. We consider this algorithm since it prevents from pivot breakdowns of the standard CG algorithm when the system matrix is not positive definite. As a consequence we perform a number of iterations enough to generate a good approximation of the inverse of the system matrix, to be used for building the preconditioner. We show how the construction of such a preconditioner is performed without storing any $n \times n$ matrix (where n is the dimension of the vector of the variables) or computing matrices products or explicit matrices inverse. Actually, we assume that the entries of the system matrix are not known and the only information of the system matrix is gained by means of a routine, which provides the product of the matrix times a vector. This routine is available for free, since already required by any implementation of a Krylov subspace method.

We use our preconditioner within a nonlinear optimization framework, namely in solving nonconvex large scale unconstrained problems. In particular, we focus on the so called *Newton–Krylov methods*, also called Truncated Newton (TN) methods (see [18] for a survey). TN methods are implementations of the Newton methods, in which a Krylov subspace method is used for approximately solving the linear system arising in computing the search direction. In this context, all we need to compute our preconditioner is already available, as by product of the inner iterations performed for computing the search direction.

Notwithstanding Truncated Newton methods have been widely studied and extensively tested, it is largely recognized (see, e.g. [20]) that, especially in dealing with large scale problems, two key aspects for the overall efficiency of the method can be still considered worthwhile to be investigated: the first one regards the formulation and handling of a preconditioner which enables to accelerate the convergence of the iterates. In fact, for Truncated Newton methods, preconditioning is today considered an essential tool for achieving a good efficiency and robustness, especially in the large scale setting. The second one concerns the possibility to tackle nonconvex problems and hence to handle the indefinite case for Newton's equation. In these paper we try to tackle both these aspects in the large scale setting. As regards the first issue, we embed our preconditioner within the inner iterations of the TN methods, according to a suited strategy. Numerical results show that this leads to an overall reduction of the number of CG inner iterations, needed in most of the test problems considered. As regards the possibility to handle nonconvex problems, we adopt the planar CG method FRL for constructing the preconditioner, instead of the standard CG methods, in order to avoid pivot breakdown. Moreover, drawing our inspirations from [12], in computing the search direction by the preconditioned CG, we adopt a strategy which allows us to overcome the usual drawbacks in dealing with indefinite problems, namely the untimely termination of the inner iterations.

The paper is organized as follows: in Section 2, we consider the problem of constructing a preconditioner for an indefinite linear system, by using a generic Krylov subspace method. In Section 3, we focus on a particular Krylov subspace method which allow us to tackle indefinite systems, the planar CG algorithm FLR, which is first recalled for sake of completeness. In Section 4 an iterative matrix decomposition is obtained for the preconditioned CG. In Section 5, we focus on Truncated Newton methods for unconstrained optimization and define a new preconditioning strategy for the class of preconditioned Truncated Newton methods. Moreover, we describe the strategy we adopt in order to efficiently tackling large scale indefinite problems. Finally, in Section 6 we report the results of an extensive numerical testing for the resulting Truncated Newton method, on a selection of large scale unconstrained test problems. We show that the approach we propose in this paper is reliable and enables a significant reduction of the number of the inner iterations, in solving both convex and nonconvex problems.

2 Iterative matrix decomposition and the new preconditioner

In this section we consider a general symmetric linear system, which is solved by using a Krylov subspace method. We describe the conditions which should be satisfied by the Krylov subspace method, in order to iteratively obtaining a decomposition of the system matrix, for constructing the preconditioner we propose. To this aim, consider the *indefinite* linear system

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and *nonsingular*, n is *large* and $b \in \mathbb{R}^n$, and consider any Krylov subspace methods for the solution of such a system (see, e.g. [10]). Krylov subspace methods are extensively used for the iterative solution of linear systems. The Lanczos method and the Conjugate Gradient method are among the most popular. They are equivalent as long as the matrix A is positive definite, whereas the Conjugate Gradient, though cheaper, does not cope with the indefinite case.

Our aim is to focus on those Krylov subspace methods which enable to iteratively construct a preconditioner to be used in the solution of the system (2.1). The next Assumption 2.1 summarizes the least requirements for the Krylov subspace method we adopt. Suppose to perform a finite number of iterations, say $h \ll n$, of an iterative Krylov subspace method.

Assumption 2.1 *At the step $h \geq 1$ of the Krylov method the matrices R_h, T_h, L_h and B_h are generated, such that*

$$\begin{aligned} R_h^T A R_h &= T_h, & (2.2)\\ T_h &= L_h B_h L_h^T, & (2.3) \end{aligned}$$

where

$R_h = (u_1 \cdots u_h), \;\; u_i^T u_j = 0, \;\; \|u_i\| = 1, \;\; i = 1, \dots, h,$

T_h is symmetric tridiagonal and irreducible,

L_h is lower triangular with a "simple pattern" for its entries,

B_h is a non-singular block diagonal matrix whose inverse is "easy-to-compute".

On the basis of the latter assumption, we can now define the preconditioning matrix and show its properties. First we observe that, since T_h is symmetric and irreducible, the orthogonal matrix $W_h \in \mathbb{R}^{h\times h}$ exists such that $T_h = W_h D_h W_h^T$, with

$D_h = diag_{1 \le i \le h}\{d_i\}$ non-singular. Then, we can define the matrix $|D_h| = diag_{1 \le i \le h}\{|d_i|\}$ and, accordingly, the matrix $|T_h|$

$$|T_h| = W_h |D_h| W_h^T.$$

Let us now introduce the following matrix

$$M_h \; = \; (I - R_h R_h^T) \; + \; R_h |T_h| R_h^T, \tag{2.4}$$

where R_h and T_h satisfy relation (2.2).

Theorem 2.1 *Consider any Krylov subspace method to solve (2.1) and suppose it performs a number $h \ll n$ of iterations, and that Assumption 2.1 holds. Then we have:*

a) M_h is symmetric and nonsingular;

b) $M_h^{-1} = (I - R_h R_h^T) \; + \; R_h |T_h|^{-1} R_h^T$;

c) M_h is positive definite and its spectrum $\Lambda(M_h)$ is given by

$$\Lambda(M_h) = \Lambda(|T_h|) \cup \Lambda(I_{n-h}),$$

where $\Lambda(I_{n-h})$ is the set of the $n-h$ unit eigenvalues of the identity matrix I_{n-h};

d) $\Lambda(M_h^{-1}A) = \Lambda(AM_h^{-1})$ and contains at least h eigenvalues in the set $\{-1, +1\}$.

Proof: As regards a), the symmetry trivially follows from the symmetry of T_h. Moreover, since A in (2.1) is symmetric, the matrix $R_{n,h}$ exists such that $R_{n,h}^T R_{n,h} = I_{n-h}$ and the columns of the matrix $[R_h \vdots R_{n,h}]$ are an orthogonal basis of $\mathbb{R}^n$. Thus, for any $v \in \mathbb{R}^n$ we can write $v = R_h v_1 + R_{n,h} v_2$, with $v_1 \in \mathbb{R}^h$ and $v_2 \in \mathbb{R}^{n-h}$. Now, to prove M_h is invertible, we show that $M_h v = 0$ implies $v = 0$. In fact,

$$M_h v = R_{n,h} v_2 + R_h |T_h| v_1 = \left[R_h |T_h| \vdots R_{n,h} \right] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0,$$

if and only of $(v_1^T \ v_2^T)^T = 0$, since from Assumption 2.1 the matrix $|T_h|$ is nonsingular.

As regards b), recalling that $R_h^T R_h = I$ and $|T_h|$ is nonsingular from Assumption 2.1, a direct computation yields the result.

As concerns item c), since $I - R_h R_h^T = R_{n,h} R_{n,h}^T$, we can write

$$M_k = [R_h \quad R_{n,h}] \begin{bmatrix} |T_h| & 0 \\ 0 & I_{n-h} \end{bmatrix} \begin{bmatrix} R_h^T \\ R_{n,h}^T \end{bmatrix}$$

which gives the results, since T_h is irreducible and thus $|T_h|$ is positive definite.

Item d) may be proved by considering that the matrices $M_h^{-1} A$ and $A M_h^{-1}$ are similar to the matrix $M_h^{-1/2} A M_h^{-1/2}$ and hence $\Lambda(M_h^{-1} A) = \Lambda(M_h^{-1/2} A M_h^{-1/2}) = \Lambda(A M_h^{-1})$. Moreover, we have

$$M_h^{-1/2} = \left[R_h \ \vdots \ R_{n,h} \right] \left[\begin{array}{c|c} |T_h|^{-1/2} & 0 \\ \hline 0 & I_{n-h} \end{array} \right] \begin{bmatrix} R_h^T \\ R_{n,h}^T \end{bmatrix},$$

and hence

$$M_h^{-1/2} A M_h^{-1/2} =$$

$$= \left[R_h \ \vdots \ R_{n,h} \right] \left[\begin{array}{c|c} |T_h|^{-1/2} & 0 \\ \hline 0 & I_{n-h} \end{array} \right] \left[\begin{array}{c|c} R_h^T A R_h & R_h^T A R_{n,h} \\ \hline R_{n,h}^T A R_h & R_{n,h}^T A R_{n,h} \end{array} \right]$$
$$\left[\begin{array}{c|c} |T_h|^{-1/2} & 0 \\ \hline 0 & I_{n-h} \end{array} \right] \begin{bmatrix} R_h^T \\ R_{n,h}^T \end{bmatrix}.$$

Recalling that $A R_h = R_h T_h$, we have

$$R_h^T A R_{n,h} = (R_{n,h}^T A R_h)^T = (R_{n,h}^T R_h T_h)^T = 0,$$

and from (2.2) we obtain

$$M_h^{-1/2} A M_h^{-1/2} = \left[R_h \ \vdots \ R_{n,h} \right] \left[\begin{array}{c|c} I_h & 0 \\ \hline 0 & R_{n,h}^T A R_{n,h} \end{array} \right] \begin{bmatrix} R_h^T \\ R_{n,h}^T \end{bmatrix}.$$

which proves that at least h eigenvalues of $M_h^{-1/2} A M_h^{-1/2}$ are in the set $\{-1, +1\}$ and this completes the proof. $\square$

Our aim is to use the matrix M_h, with $h \ll n$, as preconditioner, as we will describe in the sequel, in order to use information gathered during the iteration of the Krylov subspace method. Note that this is not the first attempt in literature to use a preconditioner of this form. In fact, a similar approach has been considered in the context of GMRES methods (see [4, 1, 14]). However, it is very important to note that GMRES information is given in the form of Hessenberg decomposition of the matrix A, and not as a tridiagonal one, as in our approach. However, the main distinguishing feature of the approach we propose consists of the choice of the particular Krylov subspace method we adopt. Indeed our choice allows us to construct and use the preconditioner without storing any $n \times n$ matrix and without any explicit matrix inversion. In particular, as described in Section 3, to perform the preconditioned method it will suffice to store $h \ll n$ n-dimensional vectors, the diagonal elements of a $h \times h$ diagonal matrix and the subdiagonal elements of a lower bidiagonal $h \times h$ matrix. All the entries are available as by product from the iterates of the Krylov method. In the next section we will describe the method we chose to adopt.

3 Computing the new preconditioner via the Planar-CG algorithm FLR

The Krylov subspace method we propose to use in this paper is the Conjugate Gradient-type method FLR introduced in [7]. Unlike the Conjugate Gradient, it copes with the indefinite case too. It is an iterative method for solving indefinite linear systems, and is a modification of the standard CG algorithm [13]. Now we report a scheme of this method.

Algorithm FLR

```
Step 1: $k = 1$, $s_1 = 0$, $r_1 = b$. If $r_1 = 0$ then STOP,
        else compute $p_1 = r_1$.

Step k: Compute $\sigma_k = p_k^T A p_k$.

    If $| \sigma_k | \geq \epsilon_k \|p_k\|^2$ then go to Step $k_A$
    else go to Step $k_B$

    – Step $k_A$ (standard CG step) :
      Set $s_{k+1} = s_k + a_k p_k$,  $r_{k+1} = r_k - a_k A p_k$ ,
      where $a_k = \frac{r_k^T p_k}{\sigma_k} = \frac{\|r_k\|^2}{\sigma_k}$.

      If $r_{k+1} = 0$ then STOP
      else compute $p_{k+1} = r_{k+1} + \beta_k p_k$
      with $\beta_k = \frac{-p_k^T A r_{k+1}}{\sigma_k} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$.

      Set $k = k + 1$ and go to Step $k$.

    – Step $k_B$ (planar CG step) :
      If $k = 1$ then compute the vector $q_k = A p_k$,
      else compute the vector

      $q_k = \begin{cases} A p_k + b_{k-1} p_{k-1}, & \text{if the previous step is } \textit{Step } (k-1)_A \\ A p_k + \frac{\hat{b}_{k-2}}{\Delta_{k-2}} \left( \sigma_{k-2} q_{k-2} - \delta_{k-2} p_{k-2} \right), & \text{if the previous step is } \textit{Step } (k-2)_B \end{cases}$

      where $b_{k-1} = -(A p_{k-1})^T A p_k / \sigma_{k-1}$ and
      $\hat{b}_{k-2} = -(A q_{k-2})^T A p_k$.
```

```
Compute $c_k = r_k^T p_k$, $\delta_k = p_k^T A q_k$, $e_k = q_k^T A q_k$,
$\Delta_k = \sigma_k e_k - \delta_k^2$ and $\hat{c}_k = (c_k e_k - \delta_k q_k^T r_k)/\Delta_k$,
$\hat{\sigma}_k = (\sigma_k q_k^T r_k - \delta_k c_k)/\Delta_k$.

Set $s_{k+2} = s_k + \hat{c}_k p_k + \hat{\sigma}_k q_k$ and
$r_{k+2} = r_k - \hat{c}_k A p_k - \hat{\sigma}_k A q_k$.

If $r_{k+2} = 0$ then STOP
else compute $p_{k+2} = r_{k+2} + \frac{\hat{\beta}_k}{\Delta_k}(\sigma_k q_k - \delta_k p_k)$,
with $\hat{\beta}_k = -q_k^T A r_{k+2}$.

Set $k = k + 2$ and go to *Step k.*
```

Further details on the FLR algorithm can be found in [7, 6]; here we simply consider some relevant results, which are used in order to obtain relations (2.2)-(2.3).

First observe that as long as at Step k the planar CG Step k_B is not performed, the FLR algorithm reduces to the standard CG and hence, at Step k_A the algorithm detects the solution of (2.1) along the conjugate direction p_k. On the contrary, if a pivot breakdown occurs at Step k (i.e. $p_k^T A p_k \approx 0$), the FLR algorithm generates another direction at Step k_B (namely q_k). Then, it performs a search for the system solution on the 2-dimensional linear manifold $s_k + \text{span}\{p_k, q_k\}$, and generates the new point s_{k+2}. In addition it can be easily proved (see [7]) that, if the indefinite matrix A is nonsingular and at Step k we have $r_k \neq 0$, then the FLR algorithm can always perform either Step k_A or Step k_B. As concerns the assessment of the parameter ϵ_k at the Step k, some proposals where considered in [8, 9], in order to avoid possible instabilities.

For sake of completeness, now we report the tridiagonalization procedure proposed in the paper [9], which will be at the basis of the construction of the matrices R_h, T_h and B_h mentioned in the Assumption 2.1.

Let us consider the Algorithm FLR at a generic Step k and let us introduce the following notation: if at Step k of the FLR algorithm the condition $|p_k^T A p_k| \geq \epsilon_k \|p_k\|^2$ is satisfied, then we set $w_k = p_k$ (standard CG step); otherwise we set $w_k = p_k$ and $w_{k+1} = q_k$ (planar CG step). According with the latter positions, the sequence $\{w_i\}$ represents the sequence of directions generated by the Algorithms FLR which contains, at most, pairs of consecutive non–conjugate directions.

As regards the sequence $\{r_i\}$ of the residuals generated by the algorithm FLR up to Step k, if a planar CG Step k_B occurs, we have two directions (w_k and w_{k+1}) and only one residual r_k. To overcome this drawback, if the Step k is the planar Step k_B, we introduce a "dummy" residual r_{k+1}, which completes the sequence of orthogonal vectors $\{r_1, \ldots, r_k, r_{k+1}\}$ [2, 5]. The possible choices for r_{k+1} in order to satisfy the conditions $r_{k+1} \in \mathcal{K}_k(A, r_1)$ and $r_{k+1} \notin \mathcal{K}_{k-1}(A, r_1)$ and to preserve the orthogonality conditions $r_{k+1}^T p_k = r_{k+1}^T r_k$ are:

$$\begin{aligned} r_{k+1} &= \pm \left[\hat{\alpha}_k r_k + (1 + \hat{\alpha}_k)\ sgn(\sigma_k) A p_k\right], \qquad (3.1) \\ \text{with} \quad \hat{\alpha}_k &= -\frac{|\sigma_k|}{\|r_k\|^2 + |\sigma_k|}. \end{aligned}$$

Therefore, after $h \leq n$ steps of the FLR algorithm the following matrices can be defined:

$$R_h = \left(\frac{r_1}{\|r_1\|} \cdots \frac{r_h}{\|r_h\|} \right) \in \mathbb{R}^{n \times h}, \quad P_h = \left(\frac{w_1}{\|r_1\|} \cdots \frac{w_h}{\|r_h\|} \right) \in \mathbb{R}^{n \times h}.$$

In the remainder of this section we give evidence that the Algorithm FLR can also provide the matrices L_h and B_h mentioned in the Assumption 2.1, such that (2.2) and (2.3) hold. To this aim we report the statement of Theorem 2.1 in [9].

Theorem 3.1 *Consider the* FLR *algorithm where A is symmetric, indefinite and nonsingular. Suppose $\epsilon_k > 0$ and let $\|r_i\| \neq 0$, $i \leq h$. Assume the only one planar CG step performed by the* FLR *algorithm is Step* $k_B < h$. *Then the following relations hold:*

$$
\begin{aligned}
P_h \tilde{L}_h^T &= R_h, && (3.2)\\
AP_h &= \left(R_h \;\vdots\; \frac{r_{h+1}}{\|r_{h+1}\|} \right) \begin{pmatrix} \bar{L}_h \\ \bar{l}_{h+1,h} e_h^T \end{pmatrix} D_h, \quad h < n, && (3.3)\\
AR_h &= \left(R_h \;\vdots\; \frac{r_{h+1}}{\|r_{h+1}\|} \right) \begin{pmatrix} T_h \\ t_{h+1,h} e_h^T \end{pmatrix}, \quad h < n, && (3.4)
\end{aligned}
$$

where

$$
\tilde{L}_h = \begin{pmatrix}
1 & & & & & & & & & \\
-\sqrt{\beta_1} & \cdot & & & & & & & & \\
 & \cdot & 1 & & & & & & & \\
 & & -\sqrt{\beta_{k-1}} & 1 & & & & 0 & & \\
 & & & \tilde{\alpha}_1 & \tilde{\alpha}_2 & & & & & \\
 & & & \tilde{\alpha}_3 & \tilde{\alpha}_4 & 1 & & & & \\
 & 0 & & & & -\sqrt{\beta_{k+2}} & \cdot & & & \\
 & & & & & & \cdot & 1 & & \\
 & & & & & & & -\sqrt{\beta_{h-1}} & 1 &
\end{pmatrix} \qquad (3.5)
$$

with ($\beta_k = \|r_{k+1}\|^2/\|r_k\|^2$, $\beta_{k+1} = \|r_{k+2}\|^2/\|r_{k+1}\|^2$)

$$
\begin{aligned}
\tilde{\alpha}_1 &= \frac{\hat{\alpha}_k}{\sqrt{\beta_k}}, & \tilde{\alpha}_2 &= (1 + \hat{\alpha}_k)\ sgn(\sigma_k), \\
\tilde{\alpha}_3 &= \frac{\hat{\beta}_k \delta_k}{\Delta_k \sqrt{\beta_{k+1}\beta_k}}, & \tilde{\alpha}_4 &= -\frac{\hat{\beta}_k\ \sigma_k}{\Delta_k \sqrt{\beta_{k+1}}},
\end{aligned} \qquad (3.6)
$$

$$D_h \;=\; \begin{pmatrix} \frac{1}{a_1} & & & & & & & \\ & \cdot & & & & & 0 & \\ & & \frac{1}{a_{k-1}} & & & & & \\ & & & \frac{1}{\xi_k} & & & & \\ & & & & \frac{1}{\xi_{k+1}} & & & \\ & 0 & & & & \frac{1}{a_{k+2}} & & \\ & & & & & & \cdot & \\ & & & & & & & \frac{1}{a_h} \end{pmatrix}, \quad (3.7)$$

$$\bar{L}_h \;=\; \begin{pmatrix} 1 & & & & & & & & \\ -\sqrt{\beta_1} & \cdot & & & & & & & \\ & \cdot & 1 & & & 0 & & & \\ & & -\sqrt{\beta_{k-1}} & \bar{\alpha}_1 & \bar{\alpha}_3 & & & & \\ & & & \bar{\alpha}_2 & \bar{\alpha}_4 & & & & \\ & & & & \bar{\alpha}_5 & 1 & & & \\ & & 0 & & & -\sqrt{\beta_{k+2}} & \cdot & & \\ & & & & & & \cdot & 1 & \\ & & & & & & & -\sqrt{\beta_{h-1}} & 1 \end{pmatrix}. \quad (3.8)$$

The coefficients ξ_k and ξ_{k+1} are independent arbitrary non-zero *parameters, and $\bar{\alpha}_i$, $i = 1, \dots, 5$, have the following values:*

$$\bar{\alpha}_1 = \frac{\sigma_k}{\|r_k\|^2}\xi_k, \qquad \bar{\alpha}_2 = \sqrt{\beta_k}\left[sgn(\sigma_k) + \frac{\sigma_k}{\|r_k\|^2}\right]\xi_k,$$

$$\bar{\alpha}_3 = \frac{\xi_{k+1}}{\sqrt{\beta_k}\hat{\sigma}_k}\left[1 - \frac{\sigma_k}{\|r_k\|^2}\hat{c}_k\right], \qquad \bar{\alpha}_4 = -\frac{\hat{c}_k\xi_{k+1}}{\hat{\sigma}_k(1+\hat{\alpha}_k)\ sgn(\sigma_k)},$$

$$\bar{\alpha}_5 = -\frac{\xi_{k+1}}{\hat{\sigma}_k}\sqrt{\beta_{k+1}}.$$

Finally, T_h is an irreducible symmetric tridiagonal matrix defined by

$$\begin{pmatrix} T_h \\ 0 \cdots 0 \ t_{h+1,h} \end{pmatrix} = \begin{pmatrix} \bar{L}_h \\ 0 \cdots 0 \ \bar{l}_{h+1,h} \end{pmatrix} D_h \tilde{L}_h^T, \tag{3.9}$$

where $\bar{l}_{h+1,h}$ and $t_{h+1,h}$ are the element $(h+1,h)$ of the matrix $\bar{L}_{h+1}$ and T_{h+1}, respectively.

Moreover, since both $\bar{L}_h$ and $\tilde{L}_h$ are nonsingular from Theorem 3.1 the following result can be obtained (see [9]).

Proposition 3.2 *Consider the* FLR *algorithm where $\epsilon_k > 0$ and let $\|r_i\| \neq 0$, $i \leq h$. Suppose the only one planar CG step performed by the* FLR *Algorithm is Step $k_B < h$. Then, the nonsingular matrix $Q_h \in \mathbb{R}^{h\times h}$ exists such that*

$$\bar{L}_h = \tilde{L}_h Q_h, \tag{3.10}$$

where

$$Q_h = \left(\begin{array}{ccc|cc|ccc} 1 & & & & & & & \\ & \ddots & & & & & 0 & \\ & & 1 & & & & & \\ \hline & & & \pi_{k,k} & \pi_{k,k+1} & & & \\ & & & \pi_{k+1,k} & \pi_{k+1,k+1} & & & \\ \hline & & & & & 1 & & \\ & 0 & & & & & \ddots & \\ & & & & & & & 1 \end{array}\right), \tag{3.11}$$

with

$$\begin{array}{llll} \pi_{k,k} = \bar{\alpha}_1, & & \pi_{k,k+1} = \bar{\alpha}_3, & \\ \pi_{k+1,k} = \dfrac{\bar{\alpha}_2 - \bar{\alpha}_1\tilde{\alpha}_1}{\tilde{\alpha}_2}, & & \pi_{k+1,k+1} = \dfrac{\bar{\alpha}_4 - \bar{\alpha}_3\tilde{\alpha}_1}{\tilde{\alpha}_2}. \end{array} \tag{3.12}$$

Hence, by (3.9) and (3.10) we obtain

$$T_h = \tilde{L}_h(Q_hD_h)\tilde{L}_h^T = \tilde{L}_hB_h\tilde{L}_h^T, \tag{3.13}$$

and iterating (3.10) in case many planar CG steps are performed, it can be proved that the matrix B_h is nonsingular, indefinite and 2×2 block diagonal. Relation (3.13) proves that the FLR Algorithm allows to construct matrices satisfying relation (2.3) of Assumption 2.1. On the other hand, considering the orthogonality conditions of the residuals, relation (3.4) immediately yields (2.2).

Therefore the application of Theorem 3.1 and Proposition 3.2 shows that by means of the Algorithm FLR the matrices R_h, T_h, L_h satisfying Assumption 2.1 can be iteratively constructed. Therefore we are now able to build the preconditioner defined in (2.4). In fact, the orthogonal matrix W exists such that $B_h = W\mathcal{D}W^T$, where $\mathcal{D} = diag_{1\le i\le h}\{\mu_i\}$, and μ_i is the i-th *nonzero* eigenvalue[1] of matrix B_h, and from the latter consideration and relation (3.13) we can build the matrices $|\mathcal{D}|$, $|B_h|$ and $|T_h|$ such that

$$\begin{array}{ll} |\mathcal{D}| = diag_{1\le i\le h}\{|\mu_i|\}, & |B_h| = W|\mathcal{D}|W^T, \\ |T_h| = \tilde{L}_h|B_h|\tilde{L}_h^T, & \end{array} \tag{3.14}$$

which can be used to construct the preconditioner M_h in (2.4). Moreover, from relations (2.4), (3.2) and (3.13), we obtain

$$M_h^{-1} = (I - P_h\tilde{L}_h^T\tilde{L}_hP_h^T) + P_h\tilde{L}_h^T|T_h|^{-1}\tilde{L}_hP_h^T \tag{3.15}$$

$$= (I - P_h\tilde{L}_h^T\tilde{L}_hP_h^T) + P_h|B_h|^{-1}P_h^T. \tag{3.16}$$

[1]Observe that since B_h is 2×2 block diagonal, the computation of the matrix W and the eigenvalues $\mu_1, \ldots, \mu_h$ is straightforward.

It is fundamental to notice that the construction of the preconditioner by the Algorithm FLR *does not require* any matrix inversion (apart from $|B_h|^{-1}$ which is a 2×2 block diagonal matrix). In particular, relation (3.16) reveals that the storage of the matrices P_h (in place of the matrix R_h) and L_h suffices to compute the preconditioned residual $M_h^{-1} r_k$. Finally, observe that P_h should be fully stored, however $\tilde{L}_h$ is sparse and has the very simple structure in (3.5).

An alternative way to construct the preconditioner could be based on the use of the Lanczos algorithm. It is well known that, in the positive definite case, the standard CG and the Lanczos methods are equivalent (see, e.g., [22]). Moreover, as regards the Assumption 2.1, at step h, the Lanczos method provides the matrices R_h and T_h such that the equality (2.2) holds. Indeed it suffices to take as matrix R_h the matrix whose columns are the Lanczos vectors u_i, $i = 1, \dots, h$, i.e. the matrix $U_h = (u_1 \quad \cdots \quad u_h)$. Moreover if we denote by $T_h^{(L)}$ the tridiagonal matrix generated by the Lanczos algorithm, if A is positive definite, the matrix $T_h^{(L)}$ is positive definite too and can be stably decomposed in the form (2.3), where L_h is a unit lower bidiagonal matrix and B_h is simply diagonal. Moreover the Lanczos algorithm can be used also in dealing with the indefinite case, thus overcoming the drawback of the standard CG due to possible pivot breakdown. Moreover, the relations between standard CG and Lanczos algorithm in the definite case are well known (see. e.g. [22]). The correspondence between the planar-CG FLR method and the Lanczos method in the indefinite case has been studied in [8] (see, in particular Theorem 4.3 in [8]).

Now we show how the Lanczos algorithm can be used to determine the matrix M_h in (2.4). However, at the same time here we show that the Algorithm FLR should be preferred not only for its competitive computational burden, but also because it yields formulae (3.15)–(3.16) which avoid the computation of the inverse of a tridiagonal matrix. In fact, the construction of the matrix (2.4) via the Lanczos algorithm, can be immediately derived similarly to the description in [1, 14], as a particular case of the Arnoldi decomposition used therein, i.e. it results

$$M_h = (I - U_h U_h^T) + U_h \left| T_h^{(L)} \right| U_h^T.$$

Here, the tridiagonal matrix $\left| T_h^{(L)} \right|$ is defined analogously to $|T_h|$ in (2.4). Moreover,

$$M_h^{-1} = (I - U_h U_h^T) + U_h \left| T_h^{(L)} \right|^{-1} U_h^T. \tag{3.17}$$

Thus, M_h may be obtained either by the Algorithm FLR or the Lanczos algorithm, without formally modifying its expression in (2.4). However, the implementation of the preconditioned iterative method in hand, for solving system (2.1), claims for an efficient computation, at each iteration i, of the preconditioned residual, i.e. of the product $M_h^{-1} r_i$. Thus, the computational cost of the preconditioner cannot include the full inversion of the tridiagonal matrix $|T_h|^{-1}$. The latter drawback may be overcome by using the Algorithm FLR by means of (3.15)-(3.16). The latter result is not immediately available for the Lanczos algorithm, i.e. the computation of the preconditioned residual by means of the Lanczos algorithm requires the calculation of the dense matrix $|T_h^{(L)}|^{-1}$.

4 Iterative matrix decomposition for preconditioned CG

In this section we analyze some properties of the preconditioned CG algorithm. We aim at carrying out formulae similar to (3.2)-(3.4) of Section 3, which were obtained with the unpreconditioned CG algorithm FLR. For sake of completeness, we now report the scheme of the algorithm Prec-CG which is a standard preconditioned CG scheme. Our aim is to obtain suitable formulae in order to exploit a decomposition of the matrix A.

Algorithm Prec-CG

```
Step 1: Set $k = 1$; $s_1 = 0$; $r_1 = b$; $M^{-1} \in \mathbb{R}^{n\times n}$;
        $z_1 = M^{-1} r_1$.

        If $r_1 = 0$ then STOP, else compute $p_1 = z_1$.

Step k: Set $s_{k+1} = s_k + \tilde{a}_k p_k,\ \ r_{k+1} = r_k - \tilde{a}_k A p_k$,
        where $\tilde{a}_k = \frac{r_k^T z_k}{p_k^T A p_k}$.
        If $r_{k+1} = 0$ then STOP
        else compute $z_{k+1} = M^{-1} r_{k+1},\ p_{k+1} = z_{k+1} + \tilde{\beta}_k p_k$,
        where $\tilde{\beta}_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k}$.
        Set $k = k + 1$ and go to Step k.
```

With a similar computation which yielded the results in Theorem 3.1, we have the following result concerning the Algorithm Prec-CG.

Theorem 4.1 *Consider the Algorithm* Prec-CG *where A is symmetric and positive definite. Suppose $\|r_i\| \neq 0$, $i \leq h+1$. Suppose the preconditioner M is given, where M is symmetric positive definite. Then, the following relations hold:*

$$\widetilde{P}_h \left[L_h^{(2)}\right]^T = Z_h, \tag{4.1}$$

$$M^{-1} A \widetilde{P}_h = \left(Z_h \ \vdots\ \frac{z_{h+1}}{\|z_{h+1}\|} \right) \begin{pmatrix} L_h^{(1)} \\ l_{h+1,h} e_h^T \end{pmatrix} \tilde{D}_h, \quad h < n, \tag{4.2}$$

$$M^{-1} A Z_h = \left(Z_h \ \vdots\ \frac{z_{h+1}}{\|z_{h+1}\|} \right) \begin{pmatrix} \tilde{T}_h C_h^T \\ \tilde{t}_{h+1,h} e_h^T \end{pmatrix}, \quad h < n, \tag{4.3}$$

where $\widetilde{P}_h = (p_1/\|z_1\| \ \cdots \ p_h/\|z_k\|)$ *and* $Z_h = (z_1/\|z_1\| \ \cdots \ z_h/\|z_h\|)$,

$$L_h^{(2)} = \begin{pmatrix} 1 & & & & \\ -\tilde{\beta}_1 \frac{\|z_1\|}{\|z_2\|} & 1 & & & \\ & \cdot & 1 & & \\ & & \cdot & 1 & \\ & & & -\tilde{\beta}_{h-1} \frac{\|z_{h-1}\|}{\|z_h\|} & 1 \end{pmatrix}, \quad (4.4)$$

$$\tilde{D}_h = \begin{pmatrix} \dfrac{1}{\tilde{a}_1} & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \dfrac{1}{\tilde{a}_h} \end{pmatrix}, \quad (4.5)$$

$$L_h^{(1)} = \begin{pmatrix} 1 & & & & \\ -\frac{\|z_2\|}{\|z_1\|} & 1 & & & \\ & \cdot & 1 & & \\ & & \cdot & 1 & \\ & & & -\frac{\|z_h\|}{\|z_{h-1}\|} & 1 \end{pmatrix}, \quad (4.6)$$

and $C_h = \{c_{i,j}^h\}$ *is given by*

$$c^h_{i,j} = \begin{cases} 0, & i < j, \\ \\ (-1)^{i+j-1} \dfrac{\Pi^h_{k=j}\gamma_k}{\Pi^h_{l=i-1}\gamma_l} \left(\dfrac{\tilde{\beta}_{i-1}}{\gamma_{i-1}} - \gamma_{i-1} \right), & i > j, \\ \\ 1, & i = j \end{cases} \tag{4.7}$$

with $\gamma_i = -\|z_{i+1}\|/\|z_i\|$*. Finally,* $\tilde{T}_h$ *is an irreducible symmetric tridiagonal matrix defined by*

$$\tilde{T}_h \;=\; L^{(1)}_h \tilde{D}_h \left[L^{(1)}_h \right]^T, \tag{4.8}$$

and

$$\tilde{t}_{h+1,h} \;=\; -\frac{1}{\tilde{\alpha}_h} \frac{\|z_{h+1}\|}{\|z_h\|}.$$

Proof: Relation (4.1) follows observing that by the preconditioned CG we have

$$\begin{aligned} \frac{p_1}{\|z_1\|} &= \frac{z_1}{\|z_1\|} \\ \frac{p_2}{\|z_2\|} &= \frac{z_2}{\|z_2\|} + \frac{\tilde{\beta}_1}{\|z_2\|}\|z_1\|\frac{p_1}{\|z_1\|} \\ &\vdots \\ \frac{p_h}{\|z_h\|} &= \frac{z_h}{\|z_h\|} + \frac{\tilde{\beta}_{h-1}}{\|z_h\|}\|z_{h-1}\|\frac{p_{h-1}}{\|z_{h-1}\|}. \end{aligned}$$

Similarly, if $z_{h+1} = 0$ we obtain

$$\begin{aligned} M^{-1}A\frac{p_1}{\|z_1\|} &= \frac{1}{\tilde{\alpha}_1}\left(\frac{z_1}{\|z_1\|} - \frac{\|z_2\|}{\|z_1\|}\frac{z_2}{\|z_2\|} \right) \\ M^{-1}A\frac{p_2}{\|z_2\|} &= \frac{1}{\tilde{\alpha}_2}\left(\frac{z_2}{\|z_2\|} - \frac{\|z_3\|}{\|z_2\|}\frac{z_3}{\|z_3\|} \right) \\ &\vdots \\ M^{-1}A\frac{p_h}{\|z_h\|} &= \frac{1}{\tilde{\alpha}_h}\left(\frac{z_h}{\|z_h\|} \right), \end{aligned}$$

which yield relation $M^{-1}A\widetilde{P}_h = Z_h L_h^{(1)} \tilde{D}_h$. On the other hand, if $z_{h+1} \neq 0$, the latter relation yields $M^{-1}A\widetilde{P}_h = Z_{h+1} L_{h+1}^{(1)} \tilde{D}_h$, where

$$L_{h+1}^{(1)} = \left(\begin{array}{c} L_h^{(1)} \\ \hline -\frac{\|z_{h+1}\|}{\|z_h\|} e_h^T \end{array} \right).$$

Thus, (4.2) holds with $l_{h+1,h} = -\|z_{h+1}\|/\|z_h\|$ and

$$Z_{h+1} = (Z_h \vdots z_{h+1}/\|z_{h+1}\|).$$

Finally, after some manipulations from (4.7) it follows

$$L_h^{(2)} = C_h L_h^{(1)}. \tag{4.9}$$

Then, by multiplying relation $AP_h = Z_{h+1} L_{h+1}^{(1)} \tilde{D}_h$ by the non-singular matrix $[L_{h+1}^{(2)}]^T$, and recalling relations (4.1) and (4.9) we obtain (4.3). $\square$

Now, we consider a preconditioned version of the algorithm FLR planar CG algorithm. We soon realize that, unfortunately, in this case the full storage of the preconditioner is required. Indeed, unlike the coefficients in the Algorithm Prec-CG, in a preconditioned version of the Algorithm FLR, the coefficients 'b_{k-1}' and '$\hat{b}_{k-2}$' at *Step* k_B still depend on the preconditioner, which implies that a preconditioned Algorithm FLR may be hardly obtained for large scale problems.
This implies that the Algorithm FLR may be used only for building and providing the preconditioner, as described previously.

5 Preconditioning Truncated Newton methods

In the previous sections we described how to iteratively construct a preconditioner. Even if this preconditioner can be exploited in different contexts involving the solution of large scale linear systems, our main interest is to fruitfully use it within the framework of Truncated Newton methods, for large scale optimization. Of course, both unconstrained and constrained nonlinear optimization problems are of interest. Here we investigate the use of the preconditioner within Truncated Newton methods for unconstrained optimization, namely for solving the problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function and n is large. In particular we aim at defining some preconditioning strategies based on the preconditioner we introduced in the previous sections, which lead an improvement of the overall efficiency of the method.

The structure of any Truncated Newton method for unconstrained optimization is well known (see e.g. [18]). It is based on two nested loops: the iterations of the Newton method (*outer iterations*) and the iterations of the solvers used, at each outer iteration j, to approximately solving the Newton system (*inner iterations*)

$$H_j \ s = -g_j, \tag{5.1}$$

where $H_j = \nabla^2 f(x_j)$ and $g_j = \nabla f(x_j)$. Thus, since a sequence of linear systems must be solved, in many cases, it is crucial to have at one's disposal a preconditioning strategy, which should enable a considerable computational saving in terms of number of inner iterations. This motives the fact that to define general purpose preconditioning strategies within Truncated Newton methods is usually still considered one the main issues which is worthwhile investigating, especially in dealing with large scale problems. As well known, the main difficulty in this context relies on the fact that, in the large scale setting, the Hessian matrix can not be stored or handled. Hence, any Truncated Newton implementation gains information on the Hessian matrix by means of a routine, which provides the product of the Hessian matrix times a vector. This prevents from using any preconditioner based on the knowledge of the actual elements of the Hessian matrix. Unfortunately, few preconditioners have been proposed up to now which do not require the full Hessian matrix. The first proposal of such a preconditioner is in [17], where a diagonal scaling which uses a diagonal approximation of the Hessian obtained by means of BFGS updating is introduced. Afterwards, in [15, 16] an automatic preconditioning based on m–step limited memory quasi–Newton updating has been proposed without requiring the knowledge of the Hessian matrix. More recently, a preconditioning strategy based on a diagonal scaling, which only gains the information of the Hessian matrix by means of the product of the Hessian times a vector has been proposed in [21]. The other preconditioners up to now proposed, usually require the knowledge of the entries of the Hessian matrix and hence are not suited to be used in large scale setting.

The preconditioner we defined in Section 2, actually is generated during the iterations of the solver, gaining the information on the Hessian matrix as by product of the iterates of the method. Moreover, as already noticed in Section 3, its application does not require to store any $n \times n$ matrix.

5.1 Our preconditioning strategy

The strategy we adopt in solving the sequence of the linear systems (5.1) is based on an adaptive rule for deciding when the application of the preconditioner is fruitful, that is the preconditioned method should be effective, or not. In fact, it is well known that the effectiveness of a preconditioner is problem dependent. Hence, whenever the application of a preconditioner is expected to led to a worsening of the performance, it should be desirable to use an unpreconditioned method. Of course, the key point is how to formulate an adaptive rule for deciding that. The adaptive scheme we adopt at each outer iteration can be summarized as follows:

1. Perform a number $h \leq 10$ of iterations of the *unpreconditioned* algorithm FLR and store P_h, $\tilde{L}_h$ and B_h.
2. Restart the CG iterations and perform h iterations of the *preconditioned* CG method.
3. On the basis of the information gained decide how to proceed:
   - if the preconditioner should be used, continue the preconditioned CG iterations,
   - otherwise continue the unpreconditioned iterates.

Observe that, if during the first h iterations of the algorithm FLR at point 1 the termination criterion is satisfied before performing h iterations, of course the inner iterations are stopped and no preconditioner is considered. Analogously, while performing the preconditioned CG iterates at point 2, if the termination criterion is satisfied before performing h iterations, the inner iterates are stopped.

Whenever the selection rule indicates to proceed with the unpreconditioned iterates, the standard unpreconditioned CG method FLR is simply continued until a termination criterion is satisfied. Otherwise, the standard preconditioned CG algorithm is used. Note that the planar CG algorithm FLR can not be preconditioned since a preconditioned version of this algorithm would need the storage of the full matrix M_h. This is due to the fact that in the *Step* k_B of the FLR algorithm, the coefficients b_{k-1} and b_{k-2} depend on M_h.

Remark 5.1 One of the most relevant point of such a strategy is the fact that, at the j-th outer iteration (i.e. when we are solving the j-th system of the sequence (5.1)) the preconditioner is computed on the basis of the information on the actual Hessian matrix H_j. On the opposite, the strategies usually adopted (see, e.g. [15]) compute the preconditioner, to be used in the j-th outer iteration, during the inner CG iterations used at the previous outer iteration. This means that the preconditioner is computed on the basis of H_{j-1}, that is on the Hessian matrix at the previous outer iteration and in general, this could be a serious drawback in case the Hessian matrix should drastically change from x_{j-1} to x_j.

5.2 The search direction

In this section we describe how the search direction is computed at each outer iteration of the Truncated Newton method. We refer to the application of the algorithm Prec-CG to solve the Newton system (5.1), assuming that the preconditioner M_h was already been computed after h planar CG iterates (of course this also applies to the case when unpreconditioned CG is used, setting $M_h = I$). We recall that we are dealing with the indefinite case, so that particular safeguard is needed in computing the search direction whenever a negative curvature is encountered, namely a direction p such that $p^T H_j p < 0$. Getting our inspiration from the strategy proposed in [12], we do not terminate the CG inner iterations whenever a negative curvature direction is encountered, provided that

$$|p_k^T H_j p_k| > \epsilon \|p_k\|^2,$$

where $\epsilon > 0$ is a suitable parameter. Observe that, if $p_k^T H_j p_k < 0$ for some k, the approximate solution s_{k+1} generated at the k-th CG iteration, could be no longer a descent direction for the quadratic model

$$Q_j(s) = \frac{1}{2} s^T H_j s + g_j^T s.$$

To overcome this drawback, which arises in dealing with non-convex problems, we proceed as follows. Let us consider a generic k-th iteration of the preconditioned CG, and define the following index sets

$$\begin{aligned} I_k^+ &= \left\{ i \in \{1,k\} \ : \ p_i^T H_j p_i > \epsilon_i \|p_i\|^2 \right\}, \\ I_k^- &= \left\{ i \in \{1,k\} \ : \ p_i^T H_j p_i < -\epsilon_i \|p_i\|^2 \right\} \end{aligned}$$

where $|I_k^+| + |I_k^-| = k$, ($|C|$ is the cardinality of the set C) and the following vectors

$$\begin{aligned} s_k^P &= \sum_{i \in I_k^+} \tilde{a}_i p_i = \sum_{i \in I_k^+} \frac{r_i^T M_h^{-1} r_i}{p_i^T H_j p_i} p_i, \\ s_k^N &= -\sum_{i \in I_k^-} \tilde{a}_i p_i = -\sum_{i \in I_k^-} \frac{r_i M_h^{-1} r_i}{p_i^T H_j p_i} p_i. \end{aligned}$$

The direction s_k^P can be viewed as the minimizer of $Q_j(s)$ over the (positive) subspace generated by the vectors p_i with $i \in I_k^+$, and s_k^N is a negative curvature direction. Then, at each preconditioned CG iterate we define the vector $s_k = s_k^P + s_k^N$. With this choice we guarantee the monotonic decrease of $\{Q_j(s_k)\}$ as k increase. In fact the following result holds.

Proposition 5.1 *Suppose that the preconditioned CG algorithm (*Prec-CG*) is applied for solving the system (5.1). At each iteration k consider the following vector*

$$s_k = s_k^P + s_k^N.$$

Then the sequence $\{Q_j(s_k)\}_{k=1,2,\dots}$ is strictly decreasing, i.e.

$$Q_j(s_{k+1}) < Q_j(s_k), \qquad k = 1, 2, \dots.$$

Proof: By definition, we have

$$s_k = s_k^P + s_k^N = \sum_{i=1}^{k} |\tilde{a}_i| p_i = s_{k-1} + |\tilde{a}_k| p_k.$$

Now, using the fact that $r_1 = -g_j$ and $r_1^T p_k = r_k^T M^{-1} r_k$ we obtain

$$Q_j(s_k) = \frac{1}{2} \left[s_{k-1} + |\tilde{a}_k| p_k \right]^T H_j \left[s_{k-1} + |\tilde{a}_k| p_k \right] + g_j^T \left[s_{k-1} + |\tilde{a}_k| p_k \right]$$

$$
\begin{aligned}
&= \frac{1}{2}\left[s_{k-1}^T H_j s_{k-1} + \tilde{a}_h^2 p_k^T H_j p_k\right] + |\tilde{a}_k| p_k^T H_j s_{k-1} \\
&\qquad + g_j^T s_{k-1} + |\tilde{a}_k| g_j^T p_k \\
&= Q(s_{k-1}) + \frac{1}{2}\mathrm{sgn}(p_k^T H_j p_k)\frac{(r_k^T M_h^{-1} r_k)^2}{|p_k^T H_j p_k|} - \frac{r_k^T M_h^{-1} r_k}{|p_k^T H_j p_k|} r_1^T p_k \\
&= Q_j(s_{k-1}) + \left(\frac{1}{2}\mathrm{sgn}(p_k^T H_j p_k) - 1\right)\frac{(r_k^T M_h^{-1} r_k)^2}{|p_k^T H_j p_k|} \\
&< Q_j(s_{k-1}).
\end{aligned}
$$

□

The property stated in Proposition 5.1 plays a fundamental rule concerning the choice of the stopping criterion for the preconditioned CG iterates. In fact, this property allows us to use, also in the preconditioned CG iterates, the standard stopping rule based on the comparison of the reduction in the quadratic model with the average reduction per iteration [19]. Therefore, at the outer iteration j, the inner preconditioned CG iterations are terminated whenever

$$
\frac{Q_j(s_k) - Q_j(s_{k-1})}{Q_j(s_k)/k} \le \alpha \qquad (5.2)
$$

where α is a suited parameter. Observe that setting $M_h = I$, i.e. in the unpreconditioned case, s_k coincides with the choice of the Newton–type direction in [12]. Note that Proposition 5.1, ensuring the monotonic decrease of the quadratic model, enables to extend the use of the stopping criterion (5.2) to the nonconvex cases. Hence, criterion (5.2) may be fruitfully used alternatively to the residual–based standard criterion $\|r_j\|/\|g_j\| \le \eta_j$, where $\eta_j \to 0$ as $j \to \infty$.

5.3 Preconditioned Newton step

Now we aim at proving that the application of the preconditioner defined in Section 3 leads to an interesting property. In the following proposition, we prove that one iteration of the preconditioned CG improves the quadratic model not less than h iterations of the algorithm FLR. For sake of simplicity we assume that no planar steps are performed in the h unpreconditioned iterations needed for computing the preconditioner.

Proposition 5.2 *Let M_h be the preconditioner in (2.4), computed after h iterations of the* FLR *algorithm where no planar steps are performed. Let $p_1, \dots, p_h$ the CG directions generated, with $p_\ell^T H_j p_m = 0$, $1 \le \ell \neq m \le h$, and $p_1 = -g_j$. Consider the vectors*

$$\begin{aligned} s_h^{UN} &= s_1 + \sum_{i=1}^{h} a_i p_i = \sum_{i=1}^{h} a_i p_i \\ s_2^{PR} &= s_1 + |\tilde{a}_1| p_1 = |\tilde{a}_1| M_h^{-1} r_1 \end{aligned}$$

then we have

$$Q(s_2^{PR}) \le Q(s_h) \le Q(s_h^{UN}).$$

Proof: From the definition of M_h^{-1}, we have

$$\begin{aligned} M_h^{-1} r_1 &= \left[(I - R_h R_h^T) + R_h |T|^{-1} R_h^T \right] r_1 \\ &= r_1 - R_h \begin{pmatrix} \|r_1\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} + R_h [L_h |D_h| L_h^T]^{-1} \begin{pmatrix} \|r_1\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= R_h L_h^{-T} |D_h|^{-1} L_h^{-1} \begin{pmatrix} \|r_1\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= R_h L_h^{-T} diag_{1 \le i \le h} \{|a_i|\} \begin{pmatrix} 1 \\ \sqrt{\beta_1} \\ \vdots \\ \sqrt{\beta_1 \cdot \dots \cdot \beta_{h-1}} \end{pmatrix} \|r_1\| \\ &= P_h diag_{1 \le i \le h} \left\{ \frac{\|r_i\|^2}{|p_i^T H_j p_i|} \right\} \begin{pmatrix} \|r_1\| \\ \|r_2\| \\ \vdots \\ \|r_h\| \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \left(\frac{p_1}{\|r_1\|}\frac{p_2}{\|r_2\|}\cdots\frac{p_h}{\|r_h\|}\right)\begin{pmatrix} |a_1| & & \\ & \ddots & \\ & & |a_h| \end{pmatrix}\begin{pmatrix} \|r_1\| \\ \|r_2\| \\ \vdots \\ \|r_h\| \end{pmatrix} \\
&= \sum_{i=1}^{h} |a_i| p_i. \qquad (5.3)
\end{aligned}
$$

The latter relation proves that if $\tilde{a}_1 = 1$, the directions s_h and s_2^{PR} coincide. Now, by definition we have

$$
\begin{aligned}
Q(s_h^{UN}) &= \frac{1}{2}\left(s_h^{UN}\right)^T H_j \left(s_h^{UN}\right) + g^T s_h^{UN} \\
&= \frac{1}{2}\left(\sum_{i=1}^{h} a_i p_i\right)^T H_j \left(\sum_{i=1}^{h} a_i p_i\right) - r_1^T \left(\sum_{i=1}^{h} a_i p_i\right) \\
&= \frac{1}{2}\sum_{i=1}^{h} a_i^2 p_i^T H_j p_i - \sum_{i=1}^{h} a_i \|r_i\|^2 \\
&= \sum_{i=1}^{h}\left[\frac{1}{2} a_i \|r_i\|^2 - a_i \|r_i\|^2\right] \\
&= -\frac{1}{2}\sum_{i=1}^{h} a_i \|r_i\|^2. \qquad (5.4)
\end{aligned}
$$

$$
\begin{aligned}
Q(s_h) &= \frac{1}{2} s_h^T H_j s_h + g^T s_h \\
&= \frac{1}{2}\left(\sum_{i=1}^{h} |a_i| p_i\right)^T H_j \left(\sum_{i=1}^{h} |a_i| p_i\right) - r_1^T \left(\sum_{i=1}^{h} |a_i| p_i\right) \\
&= \frac{1}{2}\sum_{i=1}^{h} a_i^2 p_i^T H_j p_i - \sum_{i=1}^{h} |a_i| \|r_i\|^2 \\
&= \sum_{i=1}^{h}\left[\frac{1}{2} a_i \|r_i\|^2 - |a_i| \|r_i\|^2\right] \\
&= \sum_{i=1}^{h}\left[\frac{1}{2}\mathrm{sgn}(p_i^T H_j p_i) - 1\right] |a_i| \|r_i\|^2. \qquad (5.5)
\end{aligned}
$$

Moreover, from (5.3)

$$\begin{aligned}
|\tilde a_1| &= \left|\frac{r_1^T z_1}{\tilde p_1^T H_j \tilde p_1}\right| = \left|\frac{r_1^T M_h^{-1} r_1}{z_1^T H_j z_1}\right| \\
&= \left|\frac{r_1^T\left(\sum_{i=1}^h |a_i| p_i\right)}{\left(\sum_{i=1}^h |a_i| p_i\right)^T H_j \left(\sum_{i=1}^h |a_i| p_i\right)}\right| = \left|\frac{\displaystyle\sum_{i=1}^h |a_i| \|r_i\|^2}{\displaystyle\sum_{i=1}^h a_i^2 p_i^T H p_i}\right| \\
&= \frac{\sum_{i=1}^h |a_i| \|r_i\|^2}{\left|\sum_{i=1}^h a_i^2 p_i^T H_j p_i\right|};
\end{aligned}$$

thus, we have also

$$\begin{aligned}
Q(s_2^{PR}) &= \frac{1}{2}\left(\frac{\sum_{i=1}^h |a_i| \|r_i\|^2}{\sum_{i=1}^h a_i^2 p_i^T H_j p_i}\right)^2 \left(\sum_{i=1}^h |a_i| p_i\right)^T H_j \left(\sum_{i=1}^h |a_i| p_i\right) \\
&\qquad - \frac{\sum_{i=1}^h |a_i| \|r_i\|^2}{\left|\sum_{i=1}^h a_i^2 p_i^T H_j p_i\right|} r_1^T \left(\sum_{i=1}^h |a_i| p_i\right) \\
&= \frac{1}{2}\frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\sum_{i=1}^h a_i^2 p_i^T H_j p_i} - \frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\left|\sum_{i=1}^h a_i^2 p_i^T H_j p_i\right|} \\
&= \frac{1}{2}\frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\sum_{i=1}^h a_i \|r_i\|^2} - \frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\left|\sum_{i=1}^h a_i \|r_i\|^2\right|}. \qquad (5.6)
\end{aligned}$$

From (5.4) and (5.5), considering that

$$\sum_{i=1}^h \left[-\frac{1}{2}\mathrm{sgn}(p_i^T H_j p_i) - \left(\frac{1}{2}\mathrm{sgn}(p_i^T H_j p_i) - 1\right)\right] |a_i| \|r_i\|^2 \ge 0$$

we easily obtain

$$Q(s_h) \le Q(s_h^{UN}).$$

Moreover, observe that $Q(s_2^{PR}) \le Q(s_h)$ if and only if

$$\frac{1}{2}\frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\sum_{i=1}^h a_i \|r_i\|^2} - \frac{\left[\sum_{i=1}^h |a_i| \|r_i\|^2\right]^2}{\left|\sum_{i=1}^h a_i \|r_i\|^2\right|} \le \frac{1}{2}\sum_{i=1}^h a_i \|r_i\|^2 - \sum_{i=1}^h |a_i| \|r_i\|^2,$$

or equivalently

$$\frac{\frac{1}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 - \operatorname{sgn}\left(\sum_{i=1}^{h}a_i\|r_i\|^2\right)\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2}{\sum_{i=1}^{h}a_i\|r_i\|^2}$$

$$\leq \frac{1}{2}\sum_{i=1}^{h}a_i\|r_i\|^2 - \sum_{i=1}^{h}|a_i|\|r_i\|^2. \tag{5.7}$$

To prove the latter relation we separately consider two cases: the case $\sum_{i=1}^{h}a_i\|r_i\|^2 > 0$ and the case $\sum_{i=1}^{h}a_i\|r_i\|^2 < 0$. In the first case the relation (5.7) holds if and only if

$$\frac{1}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 - \operatorname{sgn}\left(\sum_{i=1}^{h}a_i\|r_i\|^2\right)\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2$$

$$\leq \frac{1}{2}\left[\sum_{i=1}^{h}a_i\|r_i\|^2\right]^2 - \left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]\sum_{i=1}^{h}a_i\|r_i\|^2$$

or equivalently if and only if

$$-\frac{1}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 \leq \left[\frac{1}{2}\sum_{i=1}^{h}a_i\|r_i\|^2 - \sum_{i=1}^{h}|a_i|\|r_i\|^2\right]\sum_{i=1}^{h}a_i\|r_i\|^2,$$

and the latter inequality holds since

$$-\frac{1}{2}\left[\left(\sum_{i=1}^{h}|a_i|\|r_i\|^2\right)^2 + \left(\sum_{i=1}^{h}a_i\|r_i\|^2\right)^2\right] + \sum_{i=1}^{h}|a_i|\|r_i\|^2\sum_{i=1}^{h}a_i\|r_i\|^2 \leq 0.$$

In the second case the relation (5.7) is equivalent to

$$\frac{3}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 \geq \frac{1}{2}\left[\sum_{i=1}^{h}a_i\|r_i\|^2\right]^2 + \left|\sum_{i=1}^{h}a_i\|r_i\|^2\right|\sum_{i=1}^{h}|a_i|\|r_i\|^2,$$

which holds since

$$\frac{3}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 = \frac{1}{2}\left[\sum_{i=1}^{h}|a_i|\|r_i\|^2\right]^2 + \left|\sum_{i=1}^{h}|a_i|\|r_i\|^2\right|\sum_{i=1}^{h}|a_i|\|r_i\|^2$$

$$\geq \frac{1}{2}\left[\sum_{i=1}^{h}a_i\|r_i\|^2\right]^2 + \left|\sum_{i=1}^{h}a_i\|r_i\|^2\right|\sum_{i=1}^{h}|a_i|\|r_i\|^2.$$

This proves that

$$Q(s_2^{PR}) \leq Q(s_h) \leq Q(s_h^{UN}).$$

□

6 Numerical Results

In this section we report the results of a preliminar numerical investigation obtained by embedding the new preconditioner in a linesearch based Truncated Newton method for unconstrained optimization. In particular, we performed a standard implementation of a Truncated Newton method which uses the planar CG algorithm FLR as a tool for constructing the preconditioner. Then the standard preconditioned CG algorithm is used as preconditioned scheme for the inner iteration.

The aim of our numerical investigation is, firstly, to assess if the preconditioning strategy proposed is reliable. The key point is to check if the use of the preconditioner leads to a computational savings in terms of overall inner iterations, with respect to the unpreconditioned case. Therefore, to this aim, in this preliminary testing we do not use any adaptive criterion for choosing if the preconditioner should be used or not at each outer iteration of the method. We simply always use the preconditioned CG for solving the sequence of Newton systems (5.1) whenever the preconditioner has been computed. We recall that the preconditioner is computed whenever at least h iterations of the unpreconditioned FLR algorithm have been performed.

As regards the test problems, we used all the unconstrained large problems contained in the CUTEr collection [11]. All the algorithms were coded in FORTRAN 90 compiled under Compaq Visual Fortran 6.6. All the runs were performed on a PC Pentium 4 – 3.2GHz with 1Gb RAM. The results are reported in terms of number of iterations, number of function evaluations, number of CG-inner iterations and CPU time needed to solve each problem. In Tables 6.1, 6.2 and 6.3 we report the complete results for the unpreconditioned Truncated Newton method. In Tables 6.4, 6.5

and 6.6 we report the complete results for the preconditioned version of the same algorithm. As regards the parameter h, we tried different values ranging from 5 to 10 and $h = 7$ is a value which seems to be a good trade–off between the computational burden required to compute the preconditioner and its effectiveness.

By comparing these results obtained by means of the preconditioned algorithm with respect to those obtained without using any preconditioner, it is very clear that in most cases the use of the preconditioner is beneficial, since it enables a significant reduction of the inner iterations needed to solve the problems. In particular, in terms of number of inner iterations, on 36 test problems the preconditioned algorithm performs better than the unpreconditioned one, and only on 10 problems it performs worse. On the remaining test problems the two algorithms led to the same results or (on 8 problems) they converge to different minimizers.

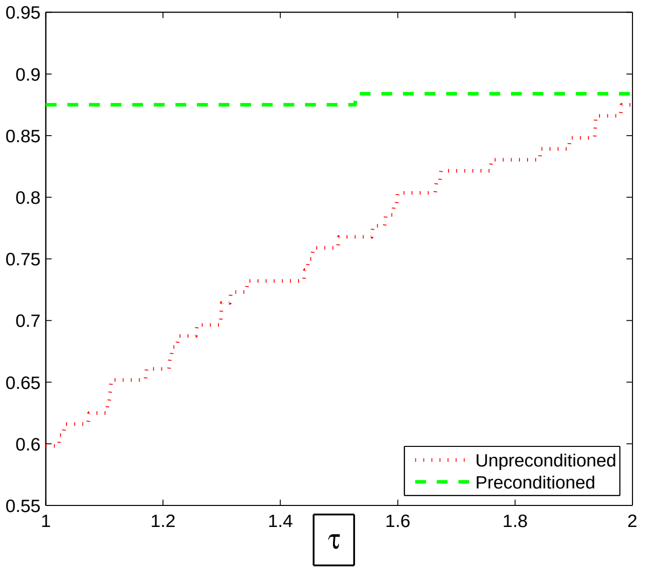
The performances of the two algorithms, in terms of number of inner iterations, are also compared by means of the performance profiles [3]. In particular, in Figure 6.1 we drew the performance profiles relative to the results reported in the previous tables. Also from these figures it is clear that the preconditioned algorithm performs the best in terms of number of inner iterations. However, it is important to note that the preconditioned algorithm leads to 6 additional failures with respect to the unpreconditioned one, due to an excessive number of outer iterations (> 10000) of CPU time (> 900 seconds). This evidences the fact that, in some cases, the application of the preconditioner is not beneficial and even leads to a severe worsening. Hence the necessity to define an adaptive criterion which enables to assess whenever it is fruitful to use the preconditioner or not.

7 Conclusions

In this paper we propose a new preconditioning technique for efficiently solving symmetric indefinite linear systems arising in large scale optimization, within the framework of Newton–type methods. Krylov subspace methods are considered for the iterative solution of the linear systems, and the construction of the preconditioner is obtained as by product of the Krylov methods iterates. In fact, the preconditioner is built by means of an iterative matrix decomposition of the system matrix, without requiring to store or handle the system matrix. The only information on this matrix is gained by means of a routine which computes the product of the matrix times a vector. The use of a planar version of the Conjugate Gradient algorithm also enables to tackle indefinite linear systems arising from nonconvex optimization problems. In order to assess if the preconditioning strategy proposed is reliable, we embed it within a linesearch based Truncated Newton method for unconstrained optimization. The results obtained showed that the proposed strategy is efficient, enabling a significant computational saving in terms of number of inner iterations need to solve the problems. Unfortunately, few additional failures are obtained by using the preconditioned algorithm. This should be due to the necessity of an adaptive criterion, which enables to decide whenever the preconditioner is beneficial and hence worthwhile to be applied or not. This will be the subject of future work.

Acknowledgement

The first author wishes to thank the “Programma Ricerche INSEAN 2007-2009”.



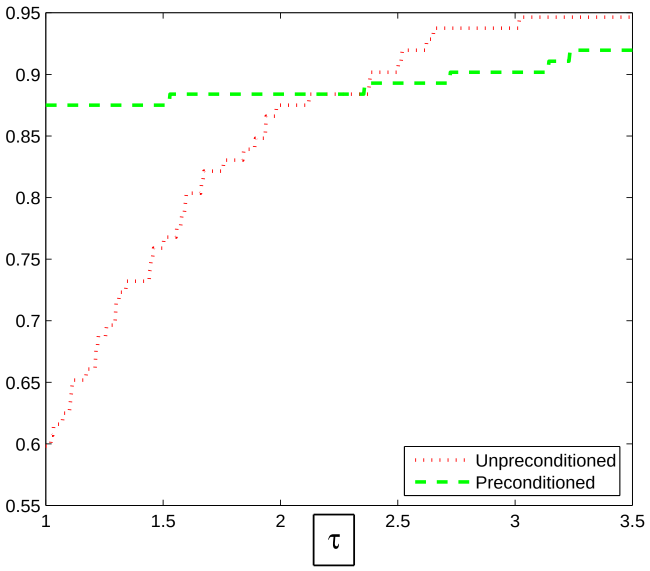


Figure 6.1: Performance Profile w.r.t. inner iterations.

References

[1] J. Baglama, D. Calvetti, G. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM Journal on Scientific Computing, 20 (1998), pp. 243–269.

[2] R. Bank and T. Chan, *A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems*, Numerical Algorithms, 7 (1994), pp. 1–16.

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

[4] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, Journal of Computational and Applied Mathematics, 69 (1996), pp. 303–318.

[5] G. Fasano, *Use of conjugate directions inside Newton–type algorithms for large scale unconstrained optimization*, PhD thesis, Università di Roma "La Sapienza", Rome, Italy, 2001.

[6] G. Fasano, *Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 2: Application*, Journal of Optimization Theory and Applications, 125 (2005), pp. 543–558.

[7] G. Fasano, *Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 1: Theory*, Journal of Optimization Theory and Applications, 125 (2005), pp. 523–541.

[8] G. Fasano, *Lanczos-conjugate gradient method and pseudoinverse computation, in unconstrained optimization*, Journal of Optimization Theory and Applications, 132 (2006), pp. 267–285.

[9] G. Fasano and M. Roma, *Iterative computation of negative curvature directions in large scale optimization*, Computational Optimization and Applications, 38 (2007), pp. 81–104.

[10] G. Golub and C. Van Loan, *Matrix Computations*, The John Hopkins Press, Baltimore, 1996. Third edition.

[11] N. I. M. Gould, D. Orban, and P. L. Toint, CUTE *(and* sifdec*), a constrained and unconstrained testing environment, revised*, ACM Transaction on Mathematical Software, 29 (2003), pp. 373–394.

[12] L. Grippo, F. Lampariello, and S. Lucidi, *A truncated Newton method with nonmonotone linesearch for unconstrained optimization*, Journal of Optimization Theory and Applications, 60 (1989), pp. 401–419.

[13] M. Hestenes, *Conjugate Direction Methods in Optimization*, Springer Verlag, New York, 1980.

[14] L. Loghin, D. Ruiz, and A. Touhami, *Adaptive preconditioners for nonlinear systems of equations*, Journal of Computational and Applied Mathematics, 189 (2006), pp. 362–374.

[15] J. Morales and J. Nocedal, *Automatic preconditioning by limited memory quasi–Newton updating*, SIAM Journal on Optimization, 10 (2000), pp. 1079–1096.

[16] J. L. Morales and J. Nocedal, *Algorithm* PREQN*: Fortran 77 subroutine for preconditioning the conjugate gradient method*, ACM Transaction on Mathematical Software, 27 (2001), pp. 83–91.

[17] S. Nash, *Preconditioning of truncated-Newton methods*, SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 599–616.

[18] S. Nash, *A survey of truncated-Newton methods*, Journal of Computational and Applied Mathematics, 124 (2000), pp. 45–59.

[19] S. Nash and A. Sofer, *Assessing a search direction within a truncated-Newton method*, Operations Research Letter, 9 (1990), pp. 219–221.

[20] J. Nocedal, *Large scale unconstrained optimization*, in The state of the art in Numerical Analysis, A. Watson and I. Duff, eds., Oxford, 1997, Oxford University Press, pp. 311–338.

[21] M. Roma, *A dynamic scaling based preconditioning for truncated newton methods in large scale unconstrained optimization*, Optimization Methods and Software, 20 (2005), pp. 693–713.

[22] J. Stoer, *Solution of large linear systems of equations by conjugate gradient type methods*, in Mathematical Programming. The State of the Art, A. Bachem, M.Grötschel, and B. Korte, eds., Berlin Heidelberg, 1983, Springer-Verlag, pp. 540–565.

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
ARWHEAD	1000	34	364	37	0.000000D+00	0.08
ARWHEAD	10000	10	102	11	1.332134D-11	0.22
BDQRTIC	1000	50	297	103	3.983818D+03	0.11
BDQRTIC	10000	461	****	2244	4.003431D+04	41.27
BROYDN7D	1000	449	1972	2686	3.823419D+00	2.52
BROYDN7D	10000	884	4058	4575	3.711778D+03	54.77
BRYBND	1000	21	65	29	9.588198D-14	0.02
BRYBND	10000	21	65	29	8.686026D-14	0.42
CHAINWOO	1000	262	440	1004	1.000000D+00	0.64
CHAINWOO	10000	287	823	911	1.000000D+00	8.78
COSINE	1000	22	64	40	-9.990000D+02	0.03
COSINE	10000	19	65	32	-9.999000D+03	0.27
CRAGGLVY	1000	46	213	110	3.364231D+02	0.14
CRAGGLVY	10000	115	775	177	3.377956D+03	3.45
CURLY10	1000	151	438	7638	-1.003163D+05	2.94
CURLY10	10000	135	1295	55801	-1.003163D+06	318.88
CURLY20	1000	267	470	7670	-1.001379D+05	4.91
CURLY20	10000	162	994	64081	-1.001313D+06	510.39
CURLY30	1000	>10000	-	-	-	-
DIXMAANA	1500	8	13	9	1.000000D+00	0.00
DIXMAANA	3000	8	14	8	1.000000D+00	0.02
DIXMAANB	1500	5	10	6	1.000000D+00	0.02
DIXMAANB	3000	5	10	6	1.000000D+00	0.02
DIXMAANC	1500	5	11	6	1.000000D+00	0.00
DIXMAANC	3000	5	11	6	1.000000D+00	0.05
DIXMAAND	1500	5	8	5	1.000000D+00	0.00
DIXMAAND	3000	5	8	5	1.000000D+00	0.00
DIXMAANE	1500	63	66	204	1.000000D+00	0.23
DIXMAANE	3000	88	91	277	1.000000D+00	0.70
DIXMAANF	1500	33	38	198	1.000000D+00	0.23
DIXMAANF	3000	32	37	238	1.000000D+00	0.58
DIXMAANG	1500	41	80	223	1.000000D+00	0.25
DIXMAANG	3000	65	133	400	1.000000D+00	1.09
DIXMAANH	1500	26	28	194	1.000000D+00	0.19
DIXMAANH	3000	28	30	256	1.000000D+00	0.66
DIXMAANI	1500	60	63	1997	1.000000D+00	1.81
DIXMAANI	3000	83	86	2585	1.000000D+00	5.23
DIXMAANJ	1500	58	118	213	1.089260D+00	0.22
DIXMAANJ	3000	63	135	291	1.176995D+00	0.75
DIXMAANK	1500	24	38	335	1.000000D+00	0.34
DIXMAANK	3000	28	41	333	1.000000D+00	0.78
DIXMAANL	1500	34	36	1648	1.000000D+00	1.59
DIXMAANL	3000	30	32	282	1.000000D+00	0.67

Table 6.1: The results for the *unpreconditioned* Truncated Newton method. Part A

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
DQDRTIC	1000	33	274	34	7.461713D-26	0.06
DQDRTIC	10000	102	868	103	2.426640D-27	2.02
DQRTIC	1000	22	81	40	2.784985D-02	0.00
DQRTIC	10000	31	111	60	4.932478D-01	0.28
EDENSCH	1000	21	89	27	6.003285D+03	0.00
EDENSCH	10000	18	85	23	6.000328D+04	0.42
ENGVAL1	1000	11	34	16	1.108195D+03	0.02
ENGVAL1	10000	12	36	19	1.109926D+04	0.22
FLETCBV2	1000	1	1	1	-5.013384D-01	0.00
FLETCBV2	10000	1	1	1	-5.001341D-01	0.02
FLETCBV3	1000	11	11	33	-4.050270D+04	0.02
FLETCBV3	10000	183	183	398	-3.519402D+09	5.34
FLETCHCR	1000	50	342	116	6.392434D-10	0.14
FLETCHCR	10000	116	1085	179	3.525524D-11	3.81
FMINSURF	1024	32	88	1193	1.000000D+00	1.44
FMINSURF	5625	29	122	5062	1.000000D+00	32.72
FREUROTH	1000	38	300	50	1.214697D+05	0.08
FREUROTH	10000	107	1052	119	1.216521D+06	3.89
GENHUMPS	1000	829	3041	5090	6.634813D-11	6.06
GENHUMPS	10000	4682	20484	37195	2.053170D-10	422.52
GENROSE	1000	838	2978	6209	1.000000D+00	4.11
GENROSE	10000	8114	28447	62732	1.000000D+00	569.25
LIARWHD	1000	42	251	61	8.352643D-19	0.08
LIARWHD	10000	112	1107	133	1.455368D-20	2.78
MOREBV	1000	6	6	70	9.088088D-09	0.03
MOREBV	10000	2	2	7	2.428066D-09	0.06
MSQRTALS	1024	178	549	6074	3.676555D-07	52.75
MSQRTBLS	1024	176	560	4545	1.281827D-07	39.78
NONCVXUN	1000	180	664	6040	2.325913D+03	5.27
NONCVXUN	10000	2173	10978	16270	2.323860D+04	185.31
NONCVXU2	1000	191	878	2028	2.317579D+03	1.98
NONCVXU2	10000	1869	9496	11198	2.316937D+04	131.89
NONDIA	1000	22	256	27	6.680969D-21	0.11
NONDIA	10000	78	1515	82	5.507180D-13	7.19
NONDQUAR	1000	43	112	215	1.135243D-04	0.08
NONDQUAR	10000	50	182	208	1.745194D-04	1.06
PENALTY1	1000	31	32	59	9.686175D-03	0.03
PENALTY1	10000	54	81	121	9.900151D-02	0.95
POWELLSG	1000	46	257	86	1.992056D-08	0.06
POWELLSG	10000	114	783	151	7.735314D-08	1.09
POWER	1000	56	180	221	2.472989D-09	0.08
POWER	10000	139	797	683	2.426355D-09	3.22
QUARTC	1000	22	81	40	2.784985D-02	0.03
QUARTC	10000	31	111	60	4.932478D-01	0.36
SCHMVETT	1000	14	35	37	-2.994000D+03	0.08
SCHMVETT	10000	19	69	38	-2.999400D+04	0.91

Table 6.2: The results for the *unpreconditioned* Truncated Newton method. Part B

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
SINQUAD	1000	37	310	49	-2.942505D+05	0.06
SINQUAD	10000	104	1517	111	-2.642315D+07	3.61
SPARSINE	1000	110	586	2942	2.759912D-04	3.02
SPARSINE	10000	412	2303	52888	3.740182D-03	739.14
SPARSQUR	1000	22	66	34	6.266490D-09	0.05
SPARSQUR	10000	22	67	39	1.069594D-08	0.91
SPMSRTLS	1000	56	219	211	6.219291D+00	0.20
SPMSRTLS	10000	5333	5796	62357	5.697109D+01	809.02
SROSENBR	1000	35	309	40	2.842418D-22	0.03
SROSENBR	10000	104	920	108	9.421397D-12	1.20
TESTQUAD	1000	174	723	2823	7.538349D-13	1.06
TOINTGSS	1000	2	3	1	1.001002D+01	0.00
TOINTGSS	10000	2	3	1	1.000100D+01	0.03
TQUARTIC	1000	21	185	27	3.767509D-10	0.03
TQUARTIC	10000	14	144	18	1.145916D-11	0.27
TRIDIA	1000	68	459	927	1.141528D-13	0.34
TRIDIA	10000	176	1803	3756	7.993474D-14	14.91
VARDIM	1000	37	37	72	1.058565D-20	0.05
VARDIM	10000	>10000	-	-	-	-
VAREIGVL	1000	20	45	167	3.903597D-10	0.17
VAREIGVL	10000	21	179	22	3.924839D-16	1.05
WOODS	1000	64	377	141	3.857513D-08	0.12
WOODS	10000	139	1095	223	5.031534D-08	2.67

Table 6.3: The results for the *unpreconditioned* Truncated Newton method. Part C

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
ARWHEAD	1000	34	364	37	0.000000D+00	0.09
ARWHEAD	10000	10	102	11	1.332134D-11	0.22
BDQRTIC	1000	50	297	103	3.983818D+03	0.11
BDQRTIC	10000	461	12751	2244	4.003431D+04	38.89
BROYDN7D	1000	459	1756	1702	3.823419D+00	2.31
BROYDN7D	10000	892	4285	3408	3.661974D+03	55.88
BRYBND	1000	21	65	29	9.588198D-14	0.05
BRYBND	10000	21	65	29	8.686026D-14	0.41
CHAINWOO	1000	154	331	507	1.000000D+00	0.42
CHAINWOO	10000	297	824	883	1.000000D+00	9.27
COSINE	1000	22	64	40	-9.990000D+02	0.03
COSINE	10000	19	65	32	-9.999000D+03	0.27
CRAGGLVY	1000	49	216	94	3.364231D+02	0.14
CRAGGLVY	10000	116	776	173	3.377956D+03	3.52
CURLY10	1000	>10000	-	-	-	-
CURLY10	10000	>10000	-	-	-	-
CURLY20	1000	>10000	-	-	-	-
CURLY20	10000	>10000	-	-	-	-
CURLY30	1000	>10000	-	-	-	-
DIXMAANA	1500	8	13	9	1.000000D+00	0.02
DIXMAANA	3000	8	14	8	1.000000D+00	0.03
DIXMAANB	1500	5	10	6	1.000000D+00	0.02
DIXMAANB	3000	5	10	6	1.000000D+00	0.02
DIXMAANC	1500	5	11	6	1.000000D+00	0.03
DIXMAANC	3000	5	11	6	1.000000D+00	0.03
DIXMAAND	1500	5	8	5	1.000000D+00	0.00
DIXMAAND	3000	5	8	5	1.000000D+00	0.02
DIXMAANE	1500	76	79	168	1.000000D+00	0.30
DIXMAANE	3000	114	117	258	1.000000D+00	1.11
DIXMAANF	1500	52	57	136	1.000000D+00	0.31
DIXMAANF	3000	54	59	143	1.000000D+00	0.78
DIXMAANG	1500	43	86	121	1.000000D+00	0.30
DIXMAANG	3000	75	146	251	1.000000D+00	1.20
DIXMAANH	1500	54	56	134	1.000000D+00	0.42
DIXMAANH	3000	74	76	209	1.000000D+00	1.22
DIXMAANI	1500	235	238	762	1.000001D+00	1.92
DIXMAANI	3000	218	221	686	1.000002D+00	3.50
DIXMAANJ	1500	64	117	164	1.086254D+00	0.36
DIXMAANJ	3000	78	171	224	1.165166D+00	1.69
DIXMAANK	1500	60	74	173	1.000000D+00	0.52
DIXMAANK	3000	62	75	199	1.000000D+00	1.05
DIXMAANL	1500	53	55	130	1.000001D+00	0.38
DIXMAANL	3000	55	57	149	1.000000D+00	0.88

Table 6.4: The results for the *preconditioned* Truncated Newton method. Part A

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
DQDRTIC	1000	33	274	34	7.461713D-26	0.05
DQDRTIC	10000	102	868	103	2.426640D-27	1.98
DQRTIC	1000	22	81	40	2.784985D-02	0.02
DQRTIC	10000	31	111	60	4.932478D-01	0.34
EDENSCH	1000	21	89	27	6.003285D+03	0.05
EDENSCH	10000	18	85	23	6.000328D+04	0.41
ENGVAL1	1000	11	34	16	1.108195D+03	0.02
ENGVAL1	10000	12	36	19	1.109926D+04	0.28
FLETCBV2	1000	1	1	1	-5.013384D-01	0.00
FLETCBV2	10000	1	1	1	-5.001341D-01	0.00
FLETCBV3	1000	9	9	22	-8.470587D+04	0.02
FLETCBV3	10000	111	111	132	-6.590391D+09	1.39
FLETCHCR	1000	58	350	105	1.153703D-09	0.14
FLETCHCR	10000	123	1091	161	2.483558D-08	3.16
FMINSURF	1024	68	172	210	1.000000D+00	0.56
FMINSURF	5625	209	556	664	1.000000D+00	11.98
FREUROTH	1000	38	300	50	1.214697D+05	0.08
FREUROTH	10000	107	1052	119	1.216521D+06	3.27
GENHUMPS	1000	801	2848	2894	1.715389D-11	5.61
GENHUMPS	10000	5318	9264	15635	1.282590D-13	390.52
GENROSE	1000	849	2596	2608	1.000000D+00	4.38
GENROSE	10000	8090	24307	24913	1.000000D+00	524.05
LIARWHD	1000	42	251	61	8.352643D-19	0.09
LIARWHD	10000	112	1107	133	1.455368D-20	2.55
MOREBV	1000	8	8	28	2.148161D-08	0.03
MOREBV	10000	2	2	7	2.428066D-09	0.06
MSQRTALS	1024	4726	4955	16550	3.407724D-05	342.20
MSQRTBLS	1024	3110	3337	10724	1.119480D-05	229.73
NONCVXUN	1000	676	1245	2279	2.327300D+03	5.22
NONCVXUN	10000	2765	12132	11297	2.330632D+04	195.00
NONCVXU2	1000	462	1176	1674	2.319022D+03	3.34
NONCVXU2	10000	2176	10275	8905	2.319615D+04	146.28
NONDIA	1000	22	256	27	6.680969D-21	0.12
NONDIA	10000	78	1515	82	5.507180D-13	6.62
NONDQUAR	1000	45	111	111	1.425631D-04	0.08
NONDQUAR	10000	46	175	98	3.744366D-04	0.94
PENALTY1	1000	31	32	59	9.686175D-03	0.05
PENALTY1	10000	54	81	121	9.900151D-02	0.94
POWELLSG	1000	46	257	86	1.992056D-08	0.05
POWELLSG	10000	114	783	151	7.735314D-08	1.17
POWER	1000	65	189	142	5.912729D-09	0.16
POWER	10000	243	901	616	7.784205D-09	6.06
QUARTC	1000	22	81	40	2.784985D-02	0.03
QUARTC	10000	31	111	60	4.932478D-01	0.36
SCHMVETT	1000	14	35	37	-2.994000D+03	0.06
SCHMVETT	10000	19	69	38	-2.999400D+04	0.86

Table 6.5: The results for the *preconditioned* Truncated Newton method. Part B

PROBLEM	n	ITER	FUNCT	CG-it	F. VALUE	TIME
SINQUAD	1000	37	310	49	-2.942505D+05	0.11
SINQUAD	10000	104	1517	111	-2.642315D+07	3.55
SPARSINE	1000	2718	3138	9243	1.271332D-02	27.39
SPARSINE	10000	-	-	-	-	>900
SPARSQUR	1000	22	66	34	6.266490D-09	0.06
SPARSQUR	10000	22	67	39	1.069594D-08	0.89
SPMSRTLS	1000	62	218	132	6.219291D+00	0.25
SPMSRTLS	10000	-	-	-	-	>900
SROSENBR	1000	35	309	40	2.842418D-22	0.03
SROSENBR	10000	104	920	108	9.421397D-12	1.38
TESTQUAD	1000	>10000	-	-	-	-
TOINTGSS	1000	2	3	1	1.001002D+01	0.00
TOINTGSS	10000	2	3	1	1.000100D+01	0.05
TQUARTIC	1000	21	185	27	3.767509D-10	0.03
TQUARTIC	10000	14	144	18	1.145916D-11	0.28
TRIDIA	1000	431	822	1416	8.189019D-12	1.55
TRIDIA	10000	3618	5245	12146	1.026049D-11	142.69
VARDIM	1000	37	37	72	1.058565D-20	0.06
VARDIM	10000	>10000	-	-	-	-
VAREIGVL	1000	39	64	127	1.491460D-09	0.30
VAREIGVL	10000	21	179	22	3.924839D-16	1.09
WOODS	1000	64	377	141	3.857513D-08	0.16
WOODS	10000	139	1095	223	5.031534D-08	2.81

Table 6.6: The results for the *preconditioned* Truncated Newton method. Part C