International Journal of Computer Mathematics
Vol. 84, No. 1, January 2007, 89–108

Taylor & Francis
Taylor & Francis Group

Preconditioners for the conjugate gradient algorithm using Gram–Schmidt and least squares methods

JULIEN STRAUBHAAR*

Institut de Mathématiques, Université de Neuchâtel, Rue Emile-Argand 11,
CH-2007 Neuchâtel, Switzerland

(Received 21 April 2006; revised version received 3 November 2006; accepted 4 December 2006)

This paper is devoted to the study of some preconditioners for the conjugate gradient algorithm used to solve large sparse linear and symmetric positive definite systems. The construction of a preconditioner based on the Gram–Schmidt orthogonalization process and the least squares method is presented. Some results on the condition number of the preconditioned system are provided. Finally, numerical comparisons are given for different preconditioners.

Keywords: Conjugate gradient method; Preconditioner; Gram–Schmidt orthogonalization; Least squares; Condition number

AMS Subject Classifications: 65F10; 65F35; 65F50

1. Introduction

Let us first briefly present the conjugate gradient (CG) algorithm. It is an iterative method used to solve the linear system

$$Ax = b, \tag{1}$$

where $A = (a_{ij})$ is a real symmetric positive definite (SPD) $n \times n$ matrix and b a vector in $\mathbb{R}^n$. *Approximate solutions* $x_k \in \mathbb{R}^n$ and their *residual vector* $r_k = b - Ax_k$ $(k = 0, 1, \ldots)$ are computed. The approximate solution x_N is chosen when the norm of its associated residual is lower than a fixed tolerance *tol*.

The CG algorithm, due to Hestenes and Stiefel [1–3], consists of finding the minimum $\hat{x} = A^{-1}b$ of the application

$$f(x) = \|x - \hat{x}\|_A^2 = (x|Ax) - 2(b|x) + (b|\hat{x}),$$

where $(.|.)$ denotes the usual inner product in $\mathbb{R}^n$, and $\|.\|_A$ the norm derived from the inner product $(x|y)_A := (x|Ay)$. The approximate solution x_{k+1} is obtained as the minimum of f,

*Email: julien.straubhaar@unine.ch

International Journal of Computer Mathematics
ISSN 0020-7160 print/ISSN 1029-0265 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207160601173621

starting from x_k and in the d_k-vector direction. The starting point x_0 is chosen arbitrarily, and d_0 is set to r_0. Then, for $k \geqslant 0$, the real number β_k is determined in order that the *descent direction* $d_{k+1} := r_{k+1} + \beta_k d_k$ is A-orthogonal to d_k (i.e. $(d_{k+1}|d_k)_A = 0$).

Let us give some properties of the CG algorithm (see, for example, [1–3]). Each vector d_k is A-orthogonal to each other, i.e. $(d_i|d_j)_A = 0$ if $i \neq j$ (we say that the vectors d_k are conjugate). Each residual vector r_k is orthogonal to each other (for the usual inner product); this implies that the CG algorithm converges after at most n iterations. However, in practice, it is possible that the CG algorithm does not converge after n iterations because of the fixed precision for the representation of real numbers in computers. Moreover, even if the CG algorithm converges, when n is large, the number of iterations is, in general, too large.

The following theorem gives an idea of the convergence rate of the CG algorithm.

THEOREM 1.1 [3, p. 194] *If $\hat{x} = A^{-1}b$ is the solution of* (1), *then the approximate solutions x_m satisfy*

$$\|x_m - \hat{x}\|_A \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|x_0 - \hat{x}\|_A,$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$ is the condition number of the matrix A, i.e. the ratio of its largest and smallest eigenvalues.

Therefore, if the condition number κ is close to one, the convergence is fast.

In order to improve the convergence rate of the CG algorithm, system (1) is replaced by the equivalent system

$$TAT^t\tilde{x} = Tb, \quad x = T^t\tilde{x}, \tag{2}$$

where T is an $n \times n$ regular matrix. Then the CG algorithm is applied to this new system whose matrix TAT^t is still SPD. The matrix T is called a *preconditioner* of A. The aim is to construct preconditioners that reduce as much as possible the condition number of the system.

In practice (modelling many physical phenomena, for instance), the matrix A is large and sparse, that is, having a large proportion of its coefficients equal to zero. For memory reasons and time computation, the preconditioner should also be sparse.

For the preconditioned system (2), the following CG algorithm is used [2, 3]:

Preconditioned CG algorithm
Let $x_0 \in \mathbb{R}^n$
Compute $r_0 = b - Ax_0$, $s_0 = T^t \cdot Tr_0$ and set $d_0 = s_0$
If $\|r_0\|/\|b\| < tol$: quit
For $k \geqslant 0$, do:
  $\tilde{\alpha}_k = (r_k|s_k)/(d_k|Ad_k)$
  $x_{k+1} = x_k + \tilde{\alpha}_k d_k$
  $r_{k+1} = r_k - \tilde{\alpha}_k Ad_k$
  If $\|r_{k+1}\|/\|b\| < tol$: quit the loop
  $s_{k+1} = T^t \cdot Tr_{k+1}$
  $\tilde{\beta}_k = (r_{k+1}|s_{k+1})/(r_k|s_k)$
  $d_{k+1} = s_{k+1} + \tilde{\beta}_k d_k$
End for k

Remark 1 Another way to precondition (1) consists of considering an $n \times n$ SPD (sparse) matrix M and replacing (1) by the equivalent system

$$M^{-1}Ax = M^{-1}b. \tag{3}$$

This system can be rewritten like (2) with $T = M^{-1/2} = QD^{-1/2}Q^t$, where QDQ^t is an orthogonal diagonalization of M. With this preconditioning methodology, a linear system of matrix M must be solved in each step of the CG algorithm (see the computation of vectors s_k in the preconditioned CG algorithm); for this reason, the CG algorithm is, in general, not parallelizable.

A classical way of preconditioning (1) consists of using the *incomplete Cholesky factorization* (or *ilu factorization*) of A.

THEOREM 1.2 [3, 4] *Let A be an $n \times n$ symmetric M-matrix, i.e. its non-diagonal coefficients are non-positive and A is monotone (regular and the coefficients of A^{-1} are non-negative), and let S be a symmetric set of indices with no diagonal index. Then there is a unique lower triangular matrix $L = (l_{ij})$ and a unique symmetric matrix $R = (r_{ij})$ with non-negative coefficients such that $l_{ij} = 0$ if $(i, j) \in S$, $r_{ij} = 0$ if $(i, j) \notin S$ and such that $A = LL^t - R$ is a regular splitting (i.e. LL^t is monotone and the coefficients of R are non-negative).*

Choosing $S = \{(i, j) \mid a_{ij} = 0\}$, the computation of the matrix L [3, 4] in this theorem provides a preconditioner $M = LL^t$ for system (3).

With the incomplete Cholesky factorization, the CG algorithm has a good convergence rate when it does not break down. However, the computation of the matrix L and the corresponding CG algorithm are not parallelizable. Hence, this methodology becomes impracticable with very large matrices.

Let us now present another preconditioner developed in [5]. The idea is to construct a *factorized sparse approximate inverse* of A. Let $A = L_A L_A^t$ be the Cholesky factorization of A. A sparse approximate matrix G of L_A^{-1} is constructed in the following way. Let P be a set of indices included in the lower part of the diagonal, $\{(i, i) \mid 1 \leqslant i \leqslant n\} \subset P \subset \{(i, j) \mid 1 \leqslant i \geqslant j \leqslant n\}$. We minimize the Frobenius norm $\|I - GL_A\|_F = (\mathrm{Tr}((I - GL_A)(I - GL_A)^t))^{1/2}$ with the constraints $G_{ij} = 0$ if $(i, j) \notin P$. This leads to the equations $(GL_A L_A^t)_{ij} = (L_A^t)_{ij}$, $(i, j) \in P$, i.e. $(GA)_{ij} = 0$ for $(i, j) \in P$, $i \neq j$, and $(GA)_{ii} = (L_A)_{ii}$ for all i. Since the diagonal coefficients of L_A are unknown, the matrix $\widetilde{G}$ such that

$$(\widetilde{G}A)_{ij} = 0, \quad \text{if } (i, j) \in P, \ \ i \neq j, \tag{4}$$

$$(\widetilde{G}A)_{ii} = 1, \quad \text{for } i = 1, \ldots, n, \tag{5}$$

$$\widetilde{G}_{ij} = 0, \quad \text{if } (i, j) \notin P, \tag{6}$$

is considered. Then, we set $G = D\widetilde{G}$, where D is the diagonal matrix such that $(GAG^t)_{ii} = 1$ for all i. Choosing $P = \{(i, j) \mid i \geqslant j, \ a_{ij} \neq 0\}$, the matrix $T = G$ is a preconditioner for (2).

Equations (4)–(6) form n independent systems: one for each row of $\widetilde{G}$; thus the construction of this preconditioner is readily parallelizable. However, it is very difficult to answer the following question: how to choose the set P to improve the preconditioner? Moreover, some preconditioning techniques can be found in [6].

In the following sections, a preconditioner is presented using the Gram–Schmidt orthogonalization process and least squares method. For its construction, some parameters can be modified and the filling is controlled; this guarantees some flexibility and avoids strong filling in the preconditioner. Moreover, the algorithm for its construction is readily parallelizable.

Some results for the condition number of the preconditioned system are presented. Finally, numerical comparisons are given for the different preconditioners described.

2. Conjugate Gram–Schmidt algorithm

This section is devoted to the construction of a preconditioner based on the conjugate Gram–Schmidt method [7].

The A-orthogonalization Gram–Schmidt process (i.e. the Gram–Schmidt process with the inner product $(.|.)_A$) is applied to the canonical basis $\{e_1, \ldots, e_n\}$ in $\mathbb{R}^n$. We then obtain the A-orthogonal basis $\{z_1, \ldots, z_n\}$, where

$$z_1 = e_1,$$
$$z_k = e_k - \sum_{i=1}^{k-1} \frac{(e_k|z_i)_A}{(z_i|z_i)_A} z_i, \quad k = 2, \ldots, n. \tag{7}$$

The matrix $Z = (z_1, \ldots, z_n)$, whose columns are the vectors z_k, is upper triangular, unitary diagonal, and satisfies the relation $Z^t \cdot A \cdot Z = D$, where $D = \operatorname{diag}(p_1, \ldots, p_n)$, with $p_k = (z_k|z_k)_A = z_k^t \cdot A \cdot z_k = \|z_k\|_A^2$, $k = 1, \ldots, n$.

Let

$$p_{ik} = (e_i|z_k)_A = (a_i|z_k),$$

where $a_i = Ae_i$ denotes the ith column of A. Since the vectors z_k are mutually A-orthogonal, $p_k = p_{kk}$. The conjugate Gram–Schmidt algorithm (CGS) can be written as follows.

CGS algorithm
$z_i^{(0)} = e_i, \ i = 1, \ldots, n$
For $k = 1, \ldots, n$, do:
  For $i = k, \ldots, n$, do:
    $p_i^{(k)} = (a_i \mid z_k^{(k-1)})$
  End for i
  For $i = k+1, \ldots, n$, do:
    $z_i^{(k)} = z_i^{(k-1)} - (p_i^{(k)}/p_k^{(k)}) z_k^{(k-1)}$
  End for i
End for k

We have $p_i^{(k)} = p_{ik}$ and, finally, $z_k = z_k^{(k-1)}$ and $p_k = p_k^{(k)}$. Note that the fourth row in the CGS algorithm can be replaced by $p_i^{(k)} = (a_k \mid z_i^{(k-1)})$. In fact, using (7) and the symmetry of A, we have

$$\begin{aligned}(a_i \mid z_k^{(k-1)}) &= (a_i|e_k) - \sum_{j=1}^{k-1} \frac{(a_k|z_j)}{p_j}(a_i|z_j) \\ &= \left(a_k \middle| e_i - \sum_{j=1}^{k-1} \frac{(a_i|z_j)}{p_j} z_j \right) \\ &= (a_k \mid z_i^{(k-1)}).\end{aligned}$$

The CGS algorithm provides the inverse $A^{-1} = Z \cdot D^{-1} \cdot Z^t$ of A and, with $T = D^{-1/2} \cdot Z^t$, we have $TAT^t = I$.

2.1 *Obtaining a preconditioner*

The idea for obtaining a preconditioner of (1) is to consider an approximation of the matrix Z (still denoted Z). It provides, with the corresponding diagonal matrix $D = \operatorname{diag}((z_1|Az_1), \ldots, (z_n|Az_n))$, an approximation $Z \cdot D^{-1} \cdot Z^t$ of A^{-1} and a preconditioner $T = D^{-1/2} \cdot Z^t$ for system (2), satisfying $TAT^t \approx I$.

Let P be a set of indices included in the upper part of the diagonal,

$$\{(i, i) \mid 1 \leqslant i \leqslant n\} \subset P \subset \{(i, j) \mid 1 \leqslant i \leqslant j \leqslant n\}, \tag{8}$$

and we fix

$$Z_{ij} = 0, \quad \text{if } (i, j) \notin P. \tag{9}$$

Several ways can be considered to determine the coefficients Z_{ij}, $(i, j) \in P$, and this is the purpose of the following sections.

3. Incomplete conjugate Gram–Schmidt algorithm

One of the ideas in [7] consists of using the CGS algorithm, ignoring the coefficients in Z whose indices are not in P, where P, satisfying (8) and (9), is given. We obtain the following algorithm.

INC CGS algorithm
$Z = I$ (initialization)
For $k = 1, \ldots, n$, do:
  For $i = k, \ldots, n$, do:
    $p_i = a_{ik}$
    For $j = 1, \ldots, k-1,$ do:
      If $(j, i) \in P$: $p_i = p_i + a_{jk} z_{ji}$
    End for j
  End for i
  For $i = k+1, \ldots, n$, do:
    $t = p_i / p_k$
    For $j = 1, \ldots, k,$ do:
      If $(j, i) \in P$ and $(j, k) \in P$: $z_{ji} = z_{ji} - t z_{jk}$
    End for j
  End for i
End for k

$p_i = a_{ik}$; For $j = 1, \ldots, k-1,$ do: If $(j, i) \in P$: $p_i = p_i + a_{jk} z_{ji}$; End for j $\Big\}$ $p_i^{(k)} = (a_k \mid z_i^{(k-1)})$

$t = p_i / p_k$; For $j = 1, \ldots, k,$ do: If $(j, i) \in P$ and $(j, k) \in P$: $z_{ji} = z_{ji} - t z_{jk}$; End for j $\Big\}$ $z_i^{(k)} = z_i^{(k-1)} - t z_k^{(k-1)}$

If matrix A is sparse, a choice for P can be $P = \{(i, j) \mid 1 \leqslant i \leqslant j \leqslant n,\ a_{ij} \neq 0\}$. This guarantees a reasonable computation time for the construction of this preconditioner.

4. Approximation with the least squares method

Now consider P satisfying (8); our aim is to construct the upper triangular and unitary diagonal matrix Z with the constraints (9). Assuming that the columns $z_1, \ldots, z_{k-1}$ are known, let us show how to construct the kth column z_k. Let $z_k(k) = Z_{kk} = 1$ and let

$$\mathcal{J} = \mathcal{J}_k = \{j \mid (j,k) \in P,\ j \neq k\}$$

be the indices set of the components of z_k to be determined. Assuming that $\mathcal{J}$ is non-empty, set $\mathcal{J} = \{j_1 < \cdots < j_p\} \subset \{1, \ldots, k-1\}$ and

$$y = (y_1, \ldots, y_p)^t = z_k(\mathcal{J}) = (Z_{j_1 k}, \ldots, Z_{j_p k})^t. \tag{10}$$

We look for z_k such that z_k is A-orthogonal to the vectors $z_1, \ldots, z_{k-1}$:

$$(z_k | A z_i) = 0, \quad i = 1, \ldots, k-1. \tag{11}$$

With previous notation

$$(z_k | A z_i) = (A z_k | z_i) = (a_k | z_i) + \sum_{l=1}^{p} y_l (a_{j_l} | z_i)$$

(where a_{j_l} denotes the j_lth column of A). So (11) can be rewritten as

$$\sum_{l=1}^{p} (a_{j_l} | z_i) y_l = -(a_k | z_i), \quad i = 1, \ldots, k-1.$$

This is a linear system with $k-1$ equations and p unknown variables $(y_1, \ldots, y_p)$; in matrix form, we have

$$By = c, \tag{12}$$

where $B = (b_{il})$ is a $(k-1) \times p$ matrix, with $b_{il} = (a_{j_l} | z_i)$ and $c = (c_1, \ldots, c_{k-1})^t$ is the vector in $\mathbb{R}^{k-1}$, with $c_i = -(a_k | z_i)$.

Let $Z_{k-1} = (Z_{ij})_{1 \leqslant i,j \leqslant k-1}$ be the principal submatrix of Z of order $k-1$. Also let $\widetilde{A}$ be the $(k-1) \times p$ matrix obtained by keeping in A the $k-1$ first rows and columns $j_1, \ldots, j_p$ ($\widetilde{A}_{il} = a_{ij_l}$) and $\tilde{a}_k$ the vector in $\mathbb{R}^{k-1}$ made up of the $k-1$ first components of a_k (the kth column of A). Since the components $k, k+1, \ldots, n$ of the vectors $z_1, \ldots, z_{k-1}$ are equal to zero, we have

$$B = Z_{k-1}^t \cdot \widetilde{A},$$

and

$$c = -Z_{k-1}^t \cdot \tilde{a}_k.$$

The matrix Z_{k-1} is regular (upper triangular and unitary diagonal), hence (12) is equivalent to

$$\widetilde{A} y = -\tilde{a}_k. \tag{13}$$

In general, this system has no solution since $p \leqslant k-1$. We seek a solution in the least squares sense, i.e. the vector $y \in \mathbb{R}^p$ that minimizes

$$\| \widetilde{A} y + \tilde{a}_k \|. \tag{14}$$

This solution y is given by the system (see, for example, [8, pp. 106–107])

$$\widetilde{A}^t \cdot \widetilde{A} y = -\widetilde{A}^t \cdot \tilde{a}_k. \tag{15}$$

Since matrix A is SPD, it is still the case for A_{k-1}, consequently its columns are linearly independent and $\widetilde{A}^t \cdot \widetilde{A}$ is SPD and (15) has a unique solution. To compute it, we can use, for example, the Cholesky decomposition of the matrix $\widetilde{A}^t \cdot \widetilde{A}$, which is of order p.

Equation (15) is solved by using the QR decomposition of $\widetilde{A}$. Since the columns $\tilde{a}_1, \ldots, \tilde{a}_p$ of $\widetilde{A}$ are linearly independent, we have $\widetilde{A} = QR$, where $Q = (q_1, \ldots, q_p)$ is a $(k-1) \times p$ matrix satisfying $Q^t \cdot Q = I_p$ and $R = (r_{ij})$ is a regular square and upper triangular matrix of order p. The matrices Q and R are obtained from the orthogonalization Gram–Schmidt process applied to the family $\{\tilde{a}_1, \ldots, \tilde{a}_p\}$ (see, for instance, [3, p. 11] for the algorithm of QR decomposition).

To obtain a solution of (15) it suffices to solve

$$Ry = -Q^t \tilde{a}_k. \tag{16}$$

Indeed, since $Q^t \cdot Q = I$, we have

$$\widetilde{A}^t \cdot \widetilde{A} y = -\widetilde{A}^t \tilde{a}_k \iff R^t Q^t \cdot QRy = -R^t Q^t \tilde{a}_k \iff Ry = -Q^t \tilde{a}_k.$$

Let us recall that the vector $y \in \mathbb{R}^p$ minimizing (14) is $y = z_k(\mathcal{J})$ (see (10)), i.e. the vector $\tilde{z}_k = (Z_{1k}, \ldots, Z_{k-1,k})^t$, with $Z_{jk} = y_l$, $l = 1, \ldots, p$, and $Z_{ik} = 0$ if $i \notin \mathcal{J}$, realizes the minimum

$$m = \min_{\substack{u \in \mathbb{R}^{k-1} \\ u_i = 0,\, i \notin \mathcal{J}}} \|A_{k-1} u + \tilde{a}_k\|. \tag{17}$$

It remains to choose, for each $k \in \{2, \ldots, n\}$, the set of indices $\mathcal{J}_k$ for the components of the kth column of Z, which can be non-zero (save the diagonal coefficient Z_{kk}) (or the pattern $P = \cup_{k=2}^{n} \mathcal{J}_k \cup \{(k,k) \mid k = 1, \ldots, n\}$).

Remark 2 In the computation of the kth column z_k of the matrix Z we obtain (13), where the previous columns $z_1, \ldots, z_{k-1}$ no longer appear; thus, the columns of Z can be computed *independently*, which leads to an immediate parallel algorithm for this method.

4.1 *Choosing the non-zero coefficients*

Let us propose different ways to choose the sets $\mathcal{J}_k$ (or the pattern P). We remark first that, in order to obtain a reasonable computing time for QR decomposition of the system (13), it is necessary that the number of indices in each $\mathcal{J}_k$ is sufficiently small.

A-filling The simplest choice for P is to take

$$P = \{(i, j) \mid 1 \leqslant i \leqslant j \leqslant n,\ a_{ij} \neq 0\}.$$

When the matrix A is sparse, the cardinality of each $\mathcal{J}_k$ is small.

Diagonal filling Another possibility is to fix the maximal cardinality $p_{\max}$ of each $\mathcal{J}_k$ and consider a filling of Z in the neighbourhood of the diagonal,

$$P = \{(i, j) \mid 0 \leqslant j - i \leqslant p_{\max}\}.$$

Optimal filling Let us choose the set $\mathcal{J}_k$ more judiciously. For that, we use the ideas in [9]. Let us fix k, $2 \leqslant k \leqslant n$. The set $\mathcal{J} = \mathcal{J}_k$ and the vector z_k are constructed so that the minimum m

of (17) is smaller or equal to a given number ε. For this, the maximal number $p_{\max}$ of indices in $\mathcal{J}$ and the 'additional filling' s ($1 \leqslant s \leqslant p_{\max}$) are fixed; we then proceed in the following way:

(i) start with $\mathcal{J} = \phi$;
(ii) compute the vector $\tilde{z}_k$ realizing the minimum m of (17);
(iii) if $m \leqslant \varepsilon$ or $|\mathcal{J}| \geqslant p_{\max}$, quit the loop; and
(iv) add s indices to $\mathcal{J}$ and go to (ii).

Once out of the loop, we set $z_k = (\tilde{z}_k^t, 1, 0, \ldots, 0)^t \in \mathbb{R}^n$, then $|\mathcal{J}| \leqslant p_{\max} + s - 1$. Note that the condition $m \leqslant \varepsilon$ in (ii) avoids a strong filling in Z.

It remains to describe in point (iv) the choice of the additional indices for a given set $\mathcal{J}$. The aim is to select the indices that reduce at most the minimum of (17). Let

$$r = A_{k-1}\tilde{z}_k + \tilde{a}_k,$$

where $\tilde{z}_k$ realizes the minimum of (17), i.e. $\|r\| = \min\{\|A_{k-1}u + \tilde{a}_k\| \mid u \in \mathbb{R}^{k-1},\ u_i = 0,\ i \notin \mathcal{J}\}$. Let us consider the subspace $W = \langle A_{k-1}e_j \mid j \in \mathcal{J}\rangle$ included in $\mathbb{R}^{k-1}$; since $A_{k-1}\tilde{z}_k$ is the best approximation of $-\tilde{a}_k$ in W, r is orthogonal to W (see, for instance, [10, p. 217]), i.e.

$$(r|A_{k-1}e_j) = 0, \quad \forall\ j \in \mathcal{J}. \tag{18}$$

Let $\mathcal{L} = \{1 \leqslant l \leqslant k-1 \mid r_l \neq 0\}$ and, for each $l \in \mathcal{L}$, set $\mathcal{M}_l = \{1 \leqslant j \leqslant k-1 \mid a_{lj} \neq 0,\ j \notin \mathcal{J}\}$. Then,

$$\widetilde{\mathcal{J}} = \bigcup_{l \in \mathcal{L}} \mathcal{M}_l \tag{19}$$

is the set of 'useful' indices to add to $\mathcal{J}$ in order to reduce $\|r\|$. Note that if $r \neq 0$, then $\widetilde{\mathcal{J}} \neq \phi$. Indeed, assume $\widetilde{\mathcal{J}} = \phi$; then, for all $l \in \mathcal{L}$, $\mathcal{M}_l = \phi$, i.e. $a_{lj} = 0$ if $j \notin \mathcal{J}$. Hence, $(r|A_{k-1}e_j) = \sum_{l \in \mathcal{L}} r_l a_{lj} = 0$, for all $j \notin \mathcal{J}$. Thus, using (18), r is orthogonal to all columns of A_{k-1} and, since A_{k-1} is regular, it follows that $r = 0$.

For each $j \in \widetilde{\mathcal{J}}$, the number $\mu_j \in \mathbb{R}$ realizing the minimum

$$\rho_j = \min_{\mu \in \mathbb{R}} \|r + \mu A_{k-1}e_j\|$$

is the real number such that the derivate of the quadratic function

$$f_j(\mu) = \|r + \mu A_{k-1}e_j\|^2 = \|r\|^2 + 2\mu(r|A_{k-1}e_j) + \mu^2\|A_{k-1}e_j\|^2$$

vanishes, i.e.

$$\mu_j = -\frac{(r|A_{k-1}e_j)}{\|A_{k-1}e_j\|^2}.$$

So

$$\begin{aligned}\rho_j^2 = f_j(\mu_j) &= \|r\|^2 - 2\frac{(r|A_{k-1}e_j)^2}{\|A_{k-1}e_j\|^2} + \frac{(r|A_{k-1}e_j)^2}{\|A_{k-1}e_j\|^2} \\ &= \|r\|^2 - \frac{(r|A_{k-1}e_j)^2}{\|A_{k-1}e_j\|^2}.\end{aligned}$$

Associate with each $j \in \widetilde{\mathcal{J}}$ the weight

$$\omega_j = \frac{(r|A_{k-1}e_j)^2}{\|A_{k-1}e_j\|^2}.$$

The greater the weight ω_j the more 'efficient' the index j. Therefore, we choose s indices in $\widetilde{\mathcal{J}}$ among those of largest weight and we add them to $\mathcal{J}$; if $|\widetilde{\mathcal{J}}| \leqslant s$, we select $\widetilde{\mathcal{J}}$ entirely.

Using the two disjoint sets $\mathcal{J} = \{j_1, \ldots, j_p\}$ and $\widetilde{\mathcal{J}} = \{\tilde{j}_1, \ldots, \tilde{j}_s\}$, let us show how to proceed to compute the vector $\tilde{z}_k$ realizing the minimum

$$\min_{\substack{u \in \mathbb{R}^{k-1} \\ u_i = 0,\, i \notin \mathcal{J} \sqcup \widetilde{\mathcal{J}}}} \|A_{k-1}u + \tilde{a}_k\|.$$

Let $y = (Z_{j_1 k}, \ldots, Z_{j_p k}, Z_{\tilde{j}_1 k}, \ldots, Z_{\tilde{j}_s k})^t$. In order to use (16), we compute the QR decomposition $\widehat{A} = \widehat{Q}\widehat{R}$ of the matrix

$$\widehat{A} = (A_{k-1}e_{j_1}, \ldots, A_{k-1}e_{j_p}, A_{k-1}e_{\tilde{j}_1}, \ldots, A_{k-1}e_{\tilde{j}_s})$$

from the decomposition $\widetilde{A} = QR$ of $\widetilde{A} = (A_{k-1}e_{j_1}, \ldots, A_{k-1}e_{j_p})$, known from the previous step. Since we have (using block notation)

$$\widehat{A} = (\widetilde{A}, \widehat{A}_{12}) = (Q, \widehat{Q}_{12}) \begin{pmatrix} R & \widehat{R}_{12} \\ 0 & \widehat{R}_{22} \end{pmatrix},$$

it suffices to extend the decomposition $\widetilde{A} = QR$, i.e. to compute only the last s columns of $\widehat{Q}$ and $\widehat{R}$. Note that $j_1, \ldots, j_p, \tilde{j}_1, \ldots, \tilde{j}_s$ is not necessary in increasing order.

5. Theoretical results

The preconditioner $T = D^{-1/2}Z^t$ constructed in one of the previous methods provides the preconditioned system whose matrix is

$$S = TAT^t = D^{-1/2} \cdot Z^t \cdot A \cdot Z \cdot D^{-1/2}. \tag{20}$$

Consider the matrix Z described in section 4 with optimal filling (in section 4.1); some theoretical properties concerning the matrix S are given; in particular, some upper bounds for its condition number.

THEOREM 5.1 *For $k = 2, \ldots, n$, let m_k be the minimum of equation* (17) *realized by the vector formed by the first $k-1$ components of the kth column of Z. Let $\varepsilon > 0$. Assume that $m_k \leqslant \varepsilon$ for $k = 2, \ldots, n$ and let $\lambda_{\min} = \lambda_{\min}(A)$ be the smallest eigenvalue of the matrix A; we have*

$$|s_{ik}| \leqslant \frac{\varepsilon}{\lambda_{\min}}, \quad \text{if } i \neq k,$$
$$s_{kk} = 1, \quad 1 \leqslant k \leqslant n,$$

and

$$\|S - I\|_F,\ \|S - I\|_2 \leqslant \frac{\sqrt{n(n-1)} \cdot \varepsilon}{\lambda_{\min}},$$
$$\|S - I\|_1 \leqslant \frac{(n-1)\varepsilon}{\lambda_{\min}},$$

where $\|X\|_F = (\mathrm{Tr}(XX^t))^{1/2} = (\mathrm{Tr}(X^t \cdot X))^{1/2}$ is the Frobenius norm of X and $\|X\|_i = \sup_{\|y\|_i = 1} \|X \cdot y\|_i$ is the operator norm of X derived from the norm $\|.\|_i$ in $\mathbb{R}^n$, $i = 1, 2$.

Proof When the norm is not explicitly precise, the usual Euclidean norm in $\mathbb{R}^n$ is used. According to (20) we have

$$s_{ik} = \frac{(z_i|Az_k)}{\|z_i\|_A \|z_k\|_A}.$$

It is clear that $s_{kk} = 1$ for all k. Since S is symmetric, it suffices to show the inequality $|s_{ik}| \leqslant \varepsilon/\lambda_{\min}$ for $i < k$. Fix i, k, with $i < k$, and for $x \in \mathbb{R}^n$, let $\tilde{x}$ be the vector in $\mathbb{R}^{k-1}$ formed with the first $k-1$ components of x. Since Z is an upper triangular and unitary diagonal matrix, we have

$$(z_i|Az_k) = (\tilde{z}_i|A_{k-1}\tilde{z}_k + \tilde{a}_k).$$

Since $m_k = \|A_{k-1}\tilde{z}_k + \tilde{a}_k\| \leqslant \varepsilon$, using the Cauchy–Schwartz inequality we obtain

$$|(z_i|Az_k)| \leqslant \|\tilde{z}_i\| \cdot \varepsilon = \|z_i\| \cdot \varepsilon. \tag{21}$$

The Rayleigh quotient for the matrix A defined by $\mu(x) = (x|Ax)/(x|x)$ for $x \neq 0$ satisfies $\lambda_{\min} = \min_{x\neq 0} \mu(x)$ and $\lambda_{\max} = \max_{x\neq 0} \mu(x)$ (see, for example, [3, pp. 24–25]). Hence,

$$\frac{\|z_i\|}{\|z_i\|_A} = \frac{1}{\sqrt{\mu(z_i)}} \leqslant \frac{1}{\sqrt{\lambda_{\min}}}, \tag{22}$$

and

$$\|z_k\|_A = \sqrt{\mu(z_k)} \cdot \|z_k\| \geqslant \sqrt{\lambda_{\min}} \cdot \|z_k\|. \tag{23}$$

With the estimates (21), (22) and (23), we obtain

$$|s_{ik}| = \frac{|(z_i|Az_k)|}{\|z_i\|_A \|z_k\|_A} \leqslant \frac{\|z_i\| \cdot \varepsilon}{\|z_i\|_A \cdot \|z_k\|_A} \leqslant \frac{\varepsilon}{\lambda_{\min} \cdot \|z_k\|}.$$

Since $\|z_k\| \geqslant 1$, the first part of the theorem is proved.

Let e_k be the kth vector of the usual basis in $\mathbb{R}^n$. Using the previous results,

$$\|(S-I)e_k\|_2^2 = \sum_{i=1}^{n} (s_{ik} - \delta_{ik})^2 = \sum_{i\neq k} (s_{ik})^2 \leqslant (n-1)\frac{\varepsilon^2}{\lambda_{\min}^2},$$

and

$$\|(S-I)e_k\|_1 = \sum_{i\neq k} |s_{ik}| \leqslant (n-1)\frac{\varepsilon}{\lambda_{\min}}.$$

Therefore, for the Frobenuis norm, we obtain

$$\|S-I\|_F^2 = \mathrm{Tr}((S-I)^t \cdot (S-I)) = \sum_{k=1}^{n} \|(S-I)e_k\|_2^2 \leqslant n(n-1)\frac{\varepsilon^2}{\lambda_{\min}^2}.$$

For $i = 1, 2$,

$$\|S-I\|_i = \sup_{\|x\|_i=1} \|(S-I)x\|_i = \sup_{\|x\|_i=1} \left\| \sum_{k=1}^{n} x_k (S-I)e_k \right\|_i$$

$$\leqslant \sup_{\|x\|_i=1} \sum_{k=1}^{n} |x_k| \cdot \|(S-I)e_k\|_i,$$

so

$$\|S - I\|_1 \leqslant \sup_{\|x\|_1=1} \sum_{k=1}^{n} |x_k| \cdot \|(S - I)e_k\|_1 \leqslant (n-1)\frac{\varepsilon}{\lambda_{\min}}.$$

Since $\|x\|_1 \leqslant \sqrt{n}\|x\|_2$ for all $x \in \mathbb{R}^n$ and $\sup_{\|x\|_2=1} \|x\|_1 = \sqrt{n}$, we obtain

$$\|S - I\|_2 \leqslant \sup_{\|x\|_2=1} \sum_{k=1}^{n} |x_k| \cdot \|(S - I)e_k\|_2 \leqslant \sqrt{(n-1)}\frac{\varepsilon}{\lambda_{\min}} \sup_{\|x\|_2=1} \|x\|_1$$
$$= \sqrt{n(n-1)}\frac{\varepsilon}{\lambda_{\min}}.$$ ■

COROLLARY 5.2 *With the assumptions of the previous theorem, if* $\delta = (n-1)\varepsilon/\lambda_{\min}(A)$, *we have*

$$|\lambda - 1| \leqslant \delta$$

for all eigenvalues λ *of* S. *In particular, if* $\delta < 1$, *the condition number* $\kappa(S)$ *of* S *verifies*

$$\kappa(S) \leqslant \frac{1+\delta}{1-\delta}.$$

Proof Let λ be an eigenvalue of S and v an eigenvector corresponding to λ with $\|v\|_1 = 1$. Then, from the previous theorem,

$$|\lambda - 1| = \|(\lambda - 1)v\|_1 = \|(S - I)v\|_1 \leqslant \|S - I\|_1 \cdot \|v\|_1 \leqslant \delta.$$

(Note that the same proof with the norm $\|.\|_2$ provides the inequality with $\delta = (n(n-1))^{1/2} \cdot \varepsilon/\lambda_{\min}(A)$, which is less precise.) ■

6. Variants

In this section we provide some variants for the preconditioning methodology. We first transform the system (1) with the diagonal preconditioner $T_1 = \operatorname{diag}(a_{11}^{-1/2}, \ldots, a_{nn}^{-1/2})$ and then apply any other preconditioner T_2 to the new system matrix $\widetilde{A} = T_1 A T_1^t$ (which has the same pattern of non-zero coefficients as A). We obtain the coupled preconditioner

$$T = T_2 T_1$$

of matrix A and the preconditioned system matrix is

$$T_2 \widetilde{A} T_2^t = T_2 T_1 A T_1^t T_2^t = TAT^t.$$

6.1 *Block treatment*

In this case a block decomposition of A is considered. Let $A_1, \ldots, A_M$ be the diagonal blocks, where A_i is a matrix of order m_i, $1 \leqslant i \leqslant M$, with $m_1 + \cdots + m_M = n$ (the order of A). For each block A_i, a preconditioner T_i is constructed. Then $T = \operatorname{diag}(T_1, \ldots, T_M)$ is a preconditioner of A.

Remark 3 Consider the preconditioner with optimal filling in section 4.1. For the computation of its kth column, the 'best' indices are selected among the $k-1$ positions above the diagonal. Thus, the larger k, the longer the computation of the kth column. With block treatment, the computation time is reduced and the provided preconditioner is still of good quality.

7. Numerical tests

The CG algorithm for the system $Ax = b$ is tested with the following preconditioners T:

(0) NONE;
(i) DIAG: diagonal preconditioner, $T = \text{diag}(a_{11}^{-1/2}, \ldots, a_{nn}^{-1/2})$;
(ii) INC CHO: incomplete Cholesky factorization (see Introduction and [3, 4]);
(iii) FSPAI: factorized sparse approximate inverse (see Introduction and [5]);
(iv) INC CGS: preconditioner provided by the INC CGS algorithm (see section 3) with $P = \{(i, j) \mid 1 \leqslant i \leqslant j \leqslant n,\ a_{ij} \neq 0\}$;
(v) LS CGS (A): preconditioner with A-filling in section 4.1;
(vi) LS CGS (DIAG): preconditioner with diagonal filling in section 4.1;
(vii) LS CGS (OPT): preconditioner with optimal filling in section 4.1;
(viii) DIAG+LS CGS (OPT): coupled preconditioner, diagonal, then LS CGS (OPT);
(ix) LS CGS (OPT) BLOCKS: preconditioner LS CGS (OPT) by blocks;
(x) DIAG+LS CGS (OPT) BLOCKS: preconditioner DIAG+LS CGS (OPT) by blocks.

For preconditioners LS CGS (OPT) and DIAG+LS CGS (OPT), the tolerance will be set to $\varepsilon \approx \lambda_{\min}/(n-1)$, so that δ in Corollary 5.2 is close to one. For preconditioner LS CGS (OPT), $\lambda_{\min} = \lambda_{\min}(A)$ and for preconditioner DIAG+LS CGS (OPT), $\lambda_{\min} = \lambda_{\min}(T_1 A T_1^t)$, with $T_1 = \text{diag}(a_{11}^{-1/2}, \ldots, a_{nn}^{-1/2})$. For preconditioner LS CGS (OPT) BLOCKS (resp. DIAG+LS CGS (OPT) BLOCKS), the same tolerance ε as for preconditioner LS CGS (OPT) (resp. DIAG+LS CGS (OPT)) is considered. When M blocks are considered, their order is set to $m_1 = \cdots = m_r = q + 1$ and $m_{r+1} = \cdots = m_M = q$, where $n = q \cdot M + r$ is the Euclidean division of n by M (q, r are integers with $0 \leqslant r < M$).

The convergence tolerance *tol* is set to 10^{-8} in the CG algorithm. We are interested in the number of iterations required to solve the system. When an error occurs during construction of the preconditioner, the notation 'EP' is used. If an error occurs during resolution, the notation 'ECG' is used. If the norm of the residual vector is larger than the tolerance *tol* after n iterations (n is the order of A), we write '$>n$'.

Estimates of the largest eigenvalue ($\lambda_{\max}$) and the smallest eigenvalue ($\lambda_{\min}$) are given; this allows us to obtain the condition number (κ) of the preconditioned system matrix TAT^t (see (2)).

Finally, the computation time for the construction of the preconditioner and for the resolution are presented. The tests are performed on an Intel® Pentium® 4 machine at 2.66 GHz.

The SPD test matrices shown in table 1 are considered; the order is n and the number of non-zero coefficients in the upper (or lower) part is *NNZ*. These matrices can be obtained from http://math.nist.gov/MatrixMarket/ and http://www.cise.ufl.edu/research/sparse/matrices/.

For each matrix, the results are presented in a table. The number of iterations is denoted It., the computation time (in seconds) for construction of the preconditioner is t_{prec}, the computation time (in seconds) for the resolution is t_{res} and the number of non-zero coefficients in the preconditioner (for the preconditioners LS CGS (OPT), DIAG+LS CGS (OPT), LS CGS

Table 1. SPD test matrices.

Matrix (A)	n	*NNZ*
nos1	237	627
bcsstk27	1224	28 675
s1rmt3m1	5489	112 505

(OPT) BLOCKS and DIAG+LS CGS (OPT) BLOCKS) is *nnz*. For preconditioners FSPAI, INC CGS and LS CGS (A), we have $nnz = NNZ$. (Matrix L of INC CHO also has $nnz = NNZ$ non-zero coefficients.)

7.1 *First test: matrix nos1*

We report the results for test matrix *nos1* in tables 2–9. Note that the computation times (t_{prec} and t_{res}) are not very significant, since the number of non-zero coefficients in the matrix is small.

These first tests show that, *in practice*, the CG algorithm does not necessarily converge. Moreover, they show the efficiency of preconditioners LS CGS (DIAG), LS CGS (OPT) and DIAG+LS CGS (OPT). They are the only ones that give convergence for this matrix. Note that, in the case of LS CGS (DIAG), for a small value of $p_{\max}$, the CG algorithm does not converge; the same is true when $p_{\max}$ is too large. The reason for this is probably due to rounding errors.

Now, fix the parameter $p_{\max}$ and take various values for the parameter s in preconditioners LS CGS (OPT) and DIAG+LS CGS (OPT). The results are given in tables 6 and 7.

Table 2. Results for test matrix *nos1*.

Preconditioner	It.	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
NONE	>237	0.00E + 00	1.00*E* − 02	1.23E + 02	2.46E + 09	1.99E + 07
DIAG	>237	0.00E + 00	1.00*E* − 02	5.09*E* − 07	2.00E + 00	3.93E + 06
INC CHO	EGC					
FSPAI	>237	0.00E + 00	2.00*E* − 02	1.92*E* − 06	1.69E + 00	8.79E + 05
INC CGS	>237	0.00E + 00	0.00E + 00	?	?	?
LS CGS (A)	>237	0.00E + 00	3.00*E* − 02	1.65*E* − 06	2.63E + 00	1.60E + 06

Table 3. Preconditioner LS CGS (DIAG).

Preconditioner: LS CGS (DIAG)							
$p_{\max}$	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	>237	473	0.00E + 00	4.00E − 02	5.09E − 07	2.08E + 00	4.08E + 06
5	>237	1407	1.00E − 02	3.00E − 02	2.16E − 06	3.05E + 00	1.41E + 06
10	115	2252	1.00E − 02	2.00E − 02	2.83E − 05	2.05E + 00	7.25E + 04
20	58	4767	3.00E − 02	1.00E − 02	2.04E − 04	1.89E + 00	9.25E + 03
50	25	10 812	1.10E − 01	0.00E + 00	4.74E − 03	1.64E + 00	3.46E + 02
100	145	18 887	5.70E − 01	9.00E − 02	3.17E − 07	1.28E + 01	4.06E + 07
200	>237	27 537	1.84E + 00	1.60E − 01	?	1.84E + 01	?

Table 4. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with ε = 5.23E-01, $s = 1$							
$p_{\max}$	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	>237	471	1.00E − 02	3.00E − 02	1.31E − 06	2.65E + 00	2.02E + 06
5	>237	1391	3.00E − 02	3.00E − 02	2.15E − 06	4.25E + 00	1.97E + 06
10	>237	2496	5.00E − 02	3.00E − 02	3.05E − 06	2.69E + 00	8.80E + 05
20	179	4556	1.30E − 01	3.00E − 02	4.43E − 06	2.47E + 00	5.57E + 05
50	44	9536	7.40E − 01	1.00E − 02	1.20E − 04	2.21E + 00	1.84E + 04
100	14	14 067	2.20E + 00	1.00E − 02	1.75E − 02	1.73E + 00	9.85E + 01
200	2	15 720	3.20E + 00	0.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00

Table 5. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 2.16\text{E-}09$, $s = 1$							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	>237	471	1.00E − 02	1.00E − 02	1.20E − 06	2.02E + 00	1.67E + 06
5	151	1391	2.00E − 02	1.00E − 02	9.25E − 06	1.79E + 00	1.93E + 05
10	112	2496	4.00E − 02	1.00E − 02	1.30E − 05	1.80E + 00	1.39E + 05
20	98	4556	1.00E − 01	1.00E − 02	1.33E − 05	2.01E + 00	1.51E + 05
50	33	9536	5.00E − 01	1.00E − 02	1.61E − 04	1.88E + 00	1.17E + 04
100	15	14 067	1.40E + 00	1.00E − 02	5.30E − 03	1.72E + 00	3.25E + 02
200	2	15 720	2.17E + 00	0.00E + 00	1.00E + 00	1.00E + 00	1.00E + 00

The reason why the construction of the preconditioner fails (EP) is that, during the computation of a column, the set $\widetilde{\mathcal{J}}$ (see (19)) is empty. Increasing the parameter ε allows us to compute the preconditioner entirely.

We report the results for preconditioners LS CGS (OPT) and DIAG+LS CGS (OPT) with block computation in tables 8 and 9.

7.2 *Second test: matrix bcsstk27*

The results for test matrix *bcsstk27* are presented in tables 10–17. Preconditioner INC CGS has $nnz = 28\,675$ non-zero coefficients and 99 iterations (see table 10) are needed. With preconditioner DIAG+LS CGS (OPT), $p_{\max} = 20$ (table 13), the filling is $nnz = 25\,494$ and 70 iterations are necessary to obtain convergence. Therefore, the latter method gives the best results for this kind of preconditioner.

With the same parameters $p_{\max}$, ε and s in the preconditioner LS CGS (OPT) (resp. DIAG+LS CGS (OPT)), when the numbers of blocks is increased, the number of iterations also increases (see tables 12 and 16 (resp. 13 and 17)).

Table 6. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with $\varepsilon = 5.23\text{E-}01$, $p_{\max} = 50$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	37	9536	4.60E − 01	0.00E + 00	2.40E − 04	2.09E + 00	8.68E + 03
3	>237	9643	3.10E − 01	5.00E − 02	?	4.08E + 00	?
4	EP						
5	>237	9802	1.80E − 01	6.00E − 02	?	6.26E + 00	?
50	>237	9802	1.70E − 01	6.00E − 02	?	6.26E + 00	?

Table 7. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 2.16\text{E-}09$, $p_{\max} = 50$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	25	9536	2.60E − 01	1.00E − 02	8.75E − 05	1.91E + 00	2.19E + 04
3	147	9643	2.30E − 01	4.00E − 02	2.44E − 06	2.54E + 00	1.04E + 06
4	EP						
5	EP						
50	EP						

Table 8. Preconditioner LS CGS (OPT) BLOCKS.

Preconditioner: LS CGS (OPT) BLOCKS with $\varepsilon = 5.23\text{E}-01$, $p_{\max} = 10$, $s = 1$							
Blocks	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	>237	2386	4.00E − 02	3.00E − 02	3.07E − 06	2.68E + 00	8.75E + 05
3	>237	2275	3.00E − 02	4.00E − 02	3.12E − 06	2.67E + 00	8.58E + 05
4	>237	2164	2.00E − 02	2.00E − 02	3.14E − 06	2.65E + 00	8.46E + 05
6	216	1942	2.00E − 02	2.00E − 02	3.24E − 06	2.52E + 00	7.78E + 05
8	191	1721	2.00E − 02	1.00E − 02	3.44E − 06	2.30E + 00	6.68E + 05
16	179	1080	0.00E + 00	2.00E − 02	2.54E − 06	2.00E + 00	7.87E + 05
24	>237	757	1.00E − 02	1.00E − 02	1.70E − 06	2.00E + 00	1.18E + 06
32	>237	594	1.00E − 02	1.00E − 02	1.24E − 06	2.00E + 00	1.61E + 06

Table 9. Preconditioner DIAG+LS CGS (OPT) BLOCKS.

Preconditioner: DIAG+LS CGS (OPT) BLOCKS with $\varepsilon = 2.16\text{E}-09$, $p_{\max} = 10$, $s = 1$							
Blocks	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	120	2386	3.00E − 02	2.00E − 02	7.95E − 06	2.00E + 00	2.52E + 05
3	130	2275	2.00E − 02	1.00E − 02	7.34E − 06	2.04E + 00	2.78E + 05
4	136	2164	2.00E − 02	1.00E − 02	6.43E − 06	2.04E + 00	3.17E + 05
6	149	1942	1.00E − 02	1.00E − 02	5.21E − 06	2.04E + 00	3.92E + 05
8	158	1721	1.00E − 02	2.00E − 02	4.31E − 06	2.04E + 00	4.73E + 05
16	184	1080	1.00E − 02	1.00E − 02	2.54E − 06	2.00E + 00	7.87E + 05
24	>237	757	0.00E + 00	3.00E − 02	1.70E − 06	2.00E + 00	1.18E + 06
32	>237	594	0.00E + 00	3.00E − 02	1.24E − 06	2.00E + 00	1.61E + 06

Table 10. Results for test matrix *bcsstk27*.

Preconditioner	It.	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
NONE	1190	0.00E + 00	5.50E − 01	1.44E + 02	3.47E + 06	2.41E + 04
DIAG	253	0.00E + 00	1.40E − 01	2.12E − 03	4.37E + 00	2.06E + 03
INC CHO	24	5.00E − 02	4.00E − 02			
FSPAI	89	7.00E − 02	1.10E − 01	5.31E − 03	1.71E + 00	3.22E + 02
INC CGS	99	2.00E − 02	1.10E − 01	9.24E − 03	3.12E + 00	3.38E + 02
LS CGS (A)	213	6.23E + 00	2.80E − 01	1.67E − 03	3.17E + 00	1.90E + 03

Table 11. Preconditioner LS CGS (DIAG).

Preconditioner: LS CGS (DIAG)							
$p_{\max}$	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	343	2447	2.20E − 01	2.60E − 01	1.18E − 03	4.65E + 00	3.93E + 03
5	331	7329	3.60E − 01	2.10E − 01	9.90E − 04	3.90E + 00	3.94E + 03
10	401	13 409	8.10E − 01	3.20E − 01	4.60E − 04	3.37E + 00	7.33E + 03
20	439	25 494	2.71E + 00	5.20E − 01	4.81E − 04	3.98E + 00	8.27E + 03
50	343	61 149	3.13E + 01	7.00E − 01	1.00E − 03	5.13E + 00	5.10E + 03
100	179	118 574	2.10E + 02	8.60E − 01	3.61E − 03	4.47E + 00	1.24E + 03

Table 12. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with $\varepsilon = 1.17\text{E}-01$, $s = 1$							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	144	2447	1.16E + 00	8.00E − 02	4.19E − 03	2.47E + 00	5.89E + 02
5	119	7329	8.74E + 00	9.00E − 02	6.19E − 03	2.62E + 00	4.23E + 02
10	105	13 409	2.64E + 01	7.00E − 02	6.90E − 03	2.52E + 00	3.65E + 02
20	93	25 487	8.60E + 01	1.00E − 01	8.22E − 03	2.36E + 00	2.87E + 02
50	70	61 139	5.20E + 02	1.30E − 01	1.27E − 02	2.01E + 00	1.58E + 02
100	57	118 549	1.87E + 03	1.90E − 01	2.09E − 02	1.95E + 00	9.32E + 01

Table 13. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 1.73\text{E}-06$, $s = 1$							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	139	2447	1.18E + 00	8.00E − 02	4.56E − 03	2.46E + 00	5.39E + 02
5	103	7329	7.69E + 00	7.00E − 02	5.06E − 03	1.89E + 00	3.73E + 02
10	84	13 409	2.16E + 01	9.00E − 02	7.46E − 03	1.69E + 00	2.26E + 02
20	70	25 494	6.47E + 01	9.00E − 02	1.15E − 02	1.69E + 00	1.47E + 02
50	52	61 149	3.74E + 02	1.20E − 01	2.00E − 02	1.71E + 00	8.57E + 01
100	40	118 570	1.49E + 03	1.40E − 01	3.01E − 02	1.56E + 00	5.18E + 01

Table 14. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with $\varepsilon = 1.17\text{E}-01$, $p_{\max} = 50$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	75	61 145	2.31E + 02	1.40E − 01	1.24E − 02	2.03E + 00	1.63E + 02
3	>1224	62 319	1.97E + 02	2.53E + 00	?	7.08E + 00	?
4	>1224	63 492	1.61E + 02	2.38E + 00	?	6.13E + 00	?
5	76	61 146	1.19E + 02	1.50E − 01	1.16E − 02	2.08E + 00	1.80E + 02
10	77	61 146	8.18E + 01	1.50E − 01	1.06E − 02	2.23E + 00	2.11E + 02
25	EP						
50	EP						

Table 15. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 1.73\text{E}-06$, $p_{\max} = 50$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	53	61 149	1.76E + 02	8.00E − 02	1.80E − 02	1.62E + 00	8.96E + 01
3	119	62 322	1.31E + 02	2.30E − 01	1.25E − 02	3.58E + 00	2.86E + 02
4	450	63 494	1.12E + 02	8.70E − 01	9.29E − 10	3.53E + 00	3.80E + 09
5	52	61 149	1.01E + 02	1.10E − 01	1.69E − 02	1.46E + 00	8.63E + 01
10	52	61 149	6.93E + 01	1.00E − 01	1.54E − 02	1.42E + 00	9.22E + 01
25	58	61 196	5.34E + 01	1.10E − 01	1.34E − 02	1.87E + 00	1.39E + 02
50	EP						

Table 16. Preconditioner LS CGS (OPT) BLOCKS.

Preconditioner: LS CGS (OPT) BLOCKS with $\varepsilon = 1.17\text{E}-01$, $p_{\max} = 20$, $s = 1$							
Blocks	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	95	25 128	7.60E + 01	8.00E − 02	7.81E − 03	2.36E + 00	3.02E + 02
4	99	24 409	5.72E + 01	1.00E − 01	7.21E − 03	2.36E + 00	3.27E + 02
8	104	23 554	2.78E + 01	1.30E − 01	5.64E − 03	2.30E + 00	4.08E + 02
16	126	21 251	7.47E + 00	1.40E − 01	3.79E − 03	2.10E + 00	5.54E + 02
32	168	16 849	1.46E + 00	1.40E − 01	1.95E − 03	2.19E + 00	1.12E + 03
64	192	10 151	2.70E − 01	1.40E − 01	1.81E − 03	2.47E + 00	1.37E + 03
128	200	5907	4.00E − 02	1.30E − 01	1.95E − 03	2.64E + 00	1.35E + 03

Table 17. Preconditioner DIAG+LS CGS (OPT) BLOCKS.

Preconditioner: DIAG+LS CGS (OPT) BLOCKS with $\varepsilon = 1.73\text{E}-06$, $p_{\max} = 20$, $s = 1$							
Blocks	It.	*nnz*	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	76	25 135	5.70E + 01	7.00E − 02	1.04E − 02	1.92E + 00	1.85E + 02
4	85	24 416	4.75E + 01	8.00E − 02	8.93E − 03	1.98E + 00	2.22E + 02
8	96	23 572	2.43E + 01	1.30E − 01	6.47E − 03	2.08E + 00	3.22E + 02
16	125	21 277	6.84E + 00	1.50E − 01	3.97E − 03	2.04E + 00	5.14E + 02
32	168	16 876	1.43E + 00	1.30E − 01	1.95E − 03	2.18E + 00	1.12E + 03
64	192	10 187	2.70E − 01	1.40E − 01	1.81E − 03	2.47E + 00	1.37E + 03
128	200	5914	7.00E − 02	1.50E − 01	1.95E − 03	2.64E + 00	1.35E + 03

Moreover, comparing table 16 with table 17, we see that the first diagonal preconditioner (table 17) has few effects when the number of blocks is sufficiently high.

7.3 *Third test: matrix s1rmt3m1*

Finally, we present the results for test matrix *s1rmt3m1* in tables 18–25. Once again for this matrix, preconditioner DIAG+LS CGS (OPT) with $p_{\max} = 10$ (table 21) gives better results than preconditioner INC CGS (table 18): 309 iterations with $nnz = 60\,323$ for the first method and 338 iterations with $nnz = 112\,505$ for the second method.

We remark that, in the last two tables, on increasing the number of blocks, there is considerable time saving in the construction of the preconditioner.

7.4 *Observations*

In general, as predicted by Theorem 1.1, we observe that the smaller the condition number of the preconditioned system matrix, the fewer the number of iterations.

Table 18. Results for test matrix *s1rmt3m1*.

Preconditioner	It.	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
NONE	>5489	0.00E + 00	1.07E + 01	3.80E − 01	9.67E + 05	2.55E + 06
DIAG	926	0.00E + 00	1.80E + 00	6.87E − 06	3.65E + 00	5.31E + 05
INC CHO	225	1.30E − 01	1.33E + 00			
FSPAI	329	2.10E − 01	1.61E + 00	2.34E − 05	1.76E + 00	7.52E + 04
INC CGS	338	1.20E − 01	1.66E + 00	3.20E − 05	2.66E + 00	8.33E + 04
LS CGS (A)	1380	2.23E + 01	6.65E + 00	3.22E − 06	5.54E + 00	1.72E + 06

Table 19. Preconditioner LS CGS (DIAG).

Preconditioner: LS CGS (DIAG)							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	902	10 977	4.50E + 00	2.31E + 00	6.33E − 06	3.57E + 00	5.64E + 05
5	967	32 919	5.98E + 00	2.84E + 00	6.44E − 06	3.76E + 00	5.84E + 05
10	1052	60 324	8.91E + 00	3.82E + 00	6.35E − 06	4.57E + 00	7.20E + 05
20	1280	115 059	1.88E + 01	6.15E + 00	5.29E − 06	5.34E + 00	1.01E + 06
50	2436	278 664	8.23E + 01	2.04E + 01	4.62E − 06	1.59E + 01	3.45E + 06

Table 20. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with $\varepsilon = 6.92\text{E} - 05$, $s = 1$							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	826	10 976	1.49E + 01	2.11E + 00	1.01E − 05	4.45E + 00	4.38E + 05
5	1221	32 918	5.63E + 01	3.62E + 00	3.75E − 06	5.46E + 00	1.45E + 06
10	1312	60 323	1.41E + 02	4.80E + 00	4.11E − 06	6.33E + 00	1.54E + 06
20	1010	115 058	4.43E + 02	4.90E + 00	6.36E − 06	5.28E + 00	8.31E + 05
50	663	278 663	2.66E + 03	5.61E + 00	1.38E − 05	4.23E + 00	3.06E + 05

Table 21. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 1.25\text{E} - 09$, $s = 1$							
$p_{\max}$	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
1	596	10 976	1.50E + 01	1.50E + 00	1.11E − 05	2.72E + 00	2.44E + 05
5	372	32 918	5.55E + 01	1.07E + 00	1.75E − 05	1.82E + 00	1.04E + 05
10	309	60 323	1.37E + 02	1.14E + 00	2.55E − 05	1.83E + 00	7.17E + 04
20	243	115 058	4.38E + 02	1.21E + 00	3.72E − 05	1.69E + 00	4.55E + 04
50	178	278 663	2.33E + 03	1.51E + 00	6.56E − 05	1.50E + 00	2.28E + 04

Table 22. Preconditioner LS CGS (OPT).

Preconditioner: LS CGS (OPT) with $\varepsilon = 6.92\text{E} - 05$, $p_{\max} = 20$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	981	115 058	2.07E + 02	4.97E + 00	7.41E − 06	5.85E + 00	7.89E + 05
3	>5489	120 526	1.53E + 02	2.80E + 01	?	8.40E + 00	?
4	980	115 058	1.11E + 02	4.93E + 00	6.66E − 06	5.20E + 00	7.82E + 05
5	931	115 058	9.29E + 01	4.66E + 00	7.25E − 06	4.81E + 00	6.64E + 05
10	1396	115 067	5.39E + 01	6.88E + 00	?	4.89E + 00	?
20	EP						

Preconditioners LS CGS (OPT) and DIAG+LS CGS (OPT) are interesting because their parameters provide some flexibility and allow us to take into account different demands during construction. The second preconditioner appears to be better and more stable. The computation time for these preconditioners is long; however, they can easily be parellelized and thus the CPU time can be significantly reduced (see Remark 3). When increasing the additional filling s, the preconditioners can be of poor quality. The block methodology is the best way to save computer time. However, when increasing the number of blocks, the number of iterations also increases. This is compensated for by the computation of the non-zero coefficients in the preconditioner (nnz decreases). Therefore, the time spent on resolution is rather stable.

Table 23. Preconditioner DIAG+LS CGS (OPT).

Preconditioner: DIAG+LS CGS (OPT) with $\varepsilon = 1.25\text{E}-09$, $p_{\max} = 20$							
s	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	248	115 058	1.97E + 02	1.23E + 00	3.60E − 05	1.72E + 00	4.78E + 04
3	331	120 526	1.47E + 02	1.75E + 00	3.08E − 05	2.61E + 00	8.50E + 04
4	267	115 058	1.04E + 02	1.31E + 00	3.34E − 05	1.80E + 00	5.39E + 04
5	279	115 058	8.89E + 01	1.38E + 00	3.20E − 05	1.83E + 00	5.71E + 04
10	292	115 067	4.93E + 01	1.44E + 00	2.91E − 05	1.80E + 00	6.17E + 04
20	350	117 200	3.70E + 01	1.83E + 00	2.55E − 05	2.24E + 00	8.76E + 04

Table 24. Preconditioner LS CGS (OPT) BLOCKS.

Preconditioner: LS CGS (OPT) BLOCKS with $\varepsilon = 6.92\text{E}-05$, $p_{\max} = 50$, $s = 1$							
Blocks	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	662	276 128	2.42E + 03	5.54E + 00	1.37E − 05	4.11E + 00	3.01E + 05
4	638	271 406	1.69E + 03	5.32E + 00	1.35E − 05	3.92E + 00	2.91E + 05
8	577	262 217	6.42E + 02	4.62E + 00	1.59E − 05	3.23E + 00	2.03E + 05
16	452	243 731	2.10E + 02	3.47E + 00	1.94E − 05	2.48E + 00	1.28E + 05
32	574	207 705	6.96E + 01	3.94E + 00	1.03E − 05	2.22E + 00	2.17E + 05
64	586	174 468	3.93E + 01	3.56E + 00	9.89E − 06	2.24E + 00	2.27E + 05
128	594	111 071	1.30E + 01	2.66E + 00	9.56E − 06	2.24E + 00	2.35E + 05
256	630	59 269	2.16E + 00	2.30E + 00	8.93E − 06	2.35E + 00	2.63E + 05
512	666	31 706	3.20E − 01	1.89E + 00	7.91E − 06	2.38E + 00	3.01E + 05

Table 25. Preconditioner DIAG+LS CGS (OPT) BLOCKS.

Preconditioner: DIAG+LS CGS (OPT) BLOCKS with $\varepsilon = 1.25\text{E}-09$, $p_{\max} = 50$, $s = 1$							
Blocks	It.	nnz	t_{prec}	t_{res}	$\lambda_{\min}$	$\lambda_{\max}$	κ
2	203	276 128	2.09E + 03	1.72E + 00	5.86E − 05	1.84E + 00	3.14E + 04
4	235	271 406	1.45E + 03	1.94E + 00	4.83E − 05	1.86E + 00	3.85E + 04
8	286	262 220	5.62E + 02	2.32E + 00	3.63E − 05	1.97E + 00	5.44E + 04
16	398	243 736	1.52E + 02	3.09E + 00	2.00E − 05	2.03E + 00	1.01E + 05
32	573	207 762	4.34E + 01	3.98E + 00	1.03E − 05	2.22E + 00	2.17E + 05
64	586	174 664	2.85E + 01	3.58E + 00	9.89E − 06	2.24E + 00	2.27E + 05
128	594	111 121	1.09E + 01	2.70E + 00	9.56E − 06	2.24E + 00	2.35E + 05
256	630	59 269	2.08E + 00	2.21E + 00	8.93E − 06	2.35E + 00	2.63E + 05
512	667	31 706	3.40E − 01	1.91E + 00	7.91E − 06	2.38E + 00	3.01E + 05

On the other hand, linear SPD systems are used for the numerical solution of many non-stationary partial differential equations. In this situation the solution of a linear system is required for each time step. Thus when the matrix is time independent, an efficient preconditioner allows us to make a significant saving with respect to computer time. When the number of time steps is large, the computer time for the construction of the preconditioner can be neglected.

The description, analysis and implementation of the presented algorithms on parallel computers will be presented in a forthcoming paper.

References

[1] Stoer, J. and Bulirsch, R., 1980, *Introduction to Numerical Analysis* (Berlin: Springer).
[2] Axelsson, O., 1994, *Iterative Solution Methods* (Cambridge: Cambridge University Press).
[3] Saad, Y., 1996, *Iterative Methods for Sparse Linear Systems* (Boston, MA: PWS Publishing Company).

[4] Meijerink, J.A. and van der Vorst, H.A., 1977, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, **31**, 148–162.
[5] Kolotilina, L.Yu. and Yeremin, A.Yu., 1993, Factorized sparse approximate inverse preconditionings I. theory. *SIAM Journal on Matrix Analysis and Applications*, **14**, 45–58.
[6] Benzi, M., 2002, Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, **182**, 418–477.
[7] Benzi, M., Meyer, C.D. and Tuma, M., 1996, A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, **17**, 1135–1149.
[8] Strang, G., 1976, *Linear Algebra and its Applications* (New York: Academic Press).
[9] Grote, M. and Huckle, T., 1997, Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, **18**, 838–853.
[10] Anton, H. and Rorres, C., 1987, *Elementary Linear Algebra with Applications* (New York: Wiley).