Optimization Methods and Software
Vol. 20, No. 6, December 2005, 693–713

Taylor & Francis
Taylor & Francis Group

Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization

MASSIMO ROMA*

Dipartimento di Informatica e Sistemistica 'A. Ruberti', Università di Roma 'La Sapienza', via Buonarroti, 12, Roma 00185, Italy

(Received 7 April 2003; revised 24 November 2003; in final form 21 May 2004)

This paper deals with the preconditioning of truncated Newton methods for the solution of large scale nonlinear unconstrained optimization problems. We focus on preconditioners which can be naturally embedded in the framework of truncated Newton methods, i.e. which can be built without storing the Hessian matrix of the function to be minimized, but only based upon information on the Hessian obtained by the product of the Hessian matrix times a vector. In particular we propose a diagonal preconditioning which enjoys this feature and which enables us to examine the effect of diagonal scaling on truncated Newton methods. In fact, this new preconditioner carries out a scaling strategy and it is based on the concept of equilibration of the data in linear systems of equations. An extensive numerical testing has been performed showing that the diagonal preconditioning strategy proposed is very effective. In fact, on most problems considered, the resulting diagonal preconditioned truncated Newton method performs better than both the unpreconditioned method and the one using an automatic preconditioner based on limited memory quasi-Newton updating (PREQN) recently proposed by Morales and Nocedal [Morales, J.L. and Nocedal, J., 2000, Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, **10**, 1079–1096].

Keywords: Truncated Newton method; Conjugate gradient (CG) method; Preconditioning; Row-column scaling; Equilibrated matrix

1. Introduction

This paper deals with large scale nonlinear unconstrained optimization problems, i.e. problems of finding a local minimizer of a real valued function $f: \mathbb{R}^n \to \mathbb{R}$, namely

$$\begin{aligned} &\min f(x) \\ &x \in \mathbb{R}^n, \end{aligned}$$

where the number of variables n is large. The function f is assumed to be twice continuously differentiable.

The growing interest in solving large scale problems is mainly due to the fact that in many and different contexts and applications such problems arise very frequently. Moreover, as well known, efficiently solving unconstrained problems is very important in the framework of constrained optimization too, for instance when a penalty-barrier approach is used.

*Corresponding author. Email: roma@dis.uniroma1.it

Optimization Methods and Software
ISSN 1055-6788 print/ISSN 1029-4937 online © 2005 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/10556780410001727709

Among the existing methods in large scale unconstrained optimization [see, e.g. refs. 1,2 for a survey] the truncated Newton methods represent one of the most powerful, reliable and flexible tools for solving large scale problems. In fact, they have a solid convergence theory, they retain the good Newton-like convergence properties exhibiting an excellent rate of convergence; moreover, they are robust and efficient also in solving 'difficult problems' as highly nonlinear and ill-conditioned problems. Moreover, the truncated Newton methods have been successfully used in solving real world problems arising in many applications and several algorithmic enhancements have been developed and studied in the last years, too; see the excellent survey of truncated Newton methods [3] recently published by Nash and the references reported therein.

Although the truncated Newton methods have been widely studied and extensively tested a number of important open questions still exist [see ref. 1, section 3.2] and motivate this work. One of these open questions addresses the formulation of an effective preconditioning strategy which enables to improve the behavior of the truncated Newton method in tackling large scale problems. In this paper this question is considered, focusing on general purpose preconditioners suitable within the truncated Newton method framework, that is preconditioners which do not require the explicit knowledge of the Hessian matrix. The information about the Hessian matrix are gained only by means of a routine which provides us with the matrix–vector products of the Hessian matrix times a vector, i.e. by using a tool already required by any truncated Newton method implementation. Of course, this makes this kind of preconditioners also suitable within discrete truncated Newton methods where these matrix–vector products are approximated by finite differences [4]. Moreover, we focus our attention on preconditioners which are also 'dynamic', i.e. which change during the iterations of the method.

Very few preconditioners enjoying all these features have been developed up to now. The most recent and efficient proposal of such a preconditioner is due to Morales and Nocedal [5] which proposed an automatic preconditioning strategy based on limited memory quasi-Newton updating.

In this work, we propose a diagonal preconditioning strategy which enjoys all these desired features. It is based on the equilibration of data in linear systems and it consists of scaling the column vectors of the Hessian matrix. Generally speaking, column scaling is carried out by dividing each column of the matrix by the norm of the column, where different norms may be considered. Equilibrating matrices is a topic of great importance in the numerical solution of linear systems, and, as far as the author is aware, it has not yet been fully exploited as a useful tool for building good general purpose preconditioners for the truncated Newton methods, even though the optimal properties of the scaled matrices – in terms of condition numbers – were well known since thirty years ago [see, e.g. refs. 6,7]. The need of equilibrating matrices in solving linear systems arising from real world problems is, in general, very evident; in fact, many times, these matrices have entries that vary over many orders of magnitude and they are very spread. In this case, a simple diagonal scaling would greatly reduce the condition number of the matrices. These general considerations are of fundamental importance in the framework of truncated Newton methods, since at each iteration of these methods a linear system must be (approximately) solved.

More specifically, we define the diagonal preconditioner $M = \mathrm{diag}\{\|w_1\|_1, \ldots, \|w_n\|_1\}$, where $\|w_j\|_1$ denote the ℓ_1-norm of the jth column vector w_j of the Hessian matrix (with some modifications in order to avoid too small elements). By using this diagonal preconditioner, we obtain a system with a column equilibrated matrix in the ℓ_1-norm. At each outer iteration of a truncated Newton method we approximately compute this preconditioner by means of a single product of the Hessian matrix times the vector $e = (1, 1, \ldots, 1)^{\mathrm{T}}$, thus limiting the additional cost only to an extra matrix–vector product, for each outer iteration.

We embedded the diagonal preconditioner proposed within a linesearch based implementation of the truncated Newton method and firstly, to check its reliability, we numerically tested it on some real problems arising as finite dimensional approximation models taken from the MINPACK-2 collection [8]. Then, we performed an extensive numerical testing on a very large set of test problems of the CUTE collection [9]. The results obtained confirmed our expectation: in fact this preconditioning strategy proved inexpensive and very effective, reducing both the number of conjugate gradient (CG) iterations and the CPU time needed for solving most of the problems considered with respect to the unpreconditioned method and, in some cases, the gain is impressive. An investigation on the eigenvalues distribution clearly showed the clustering effect due to the preconditioner. Moreover, a comparison with the results obtained on the same set of test problems by using the automatic preconditioner based on limited memory quasi-Newton updating PREQN proposed in ref. [5] has been performed. Also from this comparison, it is clearly pointed out that the efficiency of the proposed diagonal preconditioner, in particular, a reduction of the number of the inner iterations is showed and this is always joined with a savings of CPU time while, as regards PREQN, an increase of the time needed is often observed even on those problems, where a reduction of the number of inner iterations is obtained, due to the cost of the quasi-Newton preconditioner.

The paper is organized as follows: in section 2 the truncated Newton methods are briefly recalled pointing out those questions which are still considered open problems and, in particular, concerning preconditioning. In section 3, the new diagonal preconditioning strategy is described and in section 4, the results of the extensive numerical investigation are reported.

Throughout the paper the following notations are used: $g(x) = \nabla f(x)$ denotes the gradient and $H(x) = \nabla^2 f(x)$ the Hessian matrix of the function f. Moreover, $\|v\|_p$ denotes the ℓ_p-norm of a vector $v \in \mathbb{R}^n$ and for a $n \times n$ matrix A, $\|A\|_p$ denotes the induced matrix norm. Moreover, $\kappa_p(A) = \|A\|_p \|A^{-1}\|_p$ denotes the condition number of A in the ℓ_p-norm.

2. Preconditioned truncated Newton methods

In this section, the truncated Newton methods (TN) are briefly recalled, focusing on their major components and in particular on preconditioning, a more precise description of truncated Newton methods can be found in ref. [10] or, e.g. in the survey paper [3].

As well known, given a guess x_k of a solution $x_\star$, the truncated Newton method is based on the quadratic model of the objective function f given by

$$q_k(d) = f(x_k) + g(x_k)^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} H(x_k) d \tag{1}$$

and it is defined by iterations of the form

$$x_{k+1} = x_k + d_k,$$

where the search direction d_k is obtained by approximately minimizing the quadratic model of the objective function (1) over $\mathbb{R}^n$. Of course, whenever the Hessian matrix $H(x_k)$ is positive definite, to minimize the quadratic model q_k is equivalent to solve the linear system

$$H(x_k) d = -g(x_k). \tag{2}$$

A good convergence rate of the method is still ensured by using a particular trade-off rule between the computational burden required to solve the system (2) and the accuracy with

which it is solved [11]. A natural measure of this accuracy is the relative residual. In the truncated Newton methods, an approximation of the Newton direction d_k is computed by applying an iterative method for approximately solving the linear system (2) and 'truncating' the iterates whenever a required accuracy is achieved [10]. To this aim, the most frequently used algorithm is the linear CG method since it is an efficient iterative method for solving positive definite linear systems.

The components of the truncated Newton method are several, providing the method with a great flexibility. In particular, three key aspects for the overall efficiency of the truncated Newton method can be still considered open problems [see ref. 1, section 3.2]: (i) the formulation of an effective 'truncation criterion' for the inner loop; (ii) how to handle the case when the Hessian matrix is not positive definite; (iii) the formulation and handling of a preconditioner to accelerate the convergence in the inner CG iterates.

In this paper we concentrate on preconditioning, which is considered one of the most important issues of the current research on the truncated Newton methods. As well known, preconditioning the inner CG iterations can considerably accelerate the convergence of the CG method affecting the overall efficiency of the truncated Newton method. In fact, the convergence rate of the CG inner iterations is based on the condition number and the number of distinct eigenvalues of the Hessian matrix $H(x_k)$ [see, e.g. ref. 12]. In particular, the larger the condition number, the slower is the convergence of the method; moreover, the more the eigenvalues are clustered, the sooner the error is reduced. Preconditioning enables to reduce either the condition number and the number of distinct eigenvalues thus accelerating the convergence of the CG method. More precisely, if M denotes the preconditioner, the convergence depends on the condition number $\kappa_2(H(x_k)M^{-1})$ and on the number of distinct eigenvalues of $H(x_k)M^{-1}$.

It is out of the scope of this paper to discuss the well known features of preconditioning or to survey the very broad field of preconditioners [see, e.g. refs. 13–17]. We mention in the sequel of this section, the main preconditioning strategies up to now proposed in the framework of truncated Newton methods.

A preconditioning strategy which has been successfully used within the truncated Newton methods is based on the incomplete (modified) Cholesky factorization [18–20]. It is a valuable choice since it is general purpose and a great reduction of the number of the inner CG iterations can be obtained. However, unfortunately, it requires the knowledge of the Hessian matrix. Another approach is based on efficiently exploiting the problem structure when it exists, in particular, many large scale problems possess the so called 'partial separability' property, i.e. the objective function can be written as sum of simpler functions and this property can be exploited for defining 'ad hoc' preconditioners [21].

In the context of truncated Newton methods, a first key point in developing an efficient preconditioning strategy for solving the sequence of systems (2) is to use a 'dynamic preconditioner', i.e. a preconditioner which changes with the outer iterations. In particular, since at each outer iteration k the Hessian matrix changes, consequently the preconditioner should be changed. One possibility is to define, at each outer iteration k, a new preconditioner. This idea is not new, and its importance, as a challenge topic in large scale optimization, has already been clearly pointed out [see ref. 1, section 3.2].

Another key point is the availability of a preconditioner which can be effectively used within the truncated Newton method, that is, which does not require to store and handle any matrix. By using this kind of preconditioners, the PCG algorithm is still free from the storage and handling of the Hessian matrix, like the unpreconditioned CG algorithm. Of course, this feature is enjoyed by a preconditioner which can be built by using only information on the Hessian matrix obtained by performing products of the Hessian matrix times a vector. We remark that preconditioners of this kind can be used within discrete Newton methods [4] or when automatic differentiation [22] is used.

The first proposal of the preconditioned truncated Newton method which satisfies both these requirements was performed in ref. [23]. It is based on a two-step limited memory BFGS updating; in particular, a diagonal scaling which uses a diagonal approximation of the Hessian matrix, obtained by means of BFGS updating is defined. This preconditioning strategy has been embedded in a discrete truncated Newton method which has been numerically tested and compared with other methods for large scale unconstrained optimization [24].

Recently, an automatic preconditioning based on m-step limited memory quasi-Newton updating has been proposed [5,25]. This is a dynamic preconditioner which does not require the explicit knowledge of the Hessian matrix, too. This preconditioner has been numerically tested within a discrete Newton method and in the solution of linear systems arising in some finite element models. The results obtained in the solution of large scale unconstrained optimization problems indicate that this preconditioning strategy enables to reduce the number of CG inner iterations but it is expensive, due to the considerable computational cost of building and handling the preconditioner. This makes this particular preconditioner suited for problems for which the matrix–vector products of the Hessian matrix times a vector are expensive to compute. However, today this automatic preconditioning technique seems to be the best proposal within the class of dynamic preconditioners that we are considering.

3. Diagonal preconditioning truncated Newton methods

On the basis of the remarks contained in the previous section, we now focus on diagonal preconditioning techniques since they can be efficiently used in the context of truncated Newton methods. In fact, we propose a diagonal preconditioner which possesses the desired features which makes its use appealing within the truncated Newton methods. Before going into detail, we briefly report the main features of the generic diagonal preconditioner considering the solution of a generic linear system $As = b$, where A is a $n \times n$ nonsingular symmetric matrix and a_{ij} denotes its entries.

The diagonal preconditioning is the simplest form of preconditioning and it is very inexpensive to handle. The matrix M is chosen as a diagonal matrix

$$M = \operatorname{diag}\{d_1 \cdots d_n\} = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix},$$

where $d_i \in \mathbb{R}$, $d_i > 0$, $i = 1, \ldots, n$. This preconditioning corresponds to a *scaling* of the columns of the matrix A, aiming at obtaining a system with a column equilibrated matrix which is a better conditioned system. Analogously, a system with a row equilibrated matrix can be considered, while it is more complicated to obtain an equilibrated matrix. The advantages of having linear systems with an equilibrated matrix is well known and its importance in numerical analysis is witnessed by the number of papers devoted to this topic [see, e.g. refs. 26–28]. Moreover, there exist algorithms for equilibrating a matrix in a specific norm; some examples were proposed in refs. [28–30]. Furthermore, optimality properties of scaling in ℓ_p-norm equilibrating the rows or the columns for minimizing the condition number are considered in ref. [31, section 7.3].

Note that to apply the preconditioned CG method with this diagonal preconditioner, corresponds to apply the CG method to the system with the transformed matrix $D^{-1}AD^{-1}$, where $D = \operatorname{diag}\{d_1^{1/2}, \ldots, d_n^{1/2}\}$.

It is well known that diagonal preconditioning presents some very good features; in fact it would greatly reduce the condition number of the matrix and it does not destroy the sparsity of the matrix; moreover, it requires a minimal additional work and it is suitable to handle practical problems whose matrices entries frequently vary over many orders of magnitude.

A choice which is 'optimal' with respect to diagonal preconditioners is to use the diagonal of the matrix A as diagonal preconditioner matrix (Jacobi preconditioner); in fact this choice enables to minimize the condition number of the matrix of the transformed system with respect to all the diagonal preconditioners. In fact, the following result holds [6] [see also, e.g. refs. 14,17,31].

THEOREM 3.1 *Let A be a $n \times n$ symmetric positive definite matrix. Let $\mathcal{D}_n$ denote the set of $n \times n$ nonsingular diagonal matrices and $\Delta = \operatorname{diag}\{a_{11}^{1/2}, \ldots, a_{nn}^{1/2}\}$. Then,*

$$\kappa_2(\Delta^{-1} A \Delta^{-1}) \leq p \min_{D \in \mathcal{D}_n} \kappa_2(D^{-1} A D^{-1}),$$

where p is the maximum number of nonzero elements in any row of A.

It is important to note that even in this special case of the Jacobi preconditioner no results concerning the variation of the condition number or the eigenvalue distribution – and hence a possible improvement of the convergence rate – is available. In fact, Theorem 3.1 states only the existence of a lower bound but does not provide us with information about the variation of the condition number of the transformed system. This result, however, states that this kind of preconditioner is effective for sparse matrices. As regards the numerical efficiency of Jacobi preconditioner, it is known that it typically works well whenever the matrix A is very diagonally dominant.

Simple diagonal scaling of the variables has been already proved to be very effective in the context of L-BFGS methods [32] and within partitioned quasi-Newton methods [33] especially in tackling large scale problems. Moreover, in ref. [34] the effect of diagonal scaling with projected gradient methods for bound constrained problems has been investigated.

Now we investigate the possibility to define and efficiently use a diagonal preconditioning strategy within truncated Newton methods. In particular, we note that it corresponds to a scaling of the Hessian matrix which appears in the sequence of linear systems (2) and, as well known, to have 'good' scaling in defining algorithms is always very important, even if the algorithm we are dealing with is a scaling invariant algorithm as the Newton method. More in detail, as already observed, diagonal preconditioning corresponds to an equilibration of the data in the large linear systems to be solved at each iteration of the truncated Newton method.

A classical approach for row (column) scaling is to divide each row (column) by the norm of the row (column) vector. The most commonly used norms are the ℓ_1-norm and the ℓ_∞-norm. To scale matrices such that the norm of all rows (or columns) are equal to one is an heavy task from the computational point of view, and this should be the reason for which, as far as the author is aware, this numerical tool has not yet been fully exploited for building preconditioner in the context of truncated Newton methods, tackling large scale problems. In fact to compute the norm of each row (or column) of the Hessian matrix at each outer iteration of the method is computationally too expensive and thus impracticable, whichever norm is chosen. Also, the use of the diagonal of Hessian matrix itself as diagonal preconditioner is not practicable to derive efficiently by means of a routine which provides the product of the Hessian matrix times a vector.

On the basis of these remarks, we propose a strategy based on columns scaling of the Hessian matrix $H(x_k)$ in the solution of the systems (2) and, in particular, we focus on an equilibration strategy which uses the ℓ_1-norm.

In the sequel of this section, for the sake of simplicity, we remove the dependency on k and hence we refer to the system $Hd = -g$. Let h_{ij}, $i, j = 1, \ldots, n$, denote the entries of the Hessian matrix H and for each $j = 1, \ldots, n$ let $w_j \in \mathbb{R}^n$ defined by $w_j = He_j$, where $e_j = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is the jth unit vector, that is, for each column j, w_j denotes the column vectors of the matrix H (which are also the row-vectors for the symmetry). We consider the diagonal preconditioning generated by the matrix

$$D = \operatorname{diag}\left\{\|w_1\|_1^{1/2}, \ldots, \|w_n\|_1^{1/2}\right\} = \begin{pmatrix} \|w_1\|_1^{1/2} & & 0 \\ & \ddots & \\ 0 & & \|w_n\|_1^{1/2} \end{pmatrix}$$

Obviously, in terms of the preconditioning matrix, this corresponds to choose

$$M = \operatorname{diag}\{\|w_1\|_1, \ldots, \|w_n\|_1\},$$

and it results that the matrix HM^{-1} is a *column equilibrated matrix* in the ℓ_1-norm.

It is now important to analyze the condition number and the eigenvalues distribution of the preconditioned matrix to check if the use of such a preconditioner leads to a better conditioned system. Note that the (unsymmetric) matrix HM^{-1} and the (symmetric) matrix $M^{-1/2}HM^{-1/2}$ are similar and hence they have the same eigenvalues. Firstly, we address to the condition number of the preconditioned matrix showing that the particular column scaling proposed has a very important feature, since it is an optimal column scaling strategy. In fact, the following proposition holds.

PROPOSITION 3.2 *If H is nonsingular and $\mathcal{D}_n$ denotes the set of $n \times n$ nonsingular diagonal matrices, then*

$$\kappa_1(HM^{-1}) \leq \min_{D \in \mathcal{D}_n} \kappa_1(HD).$$

Proof The results immediately follows from Theorem 7.5 in ref. [31]. ■

Therefore the proposed column scaling is the best column scaling in terms of condition number in ℓ_1-norm. As regards the eigenvalues distribution of the preconditioned matrix, by exploiting the fact that the matrix HM^{-1} is column equilibrated in the ℓ_1-norm, it is possible to prove the following result.

PROPOSITION 3.3 *Let $\rho(HM^{-1}) = \max\{|\lambda_i|: \lambda_i$ eigenvalues of $HM^{-1}\}$ be the spectral radius of the matrix HM^{-1}. Then it results*

$$\rho(HM^{-1}) \leq 1.$$

Proof The result can be obtained by observing that for a square matrix A it results $\rho(A) \leq \|A\|_p$ for every ℓ_p-norm. In fact, we have

$$\rho(HM^{-1}) \leq \|HM^{-1}\|_1 = 1,$$

where the last equality follows from the fact that $\|HM^{-1}e_j\|_1 = 1$ for all $j = 1, \ldots, n$, i.e. all the columns of HM^{-1} have unitary ℓ_1-norm. ■

The result stated in this proposition shows how the use of the proposed preconditioner enables to have clustered eigenvalues and hence the PCG iterates are expected to converge quickly.

Now, we prove a general result concerning diagonal preconditioning.

PROPOSITION 3.4 *Let $D = \mathrm{diag}\{d_1, \ldots, d_n\}$ be a $n \times n$ diagonal matrix with $d_j > 0$, $j = 1, \ldots, n$ and $z \in \mathbb{R}^n$ such that $Hz = Dz$. Then the matrix $D^{-1/2}HD^{-1/2}$ admits an eigenvalue equal to one and the corresponding eigenvector is given by $v = D^{1/2}z$.*

Proof Since $Hz = Dz$, we have

$$D^{-1/2}HD^{-1/2}v = D^{-1/2}Hz = D^{1/2}z = v.$$

that is, v is an eigenvector of $D^{-1/2}HD^{-1/2}$ corresponding to the eigenvalue equal to 1. ■

By applying this proposition, assuming that H is a nonnegative matrix, i.e. $h_{ij} \geq 0$ for all $i, j = 1, \ldots, n$, it is possible to say more about the eigenvalues of the preconditioned matrix $M^{-1/2}HM^{-1/2}$. In fact, the following result holds.

PROPOSITION 3.5 *If H is a nonnegative matrix, the preconditioned matrix $M^{-1/2}HM^{-1/2}$ admits an eigenvalue equal to one, and the corresponding eigenvector is given by $v = M^{1/2}e$, where $e = (1, \ldots, 1)^{\mathrm{T}}$.*

Proof It is enough to apply Proposition 3.4 with $z = e$. In fact, in this case, the equality $He = Me$ holds. ■

Proposition 3.3 shows that the proposed preconditioner correspond to normalize the preconditioned matrix in such a way that its largest eigenvalue is less than or equal to one. Moreover, if the Hessian is a nonnegative matrix, Proposition 3.5 implies that the largest eigenvalue of the preconditioned matrix is equal to one. Finally, Proposition 3.4 shows that this diagonal preconditioner can be viewed as a particular case of a more general setting in which in the definition of the preconditioner M, the vector e in $He = Me$ is replaced with any vector $z \in \mathbb{R}^n$ provided that $Hz = Mz$. However, note that if a vector different from e is used in the equality $He = Me$, the optimal properties of the ℓ_1-norm scaling do not necessarily hold, even if H is a nonnegative matrix.

Our aim is now to embed this diagonal preconditioning strategy in the framework of truncated Newton methods, but it is not straightforward, since the matrix H is not available. In order to obtain information on the ℓ_1-norm of the columns of the Hessian matrix, we propose to use the only tool available in this context for providing us with information on the Hessian matrix that is the routine for the product of H times a vector. In fact, we observe that an estimate of the ℓ_1-norm of the columns of the Hessian matrix can be obtained as follows: for each jth column, $j = 1, \ldots, n$ let us consider

$$\sigma_j = \left| \sum_{i=1}^{n} h_{ij} \right|. \tag{3}$$

These quantities have the following properties:

- They are a lower estimate of the ℓ_1-norm of the jth column vector.
- Due to the symmetry of H, all the σ_j, $j = 1, \ldots, n$, can be very easily computed by means of the product of the Hessian matrix H times the vector $e = (1, 1, \ldots, 1)^{\mathrm{T}}$. In fact, by taking

the absolute value of each component of the vector resulting from this product, we obtain all the σ_j. Therefore, only one call to the routine which provides us with the product of the Hessian matrix times a vector is enough to compute all the σ_j.

Obviously, for each column j for which $h_{ij} \geq 0$ for all $i = 1, \ldots, n$, we have $\sigma_j = \|w_j\|_1$. Therefore, if H is a nonnegative matrix, then it results $\sigma_j = \|w_j\|_1$, for all $j = 1, \ldots, n$.

The quantities (3) can be used to build the diagonal preconditioner defined by the following diagonal positive definite matrix

$$\widetilde{M} = \text{diag}\{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}, \tag{4}$$

where for $j = 1, \ldots n$,

$$\tilde{\sigma}_j = \begin{cases} \sigma_j & \text{if } \sigma_j > \delta \\ 1 & \text{otherwise} \end{cases}$$

with $\delta > 0$ small.

By using this diagonal preconditioning, each column of the matrix is scaled by a quantity which represents an estimate of the ℓ_1-norm of the column vectors and no scaling is performed in correspondence of those columns for which σ_j is small.

The proposed diagonal preconditioner has the following important features:

- The explicit knowledge of the Hessian matrix is not required, but only a routine which provides the product of the Hessian matrix times a vector is needed, enabling its use within a truncated Newton method.
- It carries out a dynamical preconditioning technique, i.e. the preconditioner changes from one outer iterate to the next, thus overcoming the drawback of having the same preconditioner during the solution of each of the sequence of Newton's systems (2), whereas the Hessian matrix can change drastically.
- The solution of the auxiliary system needed to compute the preconditioned residual is very inexpensive.
- The additional cost of preconditioning is essentially due to one additional call, at each outer iteration, to the subroutine which provides the matrix–vector product of the Hessian matrix times the vector $e = (1, \ldots, 1)^{\mathrm{T}}$.

This diagonal preconditioner represents a useful tool that can be viewed as a 'dynamic scaling' since it is derived from dynamic preconditioning and scaling and we name it `DSPREC`.

We note that if the Hessian matrix H is a nonnegative matrix, the quantities σ_j in equation (3) are the ℓ_1-norm of the jth column vector and therefore the results stated in Propositions 3.2, 3.3 and 3.5 apply; in this case this diagonal preconditioner possess the important theoretical properties stated. Otherwise, as it occurs in most cases when dealing with preconditioning, in the general setting, it is impossible to derive results which apply to any matrix. Therefore, since we are interested in general purpose preconditioner, the best assessment tool is represented by an extensive numerical study.

4. Numerical experiments

If a wide numerical investigation is usually considered an essential tool for assessing the efficiency of an algorithm, it is so much important in dealing with preconditioning strategies when, as often occurs, some choices are mainly based on numerical experiments rather than on solid

theoretical results. Keeping in mind the central role of the numerical experiments in evaluating the effectiveness of different preconditioning strategies, we have been trying to perform a very extensive numerical testing. To this aim we embedded the proposed preconditioning strategy in a linesearch-based truncated Newton method which uses a preconditioned CG algorithm for computing the search direction, terminating the inner PCG iterations whenever the residual-based criterion is satisfied or a negative curvature direction is encountered [10]. This criterion is preferred here with respect to the quadratic model-based criterion proposed in ref. [35] since it allows to easily control the rate of convergence of the truncated Newton method. Further details of our implementation can be found in ref. [36]; here we mention that the termination criterion of the outer iterations is the standard test $\|g(x_k)\|_2 < 10^{-5}\max(1, \|x_k\|_2)$, and that we set $\delta = 10^{-6}$ in the definition of the preconditioner (4). All the tests have been performed on a Pentium III 700 with 512 Mb RAM, and all the codes used are double precision FORTRAN codes. All the results are reported in terms of number of outer iterations (it) – which is the same as the number of gradient evaluation (ng) – number of function evaluations (nf), total number of inner iterations (CG-it) and CPU time (time).

As regards a different preconditioning strategy available in literature needed for making a comparison of the results, we have been seeking among those preconditioners which can be efficiently used within the truncated Newton framework and the state-of-the-art seems to be represented by the automatic preconditioner proposed in refs. [5,25], implemented in the available software PREQN that we already mentioned in the previous section. It has the form of a limited memory quasi-Newton matrix and it is generated using information from the CG iterations without requiring the explicit knowledge of the Hessian matrix. It is a dynamic preconditioner which exploits only the matrix–vector product of the Hessian matrix times a vector as information on the Hessian matrix and therefore a coherent comparison of the results can be performed by embeddeding the preconditioner PREQN within our implementation of the linesearch-based truncated Newton method.

Since the proposed preconditioning strategy is also motivated by the possibility of more efficiently solving real problems, we start our numerical investigation by considering real unconstrained problems taken from the MINPACK-2 collection [8] for which a routine which computes the product of the Hessian matrix times a vector is available. The aim of this preliminary test is to check the reliability of the diagonal preconditioning strategy DSPREC as well as to compare the results with those obtained by means of unpreconditioned CG and preconditioner PREQN. We report these results in table 1. As it can be observed from this table, on the problem GL2 (which is the most difficult in terms of computational burden needed to solve it) the use of the proposed diagonal preconditioning leads to a significative reduction of the total number of the inner iterations with respect to the results obtained by the unpreconditioned CG and by PREQN; moreover, a significant saving of the computing time

Table 1. Results on MINPACK-2 problems: 2-dimensional Ginzburg–Landau model for superconductivity with $n_v = 8$ (GL2); minimal surfaces (MSA); optimal design with composite materials with $\lambda = 0.008$ (ODC).

		Unpreconditioned CG				PREQN				DSPREC			
Problem	n	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time
GL2	400	28	19	6,282	30.13	34	146	6,031	33.54	17	60	4,017	18.59
	900	37	149	11,755	126.91	49	234	16,927	211.47	16	50	9,275	97.04
MSA	400	4	4	52	0.10	4	4	35	0.10	4	4	75	0.12
	900	5	5	116	0.32	5	5	79	0.35	5	5	223	0.58
ODC	400	11	15	268	0.47	11	16	168	0.49	12	15	269	0.48
	900	14	18	297	0.86	14	20	233	1.06	15	22	595	1.62

is also noticed. This situation is evidenced in both the instances, that is the beneficial effect of diagonal preconditioning is not reduced as a consequence of an increase of the dimension of the problem; moreover, note that a reduction of number of the iterations is obtained, too. As regards the other two problems, they are solved very quickly by all the three algorithms even if, on the instances of 900 variables, the diagonal preconditioning does not seems to be effective since an increase of the number of CG iterations is observed; however, note that this does not lead to a significative increase of CPU time needed to solve the problems.

Now, after this preliminary test, we turn to a more general numerical investigation. In particular, we are interested in evaluating the effectiveness of the proposed diagonal preconditioning strategy as 'general purpose' rather than an 'ad hoc' strategy for a specific problem. To this aim, we now consider a very large set of test problems; in fact we use all the large scale ($n \geq 1000$) unconstrained problems available in the CUTE collection [9]. Moreover, for those problems with variable dimension, we consider two instances, usually $n = 1000$ and $n = 10{,}000$ or different if a problem has assigned dimensions. These leads to consider 117 test problems. The computation of the matrix–vector products of the Hessian matrix times a vector is performed directly by using the routine uprod available from the CUTE environment.

The complete results obtained by our implementation of the linesearch-based truncated Newton using the unpreconditioned CG, the quasi-Newton preconditioner PREQN and the diagonal preconditioner DSPREC are reported in ref. [36]. For the sake of brevity, here we report only some summary of the results. In particular, in tables 2–4, a summary of the results considering all those problems coherently solved by the three algorithms and where they converges towards to the same point is reported. As it can be observed, the use of the diagonal preconditioner DSPREC enables a considerable computational saving on most problems with respect to both the unpreconditioned method and the one using PREQN, both in terms of number of inner iterations and in terms of CPU time and on some test problems the gain is impressive (see, e.g. problems DIXMAANE, DIXMAANF, DIXMAANG, DIXMAANH, DIXMAANI, DIXMAANJ, DIXMAANK, DIXMAANL, GENHUMPS, NONDQUAR, POWER ($n = 10{,}000$), SPARSINE, TESTQUAD, TQUARTIC ($n = 1000$), TRIDIA). We have been investigating the distribution of the eigenvalues for these problems and, for the sake of shortness, we report here only the plot of the distributions which seem to be typical and significative (in the plot we reduce the dimension in order to be able to compute and plot the spectrum of the matrices). Figure 1 reports a typical distribution of the eigenvalues of the Hessian matrix and the Hessian matrix preconditioned via DSPREC for the DIXMAANα problems. The effect of clustering due to preconditioning is clearly evidenced. Figure 2 reports the spectrum of the Hessian and the preconditioned Hessian for NONDQUAR problem ($n = 100$): the Hessian matrix presents 99 eigenvalues in the range [0, 48] and one eigenvalue over 10^3, while all the eigenvalues of the preconditioned Hessian matrix belong to [0, 1]. Finally, we report the plot of the eigenvalue distribution for the problem TESTQUAD on which the diagonal preconditioner DSPREC is really very effective. The plot reported in figure 3 gives a clear justification of this. The eigenvalues distributions now reported are only examples, but they represent the typical situation which occurs whenever the preconditioner works well; in particular, the capability of clustering the eigenvalues is clearly showed. It is at the basis of the preconditioning strategy and explains the great improvements obtained on many problems compared with the unpreconditioned method.

Moreover, it is very important to note that on those problems where DSPREC leads to a reduction of the number of the inner iterations with respect to the unpreconditioned case, a saving of CPU time is always observed, too. This means that DSPREC possesses the fundamental feature of a good preconditioner, that is, the cost of building and applying it is very low, thus succeeding not only to offset the additional computational effort, but also allowing to obtain

Table 2. Results for unpreconditioned CG, PREQN and DSPREC – Part A.

		Unpreconditioned CG				PREQN				DSPREC			
	n	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time
ARWHEAD	1,000	6	6	6	0.46	6	6	6	0.08	7	7	7	0.09
ARWHEAD	10,000	6	6	6	0.97	6	6	6	1.11	7	7	7	1.42
BDQRTIC	1,000	13	13	59	0.26	14	14	59	0.37	10	10	13	0.16
BDQRTIC	10,000	14	32	64	5.31	14	32	53	5.42	10	10	12	2.35
BRYBND	1,000	11	11	73	0.39	11	11	71	0.51	11	11	69	0.45
BRYBND	10,000	17	24	168	11.33	16	20	141	11.33	12	12	78	6.34
COSINE	1,000	7	8	9	0.07	7	8	9	0.05	9	13	12	0.07
COSINE	10,000	7	8	7	0.87	7	8	9	0.89	9	13	12	1.36
CRAGGLVY	1,000	15	15	102	0.32	16	16	92	0.45	20	21	149	0.48
CRAGGLVY	10,000	17	17	133	7.94	16	16	75	6.25	20	21	168	10.29
CURLY10	1,000	17	31	8,008	6.74	14	28	4,806	10.84	18	25	8,295	8.51
CURLY20	1,000	17	31	6,783	20.38	16	26	5,798	24.86	18	29	6,903	22.06
CURLY30	1,000	19	36	7,126	35.25	18	37	5,955	34.95	18	37	6,317	32.60
DIXMAANA	1,500	7	7	8	0.12	8	8	11	0.14	8	8	8	0.15
DIXMAANA	3,000	8	8	9	0.34	8	8	10	0.31	8	8	8	0.33
DIXMAANB	1,500	8	8	8	0.14	8	8	8	0.14	8	8	8	0.17
DIXMAANB	3,000	8	8	8	0.34	8	8	8	0.30	8	8	8	0.36
DIXMAANC	1,500	9	9	9	0.15	9	9	9	0.16	9	9	9	0.17
DIXMAANC	3,000	9	9	9	0.38	9	9	9	0.34	9	9	9	0.40
DIXMAAND	1,500	11	11	13	0.20	11	11	12	0.20	10	10	10	0.20
DIXMAAND	3,000	11	11	13	0.48	11	11	12	0.48	10	10	10	0.52
DIXMAANE	1,500	10	10	188	0.55	11	11	285	1.48	9	9	9	0.16
DIXMAANE	3,000	11	11	427	3.04	11	11	345	3.84	9	9	9	0.37
DIXMAANF	1,500	15	19	406	1.40	18	29	482	2.64	14	14	24	0.30
DIXMAANF	3,000	15	20	600	4.69	17	20	626	7.18	14	14	23	0.70
DIXMAANG	1,500	15	20	450	1.57	16	23	439	2.36	14	14	23	0.30
DIXMAANG	3,000	16	21	370	3.15	18	34	464	5.59	16	16	34	0.87
DIXMAANH	1,500	19	27	423	1.57	18	21	426	2.32	16	21	34	0.38

Table 3. Results for unpreconditioned CG, PREQN and DSPREC – Part B.

		Unpreconditioned CG				PREQN				DSPREC			
	n	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time
DIXMAANH	3,000	21	33	637	5.14	25	50	886	9.91	16	16	32	0.89
DIXMAANI	1,500	11	11	3,255	7.73	11	11	2,988	13.68	9	9	9	0.16
DIXMAANI	3,000	11	11	6,218	37.81	11	11	5,799	57.38	9	9	9	0.39
DIXMAANJ	1,500	26	58	4,069	12.60	37	132	7,028	34.55	16	16	25	0.33
DIXMAANJ	3,000	35	114	8,621	58.41	30	83	3,787	40.38	16	16	24	0.82
DIXMAANK	1,500	20	43	1,865	5.94	39	121	3,986	19.96	16	16	24	0.34
DIXMAANK	3,000	48	185	8,359	57.70	38	111	9,679	101.94	16	16	23	0.80
DIXMAANL	1,500	21	42	3,721	11.58	26	50	2,988	13.38	17	17	26	0.40
DIXMAANL	3,000	42	152	5,505	38.57	62	234	8,883	94.93	17	17	26	0.86
DQDRTIC	1,000	7	7	14	0.06	6	6	9	0.07	2	2	2	0.01
DQDRTIC	10,000	8	8	16	1.39	7	7	12	1.24	2	2	2	0.23
DQRTIC	1,000	23	23	23	0.07	23	23	23	0.08	23	23	23	0.08
DQRTIC	10,000	27	27	27	1.84	27	27	27	1.86	27	27	27	2.19
EDENSCH	1,000	14	15	29	0.21	15	17	33	0.33	13	13	19	0.20
EDENSCH	10,000	17	17	40	4.46	15	16	31	3.79	13	13	19	3.24
EIGENALS	930	39	57	811	21.81	32	38	688	18.99	38	52	138	8.75
ENGVAL1	1,000	10	10	25	0.11	10	10	19	0.18	9	9	13	0.10
ENGVAL1	10,000	10	10	20	2.13	10	10	16	2.03	9	9	13	1.95
FLETCBV2	1,000	1	1	1	0.01	1	1	1	0.01	1	1	1	0.00
FLETCBV2	10,000	1	1	1	0.08	1	1	1	0.08	1	1	1	0.07
FLETCHCR	1,000	1,484	1,699	24,522	32.15	1,477	1,683	18,937	60.57	1,475	1,681	25,207	36.67
FMINSURF	1,024	39	243	6,854	10.12	34	223	6,162	18.58	26	167	5,547	9.03
FMINSURF	5,625	62	601	33,799	641.33	37	285	21,253	514.60	41	362	27,395	538.76
FREUROTH	1,000	12	17	30	0.19	12	17	25	0.26	13	18	24	0.26
FREUROTH	10,000	11	16	23	2.76	11	16	18	2.68	13	18	24	3.75
GENHUMPS	1,000	2,154	2,703	5,657	24.67	1,162	3,204	3,686	18.58	615	1,696	1,873	8.38
GENROSE	1,000	550	1,713	9,388	12.69	638	2,407	8,289	22.48	560	1,589	7,528	12.52
LIARWHD	1,000	16	16	23	0.18	15	15	24	0.20	12	12	20	0.13
LIARWHD	10,000	17	19	23	2.91	14	14	22	2.47	13	13	17	2.44

Table 4. Results for unpreconditioned CG, PREQN and DSPREC – Part C.

		Unpreconditioned CG				PREQN				DSPREC			
	n	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time	it/ng	nf	CG-it	Time
MOREBV	1,000	2	2	185	0.12	2	2	185	0.15	2	2	185	0.15
MOREBV	10,000	2	2	1,200	32.33	2	2	1,200	34.83	2	2	1,200	33.32
NCB20B	1,000	18	55	975	18.82	21	69	1,538	30.07	18	70	3,263	59.19
NONDQUAR	1,000	61	170	10,793	5.38	68	203	12,669	16.94	38	78	3,814	2.17
NONDQUAR	10,000	46	105	3,222	74.90	60	154	3,743	134.79	25	34	801	20.77
PENALTY1	10,000	47	49	56	5.75	47	49	56	5.70	52	54	494	18.00
POWELLSG	1,000	20	20	66	0.10	20	20	52	0.13	19	19	68	0.11
POWELLSG	10,000	21	21	74	2.77	21	21	52	2.72	20	20	75	3.01
POWER	1,000	32	32	937	0.57	32	32	296	0.51	31	37	406	0.33
POWER	10,000	38	38	2,608	49.88	38	38	1,002	32.93	36	64	121	4.87
SCHMVETT	1,000	7	7	36	0.17	7	7	29	0.19	7	7	29	0.17
SCHMVETT	10,000	8	8	46	3.28	7	7	26	2.62	8	9	32	2.98
SPARSINE	1,000	17	21	3,729	9.10	36	205	3,234	12.70	5	5	5	0.07
SPARSQUR	1,000	20	20	36	0.28	20	20	27	0.29	20	20	20	0.27
SPARSQUR	10,000	23	23	38	9.08	23	23	30	8.59	23	23	23	9.47
SPMSRTLS	1,000	12	16	143	0.59	12	15	133	0.75	18	58	727	2.59
SPMSRTLS	10,000	18	41	303	17.53	15	32	319	21.78	51	276	7,436	360.34
SROSENBR	1,000	8	8	9	0.04	8	8	10	0.05	9	9	11	0.05
SROSENBR	10,000	8	8	9	0.79	8	8	10	0.85	9	9	11	1.02
TESTQUAD	1,000	14	14	1,188	0.62	14	14	1,454	1.66	2	2	2	0.01
TOINTGSS	1,000	5	5	9	0.05	5	5	9	0.06	3	3	3	0.04
TOINTGSS	10,000	4	4	4	0.47	4	4	4	0.59	3	3	3	0.35
TQUARTIC	1,000	464	469	921	3.23	16	35	26	0.13	10	11	14	0.07
TQUARTIC	10,000	8	12	9	1.00	9	13	11	1.18	7	11	9	1.03
TRIDIA	1,000	12	12	674	0.35	12	12	488	0.60	9	9	47	0.10
TRIDIA	10,000	13	13	1,910	37.93	13	13	1,479	48.06	9	9	47	1.70
VARDIM	1,000	19	177	18	0.11	19	177	18	0.11	19	151	179	0.20
VAREIGVL	1,000	16	18	1,501	4.08	32	98	393	2.11	27	73	4,849	13.37
WOODS	1,000	296	305	1,159	2.38	47	79	136	0.45	82	100	298	0.72
WOODS	10,000	283	296	1,108	53.19	46	75	134	9.41	83	100	302	17.72

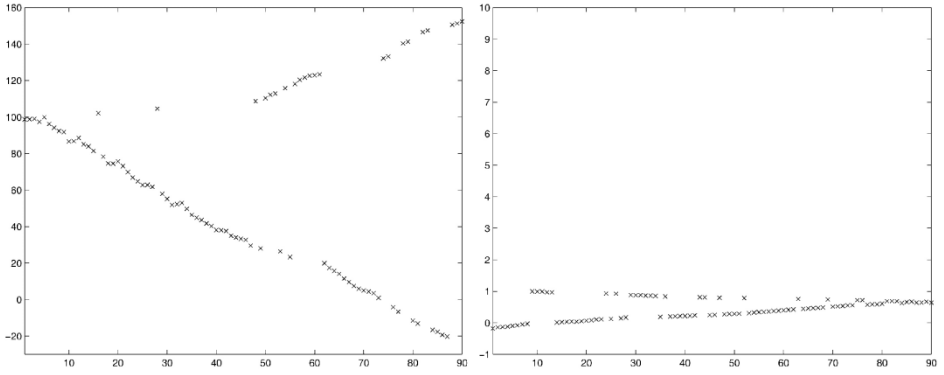

Figure 1. Distribution of the eigenvalues of the Hessian matrix H (on the left) and the preconditioned matrix $\widetilde{M}^{-1/2}H\widetilde{M}^{-1/2}$ (on the right) for the problem DIXMAANK $n = 90$.

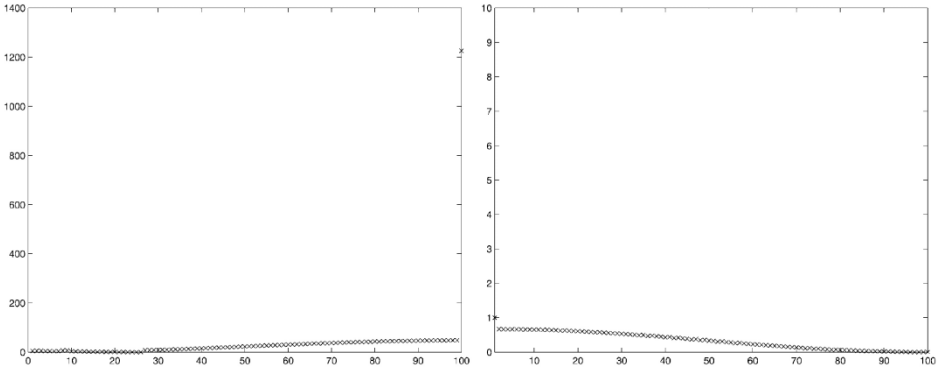

Figure 2. Distribution of the eigenvalues of the Hessian matrix H (on the left) and the preconditioned matrix $\widetilde{M}^{-1/2}H\widetilde{M}^{-1/2}$ (on the right) for the problem NONDQUAR $n = 100$.

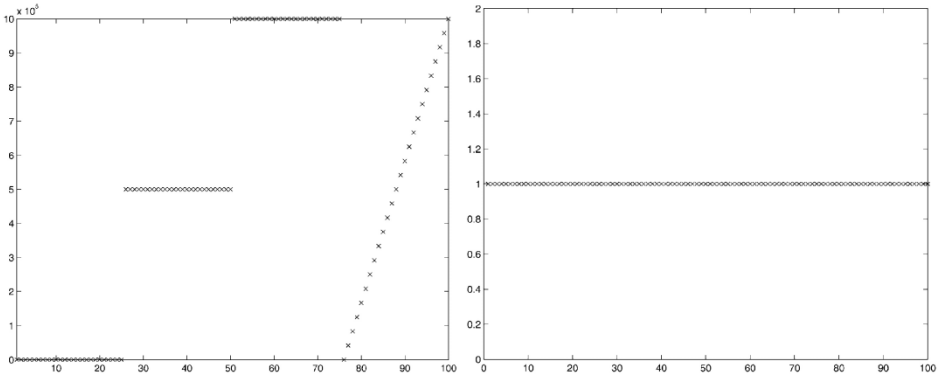

Figure 3. Distribution of the eigenvalues of the Hessian matrix H (on the left) and the preconditioned matrix $\widetilde{M}^{-1/2}H\widetilde{M}^{-1/2}$ (on the right) for the problem TESTQUAD $n = 100$.

an overall computational saving. On the contrary, as regards PREQN, in many cases, even if it enables a reduction of the inner iterations, an increase of the time needed to solve the problem is observed with respect to the unpreconditioned case, due to the cost of preconditioning.

Furthermore, by observing tables 2–4 it is worthwhile noticing that there are problems (see, e.g. CURLY30, POWER, FLETCHCR) on which both the preconditioning strategies allows to obtain a reduction of the number of inner iterations with respect to the unpreconditioned method, and moreover, PREQN behaves better than DSPREC in terms of inner iterations. However, if we observe the CPU time needed to solve these problems, we discover that PREQN is more expensive, pointing out a general consideration on the fact that a preconditioning strategy more elaborate with respect to a diagonal preconditioning can lead to a great reduction of the number of inner iterations but with an heavy computational effort (note that the differences obtained on these problems in terms of CPU time are only due to the different preconditioners, since the number of iterations is nearly the same).

Finally there are four problems (NCB20B, SPMSRTLS, VARDIM, VAREIGVL) on which DSPREC has a poor behavior in terms of inner iterations and CPU time, in comparison with the other two methods. We have been studying the distribution of the eigenvalues for those problems where the preconditioner does not work well. The plot for two typical significative situations are now reported: the spectrum of the Hessian and the Hessian matrix preconditioned via DSPREC for the problem NCB20B and for the problem SPMSRTLS are plotted in figures 4 and 5, respectively. Figure 4 points out that the poor behavior is mainly due to the fact the Hessian matrix is nearly singular. As regards figure 5, it can be observed that the eigenvalues remain similarly distributed after preconditioning; in this case, the deterioration of the performance is justified by a significative increase of the condition number of the preconditioned Hessian matrix with respect to the unpreconditioned one.

The latter case considered represents an example for which the Hessian matrix is such as to give rise to the definition of a very poor preconditioner. This relies, in general, on the fact that the diagonal elements σ_j in (3) might be a poor approximation of the ℓ_1-norm of the jth column vector of the Hessian matrix. When this occurs, the equilibration strategy based on columns scaling of DSPREC may be not successful in the sense that the Hessian matrix could remain badly scaled or even the conditioning could be worsened after preconditioning. This seems to be one of the main reasons of the poor behavior of DSPREC observed is some cases.

Another interesting analysis of the results can be carried on by considering the cumulative results, that is the total number of iterations, function evaluations, inner iterations and CPU time needed to solve all the problems considered in the tables 2–4. These cumulative results are

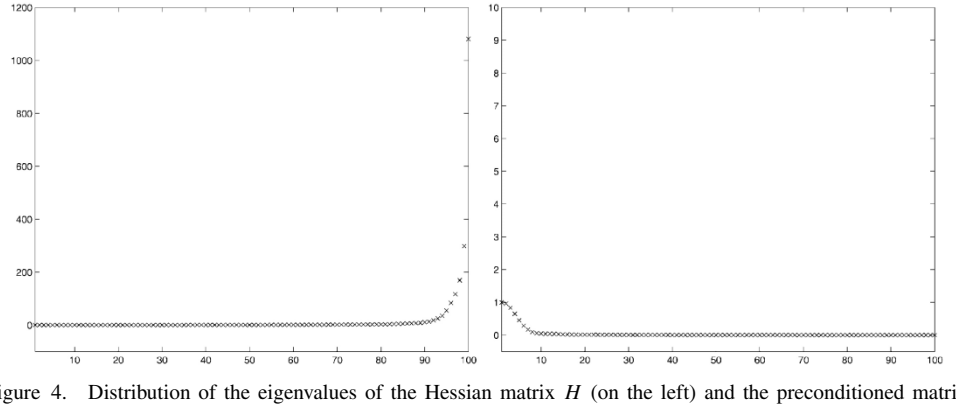

Figure 4. Distribution of the eigenvalues of the Hessian matrix H (on the left) and the preconditioned matrix $\widetilde{M}^{-1/2} H \widetilde{M}^{-1/2}$ (on the right) for the problem NCB20B $n = 100$.

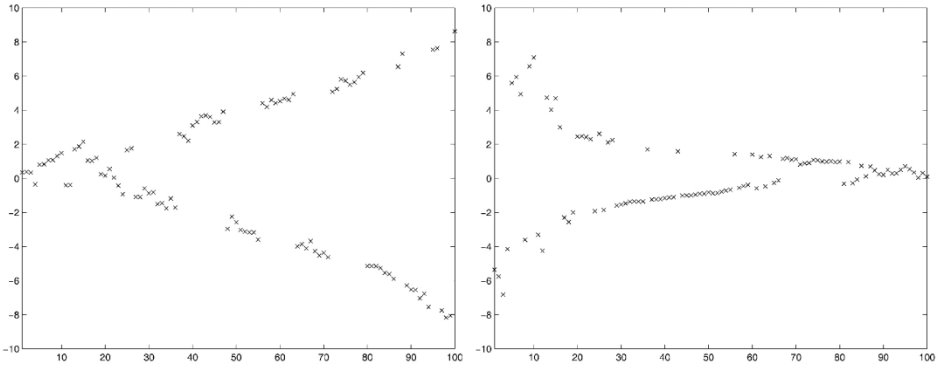

Figure 5. Distribution of the eigenvalues of the Hessian matrix H (on the left) and the preconditioned matrix $\widetilde{M}^{-1/2}H\widetilde{M}^{-1/2}$ (on the right) for the problem SPMSRTLS $n = 100$.

Table 5. Cumulative results for unpreconditioned CG, PREQN and diagonal preconditioned CG.

	Unpreconditioned CG	PREQN	DSPREC
it/ng	6,617	4,832	4,016
nf	10,260	10,712	7491
CG-it	182,027	154,257	114,833
Time	1437.95	1550.73	1279.48

reported in table 5. They confirm the effectiveness of DSPREC; in fact, the method which uses the diagonal preconditioning strategy performs the best in terms of all the criteria considered. In order to complete the numerical comparison among the three algorithms, in table 6 we report the number of times each algorithm performs the best in term of number of iterations, function evaluations, inner iterations and CPU time. This table confirms that, in most cases, the diagonal preconditioning strategy proposed in this paper produces the best results with respect to the unpreconditioned method and the automatic preconditioning PREQN.

To conclude the numerical investigations, it is interesting to compare the behavior of DSPREC with other two diagonal preconditioners which can be considered ideal, in the sense they satisfy the strong theoretical properties reported in the previous section: the *Jacobi preconditioner* and the *exact diagonal column scaling* in the ℓ_1-norm. Actually, for large scale problems, they are impracticable since to build them the actual elements of the Hessian matrix must be known. However, this comparison is very interesting to know how close DSPREC is to an 'ideal' preconditioner. Note that a huge computational effort is usually required to

Table 6. Number of times each algorithm performs the best.

	Unpreconditioned CG	PREQN	DSPREC
it/ng	24	29	55
nf	25	30	54
CG-it	18	29	54
Time	16	13	33

Table 7. Results for Jacobi preconditioner, exact diagonal column scaling and DSPREC – Part A.

		Jacobi preconditioner			Exact diagonal column scaling			DSPREC		
	n	it/ng	nf	CG-it	it/ng	nf	CG-it	it/ng	nf	CG-it
ARWHEAD	1000	7	7	7	7	7	7	7	7	7
BDQRTIC	1000	10	10	18	10	10	13	10	10	13
BRYBND	1000	10	10	32	11	11	32	11	11	69
COSINE	1000	7	8	9	7	8	9	9	13	12
CURLY10	1000	17	26	7455	17	22	7821	18	25	8295
CURLY20	1000	18	35	6678	19	32	8467	18	29	6903
CURLY30	1000	18	33	6663	18	33	5830	18	37	6317
DIXMAANA	1500	8	8	8	8	8	8	8	8	8
DIXMAANA	3000	8	8	8	8	8	8	8	8	8
DIXMAANB	1500	8	8	8	8	8	8	8	8	8
DIXMAANB	3000	8	8	8	8	8	8	8	8	8
DIXMAANC	1500	10	10	12	9	9	9	9	9	9
DIXMAANC	3000	10	10	12	9	9	9	9	9	9
DIXMAAND	1500	11	11	13	10	10	10	10	10	10
DIXMAAND	3000	11	11	13	10	10	10	10	10	10
DIXMAANE	1500	8	8	8	9	9	9	9	9	9
DIXMAANE	3000	8	8	8	9	9	9	9	9	9
DIXMAANF	1500	9	9	9	14	14	24	14	14	24
DIXMAANF	3000	9	9	9	14	14	23	14	14	23
DIXMAANG	1500	10	10	12	14	14	23	14	14	23
DIXMAANG	3000	10	10	10	17	18	39	16	16	34
DIXMAANH	1500	11	11	14	16	21	34	16	21	34
DIXMAANH	3000	11	11	14	16	16	32	16	16	32
DIXMAANI	1500	8	8	8	9	9	9	9	9	9
DIXMAANI	3000	8	8	8	9	9	9	9	9	9
DIXMAANJ	1500	10	10	11	16	16	25	16	16	25
DIXMAANJ	3000	10	10	10	16	16	24	16	16	24
DIXMAANK	1500	10	10	10	16	16	24	16	16	24
DIXMAANK	3000	10	10	10	16	16	23	16	16	23
DIXMAANL	1500	11	11	11	17	17	26	17	17	26
DIXMAANL	3000	11	11	11	17	17	26	17	17	26
DQDRTIC	1000	2	2	2	2	2	2	2	2	2
DQRTIC	1000	23	23	23	23	23	23	23	23	23
EDENSCH	1000	13	13	19	13	13	21	13	13	19

construct the preconditioner which carries out the exact diagonal column scaling, and therefore in this comparison we concentrate on the number of CG iterations needed for solving each problem without considering the CPU time. In tables 7 and 8 these results are reported for all the problems where the three algorithms converge to the same point, excluding those ones with 10,000 variables for which the computation of the exact ℓ_1-norm of the columns of the Hessian matrix is impracticable. As it can be observed from these tables, in many cases DSPREC behaves exactly like the exact column scaling or it is very close to it, and this is at the basis of the good behavior of DSPREC on those problems. On the overall, these results reveal that the use of the diagonal of the Hessian as preconditioner would be very effective in terms of number of CG inner iterations, whenever this diagonal were available at a low computational cost; as regards DSPREC, these results point out how it closely resembles the exact diagonal column scaling, leading to a great efficiency due to the fact that DSPREC can be built very inexpensively.

Table 8. Results for Jacobi preconditioner, exact diagonal column scaling and DSPREC – Part B.

		Jacobi preconditioner			Exact diagonal column scaling			DSPREC		
	n	it/ng	nf	CG-it	it/ng	nf	CG-it	it/ng	nf	CG-it
EIGENALS	930	31	45	143	37	52	129	38	52	138
ENGVAL1	1,000	9	9	13	9	9	13	9	9	13
FLETCBV2	1,000	1	1	1	1	1	1	1	1	1
FLETCHCR	1,000	1,477	1,683	23,808	1,479	1,689	23,795	1,475	1,681	25,207
FMINSURF	1,024	1,412	2,915	2,520	41	378	2,557	26	167	5,547
FREUROTH	1,000	10	15	18	11	16	17	13	18	24
GENHUMPS	1,000	306	716	585	1,142	1,429	2,306	615	1,696	1,873
GENROSE	1,000	711	2,350	6,262	669	1,853	6,035	560	1,589	7,528
LIARWHD	1,000	12	12	21	11	11	18	12	12	20
MOREBV	1,000	2	2	188	2	2	186	2	2	185
NCB20B	1,000	23	90	3,165	18	66	3,338	18	70	3,263
NONDIA	1,000	776	807	1,542	36	58	50	36	58	50
NONDQUAR	1,000	41	91	2,903	33	66	2,877	38	78	3,814
PENALTY1	1,000	42	44	70	50	57	323	49	55	324
POWELLSG	1,000	19	19	63	19	19	60	19	19	68
POWER	1,000	31	37	36	31	37	225	31	37	406
QUARTC	1,000	23	23	23	23	23	23	23	23	23
SCHMVETT	1,000	7	7	26	8	8	42	7	7	29
SINQUAD	1,000	111	168	298	393	427	1,175	89	157	221
SPARSINE	1,000	15	18	2,433	12	12	1,465	5	5	5
SPARSQUR	1,000	20	20	50	20	20	20	20	20	20
SPMSRTLS	1,000	17	35	132	11	12	119	18	58	727
SROSENBR	1,000	9	10	11	9	9	11	9	9	11
TESTQUAD	1,000	2	2	2	2	2	2	2	2	2
TOINTGSS	1,000	7	7	10	7	7	10	3	3	3
TQUARTIC	1,000	15	20	21	15	23	21	10	11	14
TRIDIA	1,000	11	11	46	11	11	46	9	9	47
VARDIM	1,000	28	101	1,340	19	151	184	19	151	179
VAREIGVL	1,000	15	17	628	15	15	736	27	73	4,849
WOODS	1,000	48	58	155	83	104	302	82	100	298

5. Concluding remarks

In this paper, the problem of defining preconditioning strategies for truncated Newton methods has been considered. In particular, dynamical preconditioners which use only information on the Hessian matrix obtained by the product of the Hessian matrix times a vector are investigated. Within this framework, a new preconditioning strategy is proposed based on a dynamic scaling of the Hessian matrix, aiming at obtaining a column equilibrated matrix in ℓ_1-norm.

The diagonal preconditioning strategy proposed has been embedded within a linesearch-based truncated Newton method and an extensive numerical investigation has been performed on a large set of large scale test problems. The obtained results have been compared with those obtained by means of the same method which does not use any preconditioner and by a preconditioned method which uses the automatic preconditioner based on limited quasi-Newton updating (PREQN) proposed in ref. [5]. This comparison evidenced that the truncated Newton method which uses this new diagonal preconditioning strategy is very effective in the solution of most problems considered, performing better than the unpreconditioned method

and the one which uses the automatic quasi-Newton preconditioner PREQN. It is also evident that both the preconditioners are not always beneficial; hence, once more, the experiments performed in this work pointed out the difficulty to define a 'general purpose' preconditioner. Further numerical experiences showed that the behavior of the new diagonal preconditioner is very closely related to the ideal exact diagonal column scaling.

Even if no final conclusion can be drawn, on the whole, the new preconditioning strategy proposed in this paper seems to be very efficient and we believe that it could be successfully used for defining efficient truncated Newton methods for the solution of large scale unconstrained problems. Moreover, some points are still worthwhile investigating: to devise adaptive rules to decide when the preconditioner should be used based on information gained from the CG iterations, getting inspiration from ref. [37]; to exploit the information obtained from the sequence of the matrix–vector products involving the Hessian computed at each CG iteration to improve the preconditioner, like, e.g. in ref. [23]; to combine the use of this diagonal preconditioner with other preconditioners, or to use it to initialize other preconditioners. Furthermore, the diagonal preconditioner could be combined with a suited transformation so that the first matrix–vector product at each first inner iteration provides the product He, thus avoiding the extra matrix–vector product needed, at each outer iteration, to construct the diagonal preconditioner.

Finally, we think that the definition of an effective preconditioning strategy is strictly connected with the choice of an efficient truncation criterion for the inner iterations which is still a challenging topic for the research in the truncated Newton methods.

Acknowledgements

The author wish to thank Stefano Lucidi for his many constructive comments and suggestions. Thanks also to Giovanni Fasano for helpful discussions and for carefully reading a preliminary version of this paper. The author is indebted to the anonymous referees for their helpful suggestions which led to improve very much the paper. This work was supported by Progetto FIRB 'Ottimizzazione non lineare su larga scala'.

References

[1] Nocedal, J., 1997, Large scale unconstrained optimization. In: A. Watson and I. Duff (Eds) *The State of the Art in Numerical Analysis* (Oxford: Oxford University Press), pp. 311–338.
[2] Roma, M., 2001, Large scale unconstrained optimization. In: C. Floudas and P. Pardalos (Eds) *Encyclopedia of Optimization*, Vol. III (Kluwer Academic Publishers), pp. 143–150.
[3] Nash, S.G., 2000, A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, **124**, 45–59.
[4] O'Leary, D.P., 1982, A discrete Newton algorithm for minimizing a function of many variables. *Mathematical Programming*, **23**, 20–33.
[5] Morales, J.L. and Nocedal, J., 2000, Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, **10**, 1079–1096.
[6] Van der Sluis, A., 1969, Condition number and equilibration of matrices. *Numerische Mathematik*, **14**, 14–23.
[7] Bauer, F.L., 1963, Optimally scaled matrices. *Numerische Mathematik*, **5**, 73–87.
[8] Averick, B.M., Carter, R.G., Moré, J.J. and Xue, G., 1992, The MINPACK-2 test problems collection. Preprint MCS-P153-0692, Argonne National Laboratoty, Argonne, IL.
[9] Bongartz, I., Conn, A.R., Gould, N.I.M. and Toint, Ph.L., 1995, CUTE: Constrained and unconstrained testing environment. *ACM Transaction on Mathematical Software*, **21**, 123–160.
[10] Dembo, R.S. and Steihaug, T., 1983, Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, **26**, 190–212.
[11] Dembo, R.S., Eisenstat, S.C. and Steihaug, T., 1982, Inexact Newton methods. *SIAM Journal on Numerical Analysis*, **19**, 400–408.
[12] Golub, G.H. and Van Loan, C.F., 1996, *Matrix Computations* (3rd edn) (Baltimore: The John Hopkins Press).
[13] Barret, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and Van der Vorst, H., 1994, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* (Philadelphia, PA: SIAM).

[14] Greenbaum, A., 1997, *Iterative Methods for Solving Linear Systems* (Philadelphia, PA: SIAM).
[15] Van Der Vorst, H. and Dekker, K., 1988, Conjugate gradient type methods and preconditioning. *Journal of Computational and Applied Mathematics*, **24**, 73–87.
[16] Axelsson, O., 1994, *Iterative Solution Methods* (Cambridge: Cambridge University Press).
[17] Meurant, G., 1999, *Computer Solution of Large Linear Systems* (Elsevier, Amsterdam: North Holland).
[18] Lin, C.-J. and Moré, J., 1999, Incomplete Cholesky factorization with limited memory. *SIAM Journal on Scientific Computing*, **21**, 24–45.
[19] Schlick, T., 1993, Modified Cholesky factorization for sparse preconditioners. *SIAM Journal on Scientific Computing*, **14**, 424–445.
[20] Bouaricha, A., Moré, J. and Wu, Z., 1998, Preconditioning Newton's method. Technical Report CRPC-TR98762, Center for Research on Parallel Computing, Rice University, Houston, TX.
[21] Daydé, M.J., L'Excellent, J.-Y. and Gould, N.I.M., 1997, Element-by-element preconditioners for large-scale partially separable optimization problems. *SIAM Journal on Scientific Computing*, **18**, 1767–1787.
[22] Griewank, A., 1989, On automatic differentiation. In: M. Iri and K. Tanake (Eds) *Mathematical Programming: Recent Developments and Applications* (Kluwer Academic Publishers), pp. 83–108.
[23] Nash, S.G., 1985, Preconditioning of truncated-Newton methods. *SIAM Journal on Scientific and Statistical Computing*, **6**, 599–616.
[24] Nash, S.G. and Nocedal, J., 1991, A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM Journal on Optimization*, **1**, 358–372.
[25] Morales, J.L. and Nocedal, J., 2001, Algorithm PREQN: Fortran 77 subroutine for preconditioning the conjugate gradient method. *ACM Transaction on Mathematical Software*, **27**, 83–91.
[26] Duff, I.S., Erisman, A.M. and Reid, J.K., 1986, *Direct Methods for Sparse Matrices* (London: Oxford University Press).
[27] Schneider, M.H. and Zenios, S., 1990, A comparative study of algorithms for matrix balancing. *Operations Research*, **38**, 439–455.
[28] Ruiz, D., 2001, A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical Report RAL-TR-2001-034, Computational Sciences and Engineering Department, Rutherford Appleton Laboratory, Oxon, UK. (Submitted to *Linear Algebra and Applications*.)
[29] Bunch, J.R., 1971, Equilibration of symmetric matrices in the max-norm. *Journal of ACM*, **18**, 566–572.
[30] Parlett, B.N. and Landis, T.L., 1982, Methods for scaling to double stochastic form. *Linear Algebra and Applications*, **48**, 53–79.
[31] Higham, N.J., 2002, *Accuracy and Stability of Numerical Algorithms* (2nd edn) (Philadelphia, PA: SIAM).
[32] Liu, D.C. and Nocedal, J., 1989, On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, **45**, 503–528.
[33] Griewank, A. and Toint, Ph.L., 1982, Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, **39**, 119–137.
[34] Barlow, J.L. and Toraldo, G., 1995, The effect of diagonal scaling on projected gradient methods for bound constrained quadratic programming problems. *Optimization Methods and Software*, **5**, 235–245.
[35] Nash, S.G. and Sofer, A., 1990, Assessing a search direction within a truncated-Newton method. *Operations Research Letter*, **9**, 219–221.
[36] Roma, M., 2002, Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization: the complete results. Technical Report R. 579, IASI-CNR. Revised 2003. Available at http://www.dis.uniroma1.it/∼roma/R579.pdf.
[37] Wang, W. and O'Leary, D.P., 2000, Adaptative use of iterative methods in predictor-corrector interior point methods for linear programming. *Numerical Algorithms*, **25**, 387–406.