A matrix-free approach to build band preconditioners for large-scale bound-constrained optimization[☆]

V. De Simone[a], D. di Serafino[a,b,*]

[a]*Dipartimento di Matematica e Fisica, Seconda Università di Napoli, viale A. Lincoln 5, I-81100 Caserta, Italy*
[b]*Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR), CNR, via P. Castellino 111, I-80131 Napoli, Italy*

Abstract

We propose a procedure for building symmetric positive definite band preconditioners for large-scale symmetric, possibly indefinite, linear systems, when the coefficient matrix is not explicitly available, but matrix-vector products involving it can be computed. We focus on linear systems arising in Newton-type iterations within matrix-free versions of projected methods for bound-constrained nonlinear optimization. In this case, the structure and the size of the matrix may significantly change in subsequent iterations, and preconditioner updating algorithms that exploit information from previous steps cannot be easily applied. Our procedure is based on a recursive approach that incrementally improves the quality of the preconditioner, while requiring a modest number of matrix-vector products. A strategy for dynamically choosing the bandwidth of the preconditioners is also presented. Numerical results are provided, showing the performance of our preconditioning technique within a trust-region Newton method for bound-constrained optimization.

Keywords: band preconditioners, matrix-free approach, bound-constrained nonlinear optimization.

2010 MSC: 65F08, 90C30.

1. Introduction

We are interested in the solution of linear systems

$$Hd = -g \tag{1}$$

[☆]This work was partially supported by INdAM-GNCS, under the 2012 Project *Advanced Numerical Methods for Preconditioning Linear Systems Arising from PDE and Optimization Problems* and the 2013 Project *Numerical Methods and Software for Large-Scale Optimization with Applications to Image Processing.*

[*]Corresponding author

Email addresses: `valentina.desimone@unina2.it` (V. De Simone), `daniela.diserafino@unina2.it` (D. di Serafino) that arise in Newton-type iterations within projected methods for large-scale optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{s. t.} & l \leq x \leq u, \end{array}$$

where $f \in C^2([l,u])$, $l \in \{\mathbb{R} \cup \{-\infty\}\}^n$, $u \in \{\mathbb{R} \cup \{+\infty\}\}^n$. In these methods H is usually (a symmetric approximation to) the reduced Hessian of f, and g is the opposite of the reduced gradient with respect to the free variables at the current step (see, e.g., [8, 12, 16, 21, 25, 26]).

We assume that the matrix H is sparse, is not available in a fully assembled form and/or is too large to fit into the main memory, but a procedure for computing matrix-vector products Hv is available. In this context, computing inexact Newton steps through variants of the Conjugate Gradient (CG) method able to deal with negative curvature directions [11, 20, 29] is a natural choice for the solution of system (1), since CG only uses the matrix through matrix-vector products. On the other hand, the convergence of CG depends on the spectral properties of the matrix, and preconditioning is usually needed to achieve reliable solutions in a resonable time.

Preconditioning in a matrix-free approach can be performed without explicitly assembling the preconditioner, i.e., in a fully matrix-free regime, or by explicitly building a preconditioner that uses a modest amount of memory, possibly $O(m)$, where m is the dimension of H. In both cases, $p \ll m$ matrix-vector products should be performed to obtain the preconditioner, to avoid that the improvement of the convergence speed is hidded by the cost of the preconditioner. When considering projected methods, it must be also taken into account that the working set, and hence the corresponding free variables, may significantly change from an iteration to the next one, hence reusing information from previous steps may not be easy.

A fully matrix-free approach for preconditioning sequences of linear systems is based on *quasi-Newton updating* techniques. The basic idea is to build an approximation to the Hessian or its inverse at the current outer iteration through BFGS updates of some previous Hessian approximation. More precisely, at iteration k, the (inverse) Hessian approximation $\widetilde{H}_k$ to be used as preconditioner at the next iteration can be obtained by a BFGS update of the previous approximation $\widetilde{H}_{k-1}$, by using a pair of correction vectors built from the last two Newton iterates x_k and x_{k-1}. The updated matrices are not computed explicitly, but only the vectors needed to perform the updates are stored; a limited-memory approach is generally used, where only p pairs, $p \ll n$, are considered [24]. Alternatively, the preconditioner for iteration k can be obtained by applying m BFGS updates to the Hessian approximation at step $k-1$, where the updates are based on correction vectors obtained as byproduct of the CG method applied to the Newton system at iteration $k-1$. The two BFGS approaches can be combined, as in the TN software [27]. The above approaches are low cost, but are usually effective on sequences of slowly varying systems; for large variations in the Hessian, the first matrix of the BFGS sequence of updates is usually reinitialized. Furthermore, in a projected framework, where the set of free variables changes until the active constraints at the solution have been identified, Hessian submatrices corresponding to different variables are considered at different iterations, thus the BFGS updates performed at a certain iteration may not be fully exploited at subsequent iterations.

A different approach, aimed at explicitly building preconditioners, employs *sparse* *matrix computation techniques.* Such techniques, which require the knowledge of the matrix sparsity pattern, date back to [10] and since then have been largely used to compute sparse Hessians (and Jacobians) when only matrix-vector products with these matrices, or approximations to the matrix-vector products via finite differences or automatic differentiation, are available (see the survey paper [17]). Generally speaking, the problem of efficiently computing the nonzero entries of a matrix H, with known sparsity pattern $\mathcal{S}(H)$, can be formulated as follows: given $\mathcal{S}(H)$, find a set of binary vectors $\{v_1, v_2, \ldots, v_p\}$, of minimum cardinality, such that the nonzero entries of H can be determined from the products $Hv_1, Hv_2, \ldots, Hv_p$. This problem is equivalent to finding a so-called structurally orthogonal partition of the columns of H that has the minimum number of column groups; if H is symmetric, as in the case we are interested in, symmetrically structurally orthogonal partitions can be used to reduce the number of matrix-vector products. As first recognized in [7], these problems can be formulated as graph coloring problems; their solution is NP-hard, but several algorithms are available to obtain practically effective colorings.

The previous concepts and techniques can be slightly modified for preconditioning purposes, to build an approximation to H made of a subset of the nonzero entries of the matrix, i.e., to perform a *partial matrix computation.* A further approximation can be obtained by choosing a simple sparsity pattern, e.g., through some sparsification of H, and then applying graph coloring algorithms to this pattern with matrix-vector products involving the whole matrix H [9]. The result is a so-called *partial matrix estimation*, i.e., an approximation to the matrix entries belonging to the selected pattern. Of course, the closer the sparsity pattern to the original one, the better the preconditioner is expected to be; on the other hand, more complete patterns usually correspond to higher computational costs for the computation and application of the preconditioner. We note that the computed preconditioner is usually applied through a factorization of it, therefore the choice of the sparsity pattern should take into account the cost for such factorization. We also observe that the preconditioner obtained through partial matrix computation or approximation may not preserve some property of the matrix H, e.g., positive definiteness, and a correction procedure must be applied if we wish to recover it. Finally, we note that the strategies proposed in [28] and [23, Approach (A3)] to build diagonal and band preconditioners, respectively, can be regarded as special cases of partial matrix estimation plus suitable corrections to obtain positive-definite matrices.

Recently, a preconditioner based on a *limited-memory partial Cholesky factorization* has been proposed in the context of matrix-free interior point methods for linear and quadratic programming [18] and extended to least squares problems [3]. The preconditioner is obtained by applying the Cholesky factorization to the k columns of the matrix H corresponding to the p largest pivots, with $p \ll m$, and using a suitable diagonal approximation to the remaining part of the matrix. If H is sparse, these columns can be computed with few matrix-vector products. However, we note that this approach requires the knowledge of the complete diagonal of H, thus limiting the application of the preconditioner.

We note that, when a sequence of linear systems must be solved, the explicit construction of a preconditioner in a matrix-free regime in principle allows for the application of preconditioner updating techniques such as those presented in [1, 2, 4, 5, 14], provided that information on the difference between the matrices of the sequence is available or can be estimated at a reasonable cost. This approach has been recently used in [15], where an approximation to the difference matrix needed by the updating algorithm is either explicitly computed by partial matrix estimation, or, in the case of separable functions providing the matrix-vector products, is implicitly obtained by exploiting separability. However, in projected methods, the part of the Hessian considered at each iteration may significantly change, thus making the previous techniques not easily applicable, and recomputing a cheap preconditioner for each system may result more efficient.

In this paper we present an approach for building symmetric positive definite band preconditioners for the matrix H by exploiting simple and low-cost partial matrix estimation techniques. This approach does not require a priori knowledge of the sparisty pattern of the matrix to be preconditioned and incrementally improves the quality of the preconditioner. Furthermore, we also present a strategy to dynamically choose the bandwidth of the preconditioner. Note that we focus on the construction of band preconditioners by using exact matrix-vector products, therefore we do not address any problems concerning the use of finite differences to estimate products involving the Hessian matrix.

The paper is organised as follows. In Section 2 our approach to approximate a band of a symmetric matrix is presented, focusing first on the case where the bandwidth of the approximation is fixed a priori, and then proposing a simple strategy to dynamically choose the bandwidth. In Section 3 some techniques for correcting the computed band approximations when they are not positive definite are reported. Numerical experiments showing the behaviour of the resulting preconditioners within the TRON optimization package [21] are presented in Section 4. Concluding remarks are given in Section 5.

Henceforth we use the following notation. For $\beta \geq 0$, we write A^β to denote a symmetric band matrix with half-bandwidth β, and $[A]^\beta$ to denote the band matrix obtained by extracting from a symmetric matrix A a band with half-bandwidth β; $\beta = 0$ identifies a diagonal matrix. We indicate the entries of any matrix A by either $a_{i,j}$ or $A_{i,j}$, and the entries of any vector v by either v_i or $(v)_i$, unless otherwise stated. We also denote by either $\mathrm{diag}(v)$ or $\mathrm{diag}(v_i)$ the diagonal matrix having the entries of v on the main diagonal. Finally, we denote by $\mathcal{S}(A)$ the sparsity pattern of A, i.e., $\mathcal{S}(A) = \{(i,j) : a_{i,j} \neq 0\}$, and by m the dimension of any matrix considered in the following.

2. A recursive approach for approximating a band of a matrix

To introduce our approach, we first recall that any symmetric band matrix $H \equiv H^\beta$ can be determined by computing the matrix-vector products

$$Hv_i^\beta, \quad i = 1, ..., \beta + 1, \tag{2}$$

where v_i^β is a vector with entries 0 and 1 defined as follows:

$$\left(v_i^\beta\right)_j = \begin{cases} 1 & \text{if } j = i + s(\beta+1) \text{ and } 0 \leq s \leq \lfloor (m-i)/(\beta+1) \rfloor, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Specifically, the part of the i-th row of H included in the upper triangle of the matrix is obtained as

$$\begin{aligned} h_{i,i} &= \left(Hv_k^\beta\right)_i, && (4)\\ h_{i,i+q} &= \left(Hv_r^\beta\right)_{i+q} \qquad\qquad \text{if } i+q-\beta-1<1, && (5)\\ h_{i,i+q} &= \left(Hv_r^\beta\right)_{i+q} - h_{i+q-\beta-1,i} \quad \text{otherwise}, && (6)\end{aligned}$$

where

$$k = \operatorname{mod}(i-1,\beta+1)+1, \quad r = \operatorname{mod}(k+q,\beta+1), \quad q=1,\ldots,\beta. \tag{7}$$

For example, it is easy to verify that a symmetric tridiagonal matrix can be built using only two matrix-vector products, involving the vectors $v_1^1 = (1,0,1,0,\ldots)^T$ and $v_2^1 = (0,1,0,1,\ldots)^T$.

If the sparsity pattern of H is unknown, a very simple approximation P^β to $[H]^\beta$ can be obtained by reasoning as if H were a band matrix with half-bandwidth β and computing it through (2)-(7). Of course, P^β may not be positive definite; we will come back to this issue in Section 3. We note that the strategy (A3) presented in [23] uses this simple approximation, coupled with a finite-difference estimation of the matrix-vector products, to build a band preconditioner. The diagonal preconditioner discussed in [28] also fits into this framework, since it computes the entries of the preconditioner by using matrix-vector products with the vector $v_1^0 = (1,1,\ldots,1)^T$. In both cases, suitable correction techniques are applied to obtain positive-definite preconditioners.

Our strategy starts from the above approximation P^β and improves it by exploiting the upper bound to the componentwise approximation error provided by the following proposition.

Proposition 2.1. *For all* $(i,j) \in \mathcal{S}(P^\beta)$, $i \le j$, *the componentwise error* $e_{i,j}^\beta = \left|p_{i,j}^\beta - h_{i,j}\right|$ *satisfies*

$$e_{i,j}^\beta \le \sum_{l\in\mathcal{I}_{i,j}^\beta} |h_{i,l}| + \bar{e}_{i,j}^\beta, \tag{8}$$

where

$$\mathcal{I}_{i,j}^\beta = \left\{l : (i,l) \in \mathcal{S}(H)\backslash\mathcal{S}(P^\beta),\ l = j \pm s(\beta+1),\ 1 \le s \le \lfloor (m-i)/(\beta+1)\rfloor\right\}$$

and

$$\bar{e}_{i,j}^\beta = \begin{cases} 0 & \text{if } i=j \text{ or } j-\beta-1<1,\\ e_{j-\beta-1,i}^\beta & \text{otherwise.}\end{cases} \tag{9}$$

Proof.. To simplify the notations we define

$$\mathcal{J}_j = \{l : l = j \pm s(\beta+1),\ 0 \le s \le \lfloor (m-i)/(\beta+1)\rfloor\}$$

and

$$\mathcal{S}_j^1 = \{l \in \mathcal{J}_j : (i,l) \in \mathcal{S}(H)\}, \qquad \mathcal{S}_j^2 = \left\{l \in \mathcal{J}_j : (i,l) \in \mathcal{S}([H]^\beta)\right\},$$
$$\mathcal{S}_j^3 = \left\{l \in \mathcal{J}_j : (i,l) \in \mathcal{S}(H)\backslash\mathcal{S}([H]^\beta)\right\},$$

where the dependence of all the sets on β and i is omitted. Let us consider first the computation of $p^{\beta}_{i,i}$ by (4), where k is defined in (7). We have

$$p^{\beta}_{i,i} = \left(Hv^{\beta}_{k}\right)_{i} = \sum_{l\in\mathcal{S}^1_i} h_{i,l} = \sum_{l\in\mathcal{S}^2_i} h_{i,l} + \sum_{l\in\mathcal{S}^3_i} h_{i,l} = h_{i,i} + \sum_{l\in\mathcal{S}^3_i} h_{i,l}$$

and hence

$$\left|p^{\beta}_{i,i} - h_{i,i}\right| \leq \sum_{l\in\mathcal{S}^3_i} |h_{i,l}| \, .$$

By observing that $\mathcal{S}^3_i = \mathcal{I}^{\beta}_{i,i}$, we get the thesis for $i = j$. With the same reasoning we can prove the thesis for $i < j$ and $j - \beta - 1 < 1$.

To prove the thesis for $i < j$ and $j - \beta - 1 \geq 1$, we observe that by (6) we have

$$\begin{aligned} p^{\beta}_{i,j} &= \left(Hv^{\beta}_{k}\right)_{j} - p^{\beta}_{j-\beta-1,i} \\ &= \sum_{l\in\mathcal{S}^2_j} h_{i,l} + \sum_{l\in\mathcal{S}^3_j} h_{i,l} - p^{\beta}_{j-\beta-1,i} + h_{j-\beta-1,i} - h_{j-\beta-1,i} \\ &= h_{i,j} + \sum_{l\in\mathcal{S}^3_j} h_{i,l} - p^{\beta}_{j-\beta-1,i} + h_{j-\beta-1,i} \end{aligned}$$

and hence

$$\left|p^{\beta}_{i,j} - h_{i,j}\right| \leq \sum_{l\in\mathcal{S}^3_i} |h_{i,l}| + \left|p^{\beta}_{j-\beta-1,i} - h_{j-\beta-1,i}\right| .$$

Therefore, we can conclude that the approximation error satisfies inequality (8), where $\bar{e}^{\beta}_{i,j}$ has the form specified in (9). □

Proposition 2.1 suggests a way to improve the approximation to $[H]^{\beta}$ provided by P^{β}. If $\beta = 2^k - 1$ and $\gamma = 2^r - 1$ with $r > k$, then for all $(i,j) \in \mathcal{S}\left(P^{\beta}\right)$ it is $\mathcal{I}^{\gamma}_{i,j} \subseteq \mathcal{I}^{\beta}_{i,j}$ and

$$\sum_{l\in\mathcal{I}^{\gamma}_{i,j}} |h_{i,l}| \leq \sum_{l\in\mathcal{I}^{\beta}_{i,j}} |h_{i,l}| \, ;$$

therefore, we expect that computing P^{γ} and extracting from it $[P^{\gamma}]^{\beta}$ may provide an approximation to $[H]^{\beta}$ more accurate than computing P^{β}.

We do not set γ a priori, but we choose it dynamically, by taking into account both the approximation error and the cost for building P^{γ}. Therefore, P^{γ} is built by using the recursive approach explained next. We first observe that, for $s \geq 1$, once the matrix-vector products needed to build $P^{2^{s-1}-1}$ have been computed, the 2^s matrix-vector products required by P^{2^s-1} can be obtained by explicitly performing the matrix-vector products

$$Hv^{2^s-1}_{i}, \quad i = 1, \ldots, 2^{s-1},$$

and by combining them with the products associated with $P^{2^{s-1}-1}$, as follows:

$$Hv^{2^s-1}_{i+2^{s-1}} = Hv^{2^{s-1}-1}_{i} - Hv^{2^s-1}_{i}, \quad i = 1, \ldots, 2^{s-1}.$$

Starting from $P^0 = \text{diag}\left(Hv_1^0\right)$, we iterate this procedure until the following inequality holds:

$$\left\| \left[P^{2^s-1}\right]^\beta - \left[P^{2^{s-1}-1}\right]^\beta \right\| \leq tol, \tag{10}$$

where $\|\cdot\|$ is a selected matrix norm and tol is a chosen tolerance, or a maximum number of steps, $maxs$, has been performed. This choice is motivated by the observation that in several optimization applications the Hessian matrix is highly sparse and then, by (8), a small number of iterations is often needed to compute a suitably accurate approximation to $[H]^\beta$. On the other hand, setting a maximum number of steps (i.e., a maximum value of γ) allows to keep the number of matrix-vector products for building P^γ much smaller than m. The algorithm for the construction of the preconditioner described so far is outlined next. It is immediate to verify that it requires a number of matrix-vector products equal to $\gamma + 1$.

Algorithm 1 Computation of an approximation to H^β with a priori choice of β.

```
choose $\beta$,  $tol$,  $maxs$
set $s = 0$,  $\gamma = 0$
compute $Hv_1^\gamma$ and build $P^\gamma$
**repeat**
  set $s = s + 1$,  $\gamma = 2^s - 1$,  $\delta = 2^{s-1} - 1$
  compute $Hv_i^\gamma$ by matrix-vector product,  $i = 1, \dots, \delta + 1$
  compute $Hv_{i+\delta+1}^\gamma = Hv_i^\delta - Hv_i^\gamma$,  $i = 1, \dots, \delta + 1$
  build $[P^\gamma]^\beta$ by applying (4)-(6)
**until** $\left(\delta \geq \beta \text{ and } \left\|[P^\gamma]^\beta - [P^\delta]^\beta\right\| \leq tol\right)$ or $(s = maxs)$
```

We note that, in practice, we found effective and computationally convenient to substitute the stopping criterion (10) with

$$\left\| d_i^{2^s-1} - d_i^{2^{s-1}-1} \right\| \leq \max\left(tola, tolr \left\| d_i^{2^s-1} \right\| \right), \quad i = 0, \dots, 2^s - 1, \tag{11}$$

where d_i^η denotes the i-th diagonal of P^η, $\|\cdot\|$ is the 2-norm, and $tola$ and $tolr$ are absolute and relative tolerances. Furthermore, we used the algorithm to compute $[P^\gamma]^\beta$ for any value of $\beta < (\gamma+1)/2$.

If $[P^\gamma]^\beta$ is positive definite, we can use it as a preconditioner for H, otherwise we must modify it to get positive definiteness (see Section 3). Letting $H = P + E + \bar{H}$, where P denotes $[P^\gamma]^\beta$ or its modification, $\bar{H} = H - [H]^\beta$ and $E = [H]^\beta - P$, we have

$$\left\| P^{-\frac{1}{2}} H P^{-\frac{1}{2}} - I \right\| \leq \|P^{-1}\| \left(\|\bar{H}\| + \|E\| \right),$$

where $\|\cdot\|$ is the 2-norm. The previous relation clearly shows that the quality of P depends on the size of the part of H outside the band, as well as on the error in the estimation of $[H]^\beta$ and the modification of the resulting matrix. Generally, the larger β the better the preconditioner is expected to be; on the other hand, the cost for building

P increases with β, therefore the choice of β should be the result of a tradeoff between effectiveness and computational cost. However, information for choosing a priori the best bandwidth may not be available. Then, a dynamic choice of β, somehow taking into account the accuracy in the approximation to the band of H and the cost for computing such approximation, may be useful.

In light of these considerations, Algorithm 1 can be modified as shown next.

Algorithm 2 Computation of an approximation to H^β with dynamic choice of β.

```
choose $tola, \ tolr, \ maxs, \ \beta_{max}$
set $s = 0, \ \gamma = 0, \ \beta = -1$
compute $Hv_1^\gamma$ and build $P^\gamma$
**repeat**
  set $s = s + 1, \ \gamma = 2^s - 1, \ \delta = 2^{s-1} - 1$
  compute $Hv_i^\gamma$ by matrix-vector product, $i = 1, \dots, \delta + 1$
  compute $Hv_{i+\delta+1}^\gamma = Hv_i^\delta - Hv_i^\gamma, \ i = 1, \dots, \delta + 1$
  build $P^\gamma$ by applying (4)-(6)
  set $j = 0, \ \beta_{old} = \beta$
  **while** $\|d_j^\gamma - d_j^\delta\| \leq \max\left(tola, tolr \, \|d_j^\gamma\|\right)$ and $j \leq \min(\delta, \beta_{max})$
    set $j = j + 1$
  **endwhile**
  **if** $j > 0$ **then** set $\beta = j - 1$
**until** $(\beta = \beta_{old}$ and $j > 0)$ or $(\beta = \beta_{max})$ or $(s = maxs)$
**if** $\beta = -1$ **then** set $\beta = \beta_{max}$
extract $[P^\gamma]^\beta$ from $P^\gamma$
```

At each step, we look for the maximum value of β such that $[P^\gamma]^\beta$ satisfies (11). The procedure stops when one of the following conditions holds: the value of β does not increase by increasing γ; a maximum number of steps, $maxs$, has been performed; β has reached a maximum value, β_{max}. The latter two conditions ensure that the computational cost of the preconditioner does not increase too much. We note that $\beta = \beta_{max}$ is also set if (11) is never satisfied; by numerical experiments we found that this choice is more effective than the choice $\beta = 0$. Finally, we observe that we can compute only the diagonals d_i^γ and d_i^δ used by the algorithm, instead of building the whole matrix P^γ at each step.

3. Getting positive definite band matrices

It has been previously noted that $[P^\gamma]^\beta$ may not be positive definite; in this case it has to be modified to be used as a preconditioner with CG-like methods. For the sake of completeness, we recall here the techniques that we use to get a positive-definite band matrix.

A simple and commonly employed technique to obtain a positive-definite matrix from any symmetric matrix A basically consists in adding to A a diagonal shift αI, where α

is a suitably large, positive constant. The selection of α is usually based on a "trial and error" approach, i.e., starting from an initial guess α_0, the value of α is increased until the Cholesky factorization of $A + \alpha I$ is terminated with success. We choose the technique applied in the context of the limited-memory incomplete Cholesky factorization algorithm proposed in [22]. Therefore, we first scale the matrix $[P^\gamma]^\beta$ as follows:

$$\hat{P} = D^{-\frac{1}{2}} \, [P^\gamma]^\beta \, D^{-\frac{1}{2}}, \quad D = \mathrm{diag}\left(\left\| [P^\gamma]^\beta \, e_i \right\| \right),$$

where e_i is the i-th standard unit vector and $\|\cdot\|$ is the 2-norm; then we compute $\hat{P} + \alpha I$ and attempt to perform its Cholesky factorization. If the factorization terminates with success, the factors of $\hat{P} + \alpha I$ are scaled back by $D^{\frac{1}{2}}$ and are taken for preconditioning, otherwise α is doubled, and the procedure is repeated until the factorization succeeds. The value of α_0 is 0 if $\min_i \hat{p}_{i,i} > 0$, and $-\min_i \hat{p}_{i,i} + \bar{\alpha}$, where $\bar{\alpha}$ is a suitably chosen positive value; $\bar{\alpha}$ is also used as upper bound on α.

For $\beta = 0, 1, 2$ we modify $[P^\gamma]^\beta$ using a different strategy, since we experimentally found that it generally leads to more effective preconditioners, in terms of both solver iterations and time. For simplicity of notation, let us denote by $\tilde{p}_{i,j}$ the (i, j)-th entry of $[P^\gamma]^\beta$. For $\beta = 0$, we set

$$p_{i,i} = \max\{|\tilde{p}_{i,i}|, \varepsilon_1\}, \tag{12}$$

where $\varepsilon_1 > 0$ is a suitably small value. For $\beta = 1, 2$ we use the approach described in [23], with some modifications. This approach is based on the following propositions.

Proposition 3.1. *Let A be a symmetric tridiagonal matrix with positive diagonal entries. If the matrices*

$$A_i = \begin{pmatrix} a_{i,i} & 2\,a_{i,i+1} \\ 2\,a_{i,i+1} & a_{i+1,i+1} \end{pmatrix} \quad i = 1, \ldots, m-1,$$

are positive semidefinite, then A is positive definite.

Proposition 3.2. *Let A be a symmetric pentadiagonal matrix with positive diagonal entries. If the matrices*

$$A_i = \begin{pmatrix} a_{i,i} & \frac{3}{2}\,a_{i,i+1} & 3\,a_{i,i+2} \\ \frac{3}{2}\,a_{i,i+1} & a_{i+1,i+1} & \frac{3}{2}\,a_{i+1,i+2} \\ 3\,a_{i,i+2} & \frac{3}{2}\,a_{i+1,i+2} & a_{i+2,i+2} \end{pmatrix}, \quad i = 1, \ldots, m-2,$$

are positive semidefinite, then A is positive definite.

From Proposition 3.1 it follows that $[P^\gamma]^1$ is positive definite if

$$\tilde{p}_{i,i} > 0, \qquad i = 1, \ldots, m, \tag{13}$$

$$\tilde{p}_{i,i}\,\tilde{p}_{i+1,i+1} - 4\,\tilde{p}_{i,i+1}^2 \geq 0, \qquad i = 1, \ldots, m-1, \tag{14}$$

Analogously, by Proposition 3.2, $[P^\gamma]^2$ is positive definite if (13) holds and

$$\tilde{p}_{i,i}\,\tilde{p}_{i+1,i+1} - \frac{9}{4}\,\tilde{p}_{i,i+1}^2 \geq 0, \qquad i = 1, \ldots, m-1, \tag{15}$$

$$\begin{aligned} &-9\,\tilde{p}_{i+1,i+1}\,p_{i,i+2}^2 + \frac{27}{2}\,\tilde{p}_{i,i+1}\,\tilde{p}_{i+1,i+2}\,\tilde{p}_{i,i+2} + \tilde{p}_{i,i}\,\tilde{p}_{i+1,i+1}\,\tilde{p}_{i+2,i+2} \\ &-\frac{9}{4}\left(\tilde{p}_{i,i}\,\tilde{p}_{i+1,i+2}^2 + \tilde{p}_{i+2,i+2}\,\tilde{p}_{i,i+1}^2\right) \geq 0, \qquad i = 1, \ldots, m-2. \end{aligned} \tag{16}$$

By exploiting (13)–(14), we can modify $[P^\gamma]^1$ to obtain a positive definite matrix P by setting $p_{i,i}$ as in (12), and $p_{i,i+1}$ as follows:

$$p_{i,i+1} = \begin{cases} \tilde{p}_{i,i+1} & \text{if (14) holds,} \\ \dfrac{\varepsilon_2}{2}\,\text{sign}(\tilde{p}_{i,i+1})\sqrt{p_{i,i}\,p_{i+1,i+1}} & \text{otherwise,} \end{cases} \tag{17}$$

where $0 < \varepsilon_1 < 1$ and $0 < \varepsilon_2 < 1$. We note that, differently from [23, Remark 3, item (3)], we choose the sign of $p_{i,i+1}$ according to the sign of $\tilde{p}_{i,i+1}$; furthermore, by using ε_2, we choose $p_{i,i+1}$ in the interior of the interval of values satisfying (14). For $\beta = 2$, to satisfy (13) and (15) we set $p_{i,i}$ as and $p_{i,i+1}$ as in (17), with $(2/3)\varepsilon_2$ instead of $\varepsilon_2/2$ in the second equality. Concerning $p_{i,i+2}$, we observe that given $\tilde{p}_{i,i}$ and $\tilde{p}_{i,i+1}$, (16) becomes a simple quadratic inequality:

$$a\,\tilde{p}_{i,i+2}^2 + b\,\tilde{p}_{i,i+2} + c \geq 0. \tag{18}$$

It is easy to verify that $a < 0$ and $b^2 - 4ac > 0$ and hence (18) is satisfied by any value $t \in [w_i, y_i]$, where w_i and y_i are the two roots of the left-hand side of (18). Therefore, following [23], we define

$$p_{i,i+2} = \begin{cases} \tilde{p}_{i,i+2} & \text{if (18) holds,} \\ \dfrac{w_i + y_i}{2} = \dfrac{3\,p_{i,i+1}\,p_{i+1,i+2}}{4\,p_{i+1,i+1}} & \text{otherwise.} \end{cases}$$

As observed in [23], Propositions 3.1 and 3.2 can be generalized to any symmetric band matrix with half-bandwidth $\beta > 2$: if suitable matrices of dimension $\beta + 1$ are postive semidefinite, then the band matrix is positive definite. However, as β grows, applying the technique based on the diagonal shift appears easier and cheaper.

4. Numerical experiments

We performed numerical experiments to analyse the behaviour of the preconditioners built with the approach described in the previous sections. As already observed, in this work we are not concerned with the problem of approximating matrix-vector products, e.g., through finite differences, therefore we tested our preconditioners within an optimization framework that does not require such approximations.

The preconditioners were applied within TRON, a Fortran 77 package which implements a trust-region Newton method for the solution of bound-constrained nonlinear optimization problems [21, 6]. TRON uses a gradient-projection method to generate a Cauchy step, then employs the Steihaug variant of the CG method preconditioned with a limited-memory incomplete Cholesky factorization to generate a descent direction for a suitable trust-region subproblem, and finally performs a projected search to compute the actual step. More precisely, the CG method is applied to solve subproblems of the form

$$\min\left\{\frac{1}{2}v^T H v + g^T v : \|DZv\| < \Delta\right\}, \tag{19}$$

where H and g are the reduced Hessian and gradient at the current iterate with respect to the free variables, D is a suitable matrix, and Z is made of the columns of the identity matrix corresponding to the free variables. Note that using D in the trust-region constraint

problem	*dimension*	*Hessian density*
BDEXP	5000	9.99e-4
BIGGSB1	1000	3.00e-4
DEGTRID	100001	3.00e-5
EXPQUAD	1200	2.49e-3
GRIDGENA	12482	1.40e-3
HADAMALS	4096	1.00e+0
JNLBRNG1	15625	3.17e-4
JNLBRNG2	15625	3.17e-4
JNLBRNGA	15625	3.13e-4
JNLBRNGB	15625	3.13e-4
MCCORMCK	10000	2.99e-4
MINSURFO	10406	6.64e-4
NONSCOMP	10000	2.99e-4
OBSTCLBL	15625	3.13e-4
OBSTCLBM	15625	3.13e-4
OBSTCLBU	15625	3.13e-4
ODNAMUR	11130	1.15e-1
TORSIONB	14884	3.33e-4
TORSIOND	14884	3.33e-4
TORSIONF	14884	3.33e-4

Table 1: Details on the test problems.

is equivalent to preconditioning H with D. In the original version of TRON the matrix D is computed by using the limited-memory incomplete Cholesky factorization implemented in ICFS [22], which requires that H is explicitly available. For our experiments TRON was modified to use the Hessian only through matrix-vector products, exploiting the CUTEr tools [19], and to apply our matrix-free band preconditioner instead of the ICFS one. A Cholesky factorization of the band preconditioner was computed to use it with the CG method. Both the construction and the factorization of the preconditioner were implemented in Fortran 95, to exploit dynamic memory management.

The experiments were carried out on 20 bound-constrained nonlinear optimization problems from the CUTEr collection. These problems are listed in Table 1, along with their dimensions and the densities of the (lower or upper) triangular parts of their Hessian matrices; the density of a triangular matrix T is defined here as $nnz(T)/(m(m+1)/2)$, where $nnz(T)$ and m are the number of nonzero entries and the dimension of T, respectively. For each problem of variable size, the dimension was chosen either as the maximum among the dimensions specified for that problem, or as the greatest dimension that could be decoded by our CUTEr implementation.

We report here the results obtained with Algorithms 1 and 2. In both cases, the maximum number of steps was set as $maxs = 6$, and the tolerances used in the stopping criterion as $tola = tolr = 10^{-3}$; $\beta_{max} = 2$ was chosen in Algorithm 2. For comparison purposes, we also run some tests using the band preconditioner with half-bandwidth β built from $\beta + 1$ matrix-vector products. In all the cases, $\varepsilon_1 = 10^{-6}$ and $\varepsilon_2 = 10^{-1}$ were used in the correction of the preconditioners with $\beta = 0, 1, 2$, while $\bar{\alpha} = 10^{-3}$ was used in the correction based on the diagonal shift. In the following, the preconditioner of half-bandwidth β built either with Algorithm 1 or with Algorithm 2 is denoted by $[P^\gamma]^\beta$, while the one obtained with $\beta+1$ matrix-vector products is denoted by P^β, although these notations have been previously used to indicate approximations to H^β without the modifications possibly performed to obtain positive definite matrices. Concerning TRON, a reduction of the 2-norm of the residual by a factor of 10^{-3} was required in the CG termination criterion, and the outer iterations were stopped when the 2-norm of the projected gradient of the objective function was smaller than 10^{-5} or a maximum number of 1000 iterations was achieved.

All the experiments were performed on an Intel Core 2 Duo E8500 processor, with clock frequency of 3.16 GHz, 4 GB of RAM, and 3 MB of cache memory, running the Debian Linux 6.0 operating system. CUTEr 2 was used as testing environment and all the software was compiled with gfortran 4.4.5.

To compare the results obtained with the various versions of the recursive algorithms, we use the performance profiles by Dolan and Moré [13]. Let $\mathcal{S}_{\mathcal{T},\mathcal{A}} \geq 0$ be a statistic corresponding to the successful solution of a test problem $\mathcal{T}$ by an algorithm $\mathcal{A}$, and suppose that the smaller this value is, the better the algorithm is considered; furthermore, let $\mathcal{S}_\mathcal{T}$ be the smallest value attained on the test $\mathcal{T}$ by one of the algorithms under analysis, and χ_M be a value such that $\chi_M > \mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_\mathcal{T}$ for all $\mathcal{T}$ and $\mathcal{A}$. The performance profile of the algorithm $\mathcal{A}$ is defined as

$$\pi(\chi) = \frac{\text{number of tests such that } \mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_\mathcal{T} \leq \chi}{\text{number of tests}}, \quad \chi \geq 1,$$

where $\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_\mathcal{T}$ is set equal to χ_M if the algorithm $\mathcal{A}$ fails in solving the test $\mathcal{T}$. In other words, $\pi(\chi)$ is the fraction of problems for which $\mathcal{S}_{\mathcal{T},\mathcal{A}}$ is within a factor χ of the smallest value $S_\mathcal{T}$. At $\chi = 1$ the performance profile gives the percentage of problems for which the algorithm $\mathcal{A}$ is the best, while $\lim_{\chi\to\chi_M^-} \pi(\chi)$ is the percentage of problems that are successfully solved by the algorithm $\mathcal{A}$. We note that when the values of $\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_\mathcal{T}$ have different magnitudes it may be convenient to use a logarithmic scale performance profile, i.e.,

$$\pi_{log}(\chi) = \frac{\text{number of tests such that } \log\left(\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_\mathcal{T}\right) \leq \chi}{\text{number of tests}}, \quad \chi \geq 0;$$

in the following we use the latter performance profile with a base-2 logarithm.

It is worth noting that, for all the test problems but HADAMALS and ODNAMUR, the optimal solutions obtained with the matrix-free version of TRON, using any of the preconditioners proposed in this paper, agree with the solutions computed by the original version. For HADAMALS and ODNAMUR we could not compare the solutions, because the original version of TRON failed to run because of too high memory requirements. We also note that the number of outer iterations performed by TRON using $[P^\gamma]^\beta$, with $\beta > 0$ fixed a priori or with the dynamic choice of β described in Algorithm 2, is about the same, except for three problems (BIGGSB1, EXPQUAD and HADAMALS). Using $[P^\gamma]^\beta$ with $\beta = 0$ significantly increases the number of outer iterations for some more problems, showing that the diagonal preconditioner may be less effective in providing accurate solutions to the subproblems (19). For most of the problems, the number of outer iterations performed by the original version of TRON is sligthly smaller.

Figure 1 shows the performance profiles of the preconditioner $[P^\gamma]^\beta$ built with Algorithm 1, which performs a dynamically chosen number of matrix-vector products, and of

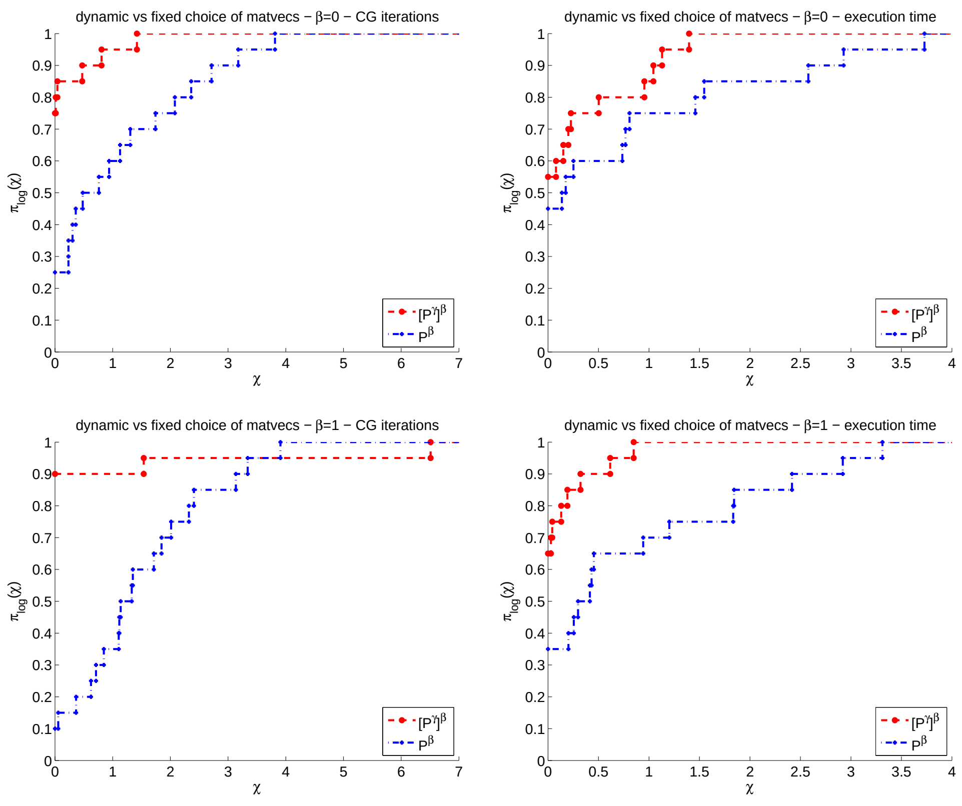

Figure 1: Performance profiles of the preconditioners $[P^{\gamma}]^{\beta}$ and P^{β}, for $\beta = 0$ (top) and $\beta = 1$ (bottom): total number of CG iterations (left) and execution time (right).

the preconditioner P^{β}, requiring exactly $\beta + 1$ matrix-vector products. The comparison is made in terms of the total number of CG iterations and the total execution time of TRON. The results concern the diagonal ($\beta = 0$) and the tridiagonal ($\beta = 1$) preconditioners, but are representative of larger bandwidths too. Note that P^0 is very close to the preconditioner presented in [28] (the latter differs from P^0 because $p_{i,i}$ is set to 1 if $\tilde{p}_{i,i} \leq \varepsilon_1$), while P^1 is very close to the preconditioner computed with the approach (A3) in [23] (this is formulated directly in terms of finite-difference approximations to the matrix-vector products involving the Hessian and uses a correction formula slightly different from (17)). We see that $[P^{\gamma}]^{\beta}$ is clearly superior in terms of both CG iterations and execution times. By a more detailed analysis of the results, we also found that the value of γ selected by our approach may vary during the TRON iterations, thus confirming the advisability of a dynamic choice of this parameter.

Figure 2 displays the performance profiles of $[P^{\gamma}]^{\beta}$ for different (fixed) values of β. We see that the tridiagonal and pentadiagonal preconditioners lead to the best iteration counts for 65% and 50% of the problems, respectively, but the percentage of problems for which they achieve the best execution time is smaller; on the other hand, the diagonal preconditioner, which is the less efficient in terms of iterations, achieves the smallest time for 35% of the problems, i.e., for the same number of problems as the tridiagonal

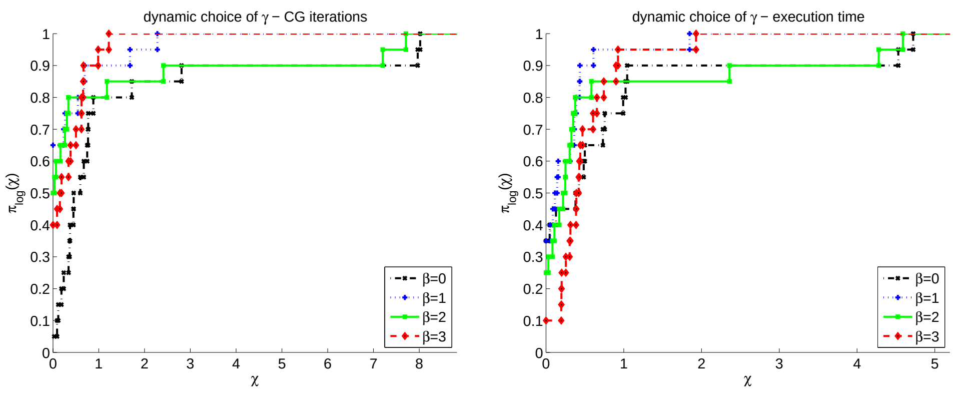

Figure 2: Performance profiles of the preconditioner $[P^\gamma]^\beta$, for $\beta = 0, 1, 2, 3$: number of CG iterations (left) and execution time (right).

preconditioner, thus showing that a very low-cost preconditioner may be convenient in some cases. We note that the correction used to have positive definite preconditioners for $\beta = 0, 1, 2$ is usually more effective than that applied for $\beta = 3$ (by using the diagonal shift correction for $\beta = 1, 2, 3$ we saw an improvement of the relative performance of the preconditioner with $\beta = 3$ over the other ones). This partly explains why the the eptadiagonal preconditioner requires more CG iterations than the tridiagonal and the pentadiagonal ones for 60% of the problems.

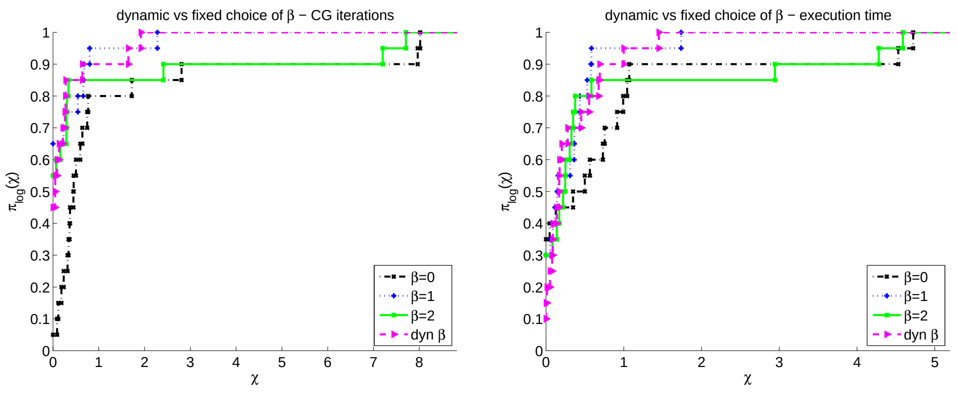

Figure 3: Performance profiles of the preconditioner $[P^\gamma]^\beta$ with fixed and dynamic choice of β: number of CG iterations (left) and execution time (right).

A comparison of the preconditioners obtained by fixing β a priori ($\beta = 0, 1, 2$) with the preconditioner where β is dynamically chosen is shown in Figure 3. The performance profiles confirm that our strategy for the dynamic selection of β allows to obtain a preconditioner that closely follows the behaviour of the band preconditioner that is the best for each problem.

5. Conclusions

We proposed a procedure for building symmetric positive definite band preconditioners suitable for linear systems arising in matrix-free Newton-type iterations within projected methods for large-scale bound-constrained nonlinear optimization. This procedure incrementally improves the approximation to a selected band of the coefficient matrix until a suitable stopping criterion is satisfied, which takes into account the change in the approximation and the computational cost. A dynamic choice of the bandwidth can be also performed, which is useful because information on the most suitable bandwidth may not be available. Finally, simple modifications are applied to the computed matrix if this is not positive definite.

Our preconditioners were implemented within a matrix-free version of the TRON package, which solves bound-constrained nonlinear optimization problems through a trust-region Newton method using projected searches, and were tested on 20 problems from the CUTEr collection. The results obtained show the effectiveness of our approach.

References

[1] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, Efficient preconditioner updates for shifted linear systems, SIAM J. Sci. Comput. 33 (2011) 1785-1809.

[2] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, A preconditioning framework for sequences of diagonally modified linear systems arising in optimization, SIAM J. Numer. Anal. 50 (2012) 3280-3302.

[3] S. Bellavia, J. Gondzio, B. Morini, A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems, SIAM J. Sci. Comput. 35 (2013) 192-211.

[4] S.Bellavia, B. Morini, M. Porcelli, New updates of incomplete LU factorizations and applications to large nonlinear systems, Optimization Methods and Software, in press. DOI:10.1080/10556788.2012.762517.

[5] M. Benzi, D. Bertaccini, Approximate inverse preconditioning for shifted linear systems, BIT Numer. Math. 43 (2003) 231-244.

[6] J.V. Burke, J.J. Moré, G. Toraldo, Convergence Properties of Trust Region Methods for Linear and Convex Constraints, Math. Program. 47 (1990) 305-336.

[7] T.F. Coleman, J.J. Moré, Estimation of sparse Jacobian matrices and graph coloring problems, SIAM J. Numer. Anal. 20 (1983) 187-209.

[8] A.R. Conn, N.I.M. Gould, Ph.L. Toint, Global convergence of a class of trust region algorithms for optimization with simple bounds, SIAM J. Numer. Anal. 25 (1988) 433-460.

[9] J. Cullum, M. Tůma, Matrix-Free Preconditioning Using Partial Matrix Estimation, BIT Numer. Math. 46 (2006) 711-729.

[10] A.R. Curtis, M.J.D. Powell, J.K. Reid, On the estimation of sparse Jacobian matrices, J. Inst. Maths. Applics. 13 (1974) 117-119.

[11] R.S. Dembo, T. Steihaug, Truncated Newton algorithms for large-scale optimization, Math. Programming 26 (1982) 190-212.

[12] M. De Santis, G. Di Pillo, S. Lucidi, An active set feasible method for large-scale minimization problems with bound constraints, Comput. Optim. Appl. 53(2) (2012) 395-423.

[13] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Programming 91(2) (2002) 201-213.

[14] J. Duintjer Tebbens, M. Tůma, Preconditioner updates for solving sequences of large and sparse nonsymmetric linear systems, SIAM J. Sci. Comput. 29 (2007) 1918-1941.

[15] J. Duintjer Tebbens, M. Tůma, Preconditioner updates for solving sequences of linear systems in matrix-free environment, Numerical Linear Algebra with Applications 17 (2010) 997-1019.

[16] F. Facchinei, S. Lucidi, L. Palagi, A Truncated Newton Algorithm for Large Scale Box Constrained Optimization, SIAM J. Optim. 12(4) (2002) 1100-1125.

[17] A.H. Gebremedhin, F. Manne, A. Pothen, What Color Is Your Jacobian? Graph Coloring for Computing Derivatives, SIAM Review 47(4) (2005) 629-705.

[18] J. Gondzio, Matrix-free interior point method, Comput. Optim. Appl. 51 (2012) 457-480.
[19] N.I.M. Gould, D. Orban, Ph.L. Toint, CUTEr and SifDec: A constrained and unconstrained testing environment, revisited, ACM Trans. Math. Software 29(4) (2003) 373-394.
[20] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal. 23 (1986) 707-716.
[21] C.J. Lin, J.J. Moré, Newton's method for large bound-constrained optimization problems, SIAM J. Optim. 9(4) (1999) 1100-1127.
[22] C.J. Lin, J.J. Moré, Incomplete Cholesky Factorizations With Limited Memory, SIAM J. Sci. Comput. 21(1) (1999) 24-45.
[23] L. Lukšan, C. Matonoha, J. Vlček, Band preconditioners for the matrix free truncated Newton method. Technical Report n. 1079, Institute of Computer Science, Academy of Sciences of the Czech Republic (2010).
[24] J.L. Morales, J. Nocedal, Automatic Preconditioning by Limited Memory Quasi-Newton Updating, SIAM J. Optim. 10(4) (1200) 1079-1096.
[25] J.J. Moré, G. Toraldo, On the solution of large quadratic programming problems with bound constraints, SIAM J. Optim. 1 (1991) 93-113.
[26] S.G. Nash, User's guide for TN/TNBC, Technical Report 397, Department of Mathematical Sciences, The Johns Hopkins University, Baltimore(1984).
[27] S.G. Nash, Preconditioning of truncated-Newton methods, SIAM J. Sci. Stat. Comput. 6 (1985) 599-616.
[28] M. Roma, A dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization, Optim. Meth. Soft. 20 (2005) 693-713.
[29] T. Steihaug, The coniugate gradient method and trust regions in large scale optimization, SIAM J. Numer. Anal. 20(3) (1983) 626-637.