NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl., **6,** 213–234 (1999)

Subspace-by-subspace Preconditioners for Structured Linear Systems

Michel J. Daydé[1]*, Jérôme P. Décamps[1] and Nicholas I. M. Gould[2]

[1]*ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse Cedex, France*
[2]*Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, UK*

We consider the iterative solution of symmetric positive-definite linear systems whose coefficient matrix may be expressed as the outer product of low-rank terms. We derive suitable preconditioners for such systems, and demonstrate their effectiveness on a number of test examples. We also consider combining these methods with existing techniques to cope with the commonly-occuring case where the coefficient matrix is the linear sum of elements, some of which are of very low rank. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS Sparse structured linear systems, iterative methods, preconditioned conjugate gradient, element-by-element preconditioners

1. Introduction

We consider the solution of n by n real linear systems of equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.1}$$

where $\boldsymbol{A}$ is symmetric positive-definite and has the form

$$\boldsymbol{A} = \sum_{i=1}^{e} \boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}} \tag{1.2}$$

Here $\boldsymbol{A}_i$ is an n by n_i real matrix, and e is a positive integer. Systems of this form arise naturally in a number of ways.

1. Normal equations for least squares (see, for instance, [1]).

* Correspondence to M. J. Daydé, ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse Cedex, France.

CCC 1070–5325/99/030213–22 $17.50

Received 26 January 1998
Revised 18 February 1999

2. The Schur complement following partial elimination in augmented systems (see, for example, [8]).
3. Newton equations for partially separable optimization of unary functions (see [12]).
4. More general partially separable optimization (see [13]).

We shall assume that n is sufficiently large that the structure of the system must be exploited, but we do *not* assume that all the $\boldsymbol{A}_i$ are sparse.

We aim to solve (1.1) using an iterative method, and, given the symmetry and definiteness of $\boldsymbol{A}$, the method of preconditioned conjugate gradients (see [3,14]) is the natural choice.

The purpose of this paper is to describe a new class of preconditioners which reflect the structure (1.2) of $\boldsymbol{A}$, and which are especially efficient when the matrices $\boldsymbol{A}_i$ are of low rank, without necessarily being sparse. An extreme case would be when a_i is a dense vector and $\boldsymbol{A}_i = a_i$, which results in a full but rank-one matrix, $a_i a_i^{\mathrm{T}}$. In this case, most traditional preconditioners would prove to be most ineffective. We do not wish to assemble the whole of $\boldsymbol{A}$, but prefer to use the components $\boldsymbol{A}_i$ in isolation. This will enable us to construct preconditioners which are appropriate for parallel computation.

In Section 2 we introduce our subspace-by-subspace (SBS) preconditioners which are a special type of element-by-element (EBE) preconditioner designed to deal with matrices of the form (1.2) and other low-rank matrices. In the following section, we apply these methods to least-squares problems, and demonstrate their effectiveness.

Quite clearly, SBS preconditioners will never be well suited to all problems, most particularly to those problems with one or more matrix $\boldsymbol{A}_i$ of (close to) full rank. In Section 4 we consider matrices for which some, but not all, terms are of the form $\boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}}$. We show that one of the great advantage of the SBS preconditioners is that they can be efficiently combined with other element-by-element preconditioners to handle substructures of low rank. This gives rise to composite preconditioners that are effective on a wide range of matrices.

2. Development

In Section 2.1, we consider the basic ideas behind element-by-element preconditioning. This is followed, in Section 2.2, by a description of our new class of preconditioners.

2.1. Element-by-element preconditioners

An obvious approach to finding a suitable preconditioner for (1.2) is to let $\boldsymbol{E}_i = \boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}}$, in which case (1.2) becomes

$$\boldsymbol{A} = \sum_{i=1}^{e} \boldsymbol{E}_i \tag{2.1}$$

Notice here that each *element* $\boldsymbol{E}_i$ is positive semi-definite. In this section and the next, we shall consider the general form (2.1) without necessarily assuming that $\boldsymbol{E}_i = \boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}}$. We shall return to this particular form in Section 2.3.

A popular class of preconditioners for systems whose coefficient matrix has the form (2.1) are the *element-by-element* preconditioners (see, [16,19]). These have been seen to be effective for systems arising from partial differential equations ([10,15]) and optimization [5].

There are four fundamental ingredients involved in the construction of such a preconditioner. We rearrange (2.1) to give

$$\boldsymbol{A} = \sum_{i=1}^{e} \boldsymbol{D}_i + \sum_{i=1}^{e} (\boldsymbol{E}_i - \boldsymbol{D}_i) = \boldsymbol{D} + \sum_{i=1}^{e} (\boldsymbol{E}_i - \boldsymbol{D}_i) \tag{2.2}$$

where $\boldsymbol{D}_i = \Delta(\boldsymbol{E}_i)$, $\boldsymbol{D} = \sum_{i=1}^{e} \boldsymbol{D}_i = \Delta(\boldsymbol{A})$ and $\Delta(\boldsymbol{M})$ denotes the diagonal matrix comprising the diagonal of the matrix $\boldsymbol{M}$. Then

$$\boldsymbol{A} = \boldsymbol{D}^{\frac{1}{2}} \left(\boldsymbol{I}_n + \sum_{i=1}^{e} \boldsymbol{D}^{-\frac{1}{2}} (\boldsymbol{E}_i - \boldsymbol{D}_i) \boldsymbol{D}^{-\frac{1}{2}} \right) \boldsymbol{D}^{\frac{1}{2}} = \boldsymbol{D}^{\frac{1}{2}} \left(\boldsymbol{I}_n + \sum_{i=1}^{e} \boldsymbol{S}_i \right) \boldsymbol{D}^{\frac{1}{2}} \tag{2.3}$$

where $\boldsymbol{I}_n$ is the n by n identity matrix and we have defined $\boldsymbol{S}_i = \boldsymbol{D}^{-\frac{1}{2}} (\boldsymbol{E}_i - \boldsymbol{D}_i) \boldsymbol{D}^{-\frac{1}{2}}$.

The first critical step is to make the approximation

$$\boldsymbol{I}_n + \sum_{i=1}^{e} \boldsymbol{S}_i \approx \prod_{i=1}^{e} (\boldsymbol{I}_n + \boldsymbol{S}_i) \tag{2.4}$$

The error in this approximation may be expressed in terms of second and higher order products of the components, and thus the approximation will be good if either the individual $\boldsymbol{S}_i$ are small or zero (this is likely to be true if $\boldsymbol{E}_i$ is very diagonally dominant), or the product of the overlapping components $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ is small or zero.

As $\boldsymbol{E}_i$ is positive semi-definite, it directly follows that $\boldsymbol{I}_n + \boldsymbol{S}_i$ is positive definite and thus has a Cholesky factorization (see Theorem 5.3 in Chapter 5 of [17])

$$\boldsymbol{W}_i \stackrel{\text{def}}{=} \boldsymbol{I}_n + \boldsymbol{S}_i = \boldsymbol{L}_i \boldsymbol{L}_i^{\mathrm{T}} \tag{2.5}$$

The matrix $\boldsymbol{W}_i$ is known as the *Winget decomposition* of $\boldsymbol{E}_i$ (see [16]). Notice that if $\boldsymbol{E}_i$ has non-zeros in e_i rows and columns, the Cholesky factor of its Winget decomposition will differ from the identity matrix only in these rows. This symmetric decomposition of $\boldsymbol{W}_i$ is the second critical step. Combining (2.3), (2.4) and (2.5), we have

$$\boldsymbol{A} \approx \boldsymbol{D}^{\frac{1}{2}} \left(\prod_{i=1}^{e} \boldsymbol{L}_i \boldsymbol{L}_i^{\mathrm{T}} \right) \boldsymbol{D}^{\frac{1}{2}} \tag{2.6}$$

Unfortunately, (2.6) is not symmetric, and thus is not a satisfactory preconditioner. The third crucial step is to make the further symmetrizing approximation

$$\prod_{i=1}^{e} \boldsymbol{L}_i \boldsymbol{L}_i^{\mathrm{T}} \approx \left(\prod_{i=1}^{e} \boldsymbol{L}_i \right) \left(\prod_{i=e}^{1} \boldsymbol{L}_i^{\mathrm{T}} \right) \tag{2.7}$$

This approximation is, as before, exact if there is no overlap between the blocks and will be good under exactly the same circumstances as its predecessor. We thus obtain the final approximation

$$\boldsymbol{A} \approx \boldsymbol{P}_{\mathrm{EBE}} = \boldsymbol{D}^{\frac{1}{2}} \left(\prod_{i=1}^{e} \boldsymbol{L}_i \right) \left(\prod_{i=e}^{1} \boldsymbol{L}_i^{\mathrm{T}} \right) \boldsymbol{D}^{\frac{1}{2}} \tag{2.8}$$

which may be used as a preconditioner for $\boldsymbol{A}$. Such a matrix is known as the *EBE* preconditioner. In order to solve the system of equations $\boldsymbol{P}_{\mathrm{EBE}}\boldsymbol{x} = \boldsymbol{y}$ efficiently, we exploit the decomposition (2.8).

We are free to order the elements in any way we choose and may thus encourage parallelism by ordering non-overlapping elements consecutively so that we can perform groups of forward and back-solves in parallel. Clearly, the efficiency of the EBE preconditioner depends on the partitioning of the initial matrix and on the size of the off-diagonal elements of the elementary matrices. With this in mind, the final critical ingredient is to *preprocess* the problem to amalgamate elements into *super-elements* with the aim of reducing the overlap between these super-elements. Reference [6] demonstrates that this is necessary in order to make the preconditioner effective in practice. It has the additional benefit that vectorization is more effective with the larger super-elements.

2.2. *Subspace-by-subspace preconditioners*

The derivation in the previous section is appropriate whether or not $\boldsymbol{E}_i$ is rank deficient. However, this is not true of an efficient implementation as we shall now see.

We suppose that $\boldsymbol{E}_i$ has non-zeros in e_i rows and columns (it can even be dense, i.e., $e_i = n$). As we have already observed, the Cholesky factors of its Winget decomposition will then differ from the identity matrix only in these *elemental* rows and columns. Denoting the non-zero rows and columns of $\boldsymbol{E}_i$ by $\boldsymbol{E}_i^{\mathscr{E}_i}$, and using similar definitions for $\boldsymbol{W}_i$, $\boldsymbol{D}_i$ and $\boldsymbol{D}$, we obtain

$$\boldsymbol{W}_i^{\mathscr{E}_i} = \boldsymbol{I}_{e_i} + (\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}}(\boldsymbol{E}_i^{\mathscr{E}_i} - \boldsymbol{D}_i^{\mathscr{E}_i})(\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}} = \boldsymbol{1}_i^{\mathscr{E}_i} + (\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}}\boldsymbol{E}_i^{\mathscr{E}_i}(\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}} \quad (2.9)$$

where $\boldsymbol{1}_i^{\mathscr{E}_i} = \boldsymbol{I}_{e_i} - (\boldsymbol{D}^{\mathscr{E}_i})^{-1}\boldsymbol{D}_i^{\mathscr{E}_i}$ is positive semi-definite. We suppose, for now, that $\boldsymbol{1}_i^{\mathscr{E}_i}$ has positive values, but will shortly return to the singular case.

Now suppose that $\boldsymbol{E}_i$ is of rank r_i. Then we see immediately that $\boldsymbol{W}_i^{\mathscr{E}_i}$ is a rank r_i modification of the positive-definite diagonal matrix $\boldsymbol{1}_i^{\mathscr{E}_i}$. Thus, if $r_i < e_i$ it would seem to be preferable to *update* the Cholesky factors of $\boldsymbol{W}_i^{\mathscr{E}_i}$ following a sequence of rank-r_i modifications rather than assembling and factoring $\boldsymbol{W}_i^{\mathscr{E}_i}$ directly (see, for example, [11]). More importantly, if $r_i \ll e_i$, an alternative to the Cholesky factorization more suited to the nature of $\boldsymbol{E}_i$ is clearly desirable.

Let $\boldsymbol{B}_i^{\mathscr{E}_i} = (\boldsymbol{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}}(\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}}\boldsymbol{E}_i^{\mathscr{E}_i}(\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}}(\boldsymbol{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}}$ be a rescaling of $\boldsymbol{E}_i^{\mathscr{E}_i}$. Then we may write (2.9) as

$$\boldsymbol{W}_i^{\mathscr{E}_i} = (\boldsymbol{1}_i^{\mathscr{E}_i})^{\frac{1}{2}}\left(\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathscr{E}_i}\right)(\boldsymbol{1}_i^{\mathscr{E}_i})^{\frac{1}{2}} \quad (2.10)$$

We now aim to decompose $\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathscr{E}_i}$ into the symmetric product of easily invertible parts. We suppose we may find a decomposition of $\boldsymbol{B}_i^{\mathscr{E}_i}$ of the form

$$\begin{aligned}\boldsymbol{B}_i^{\mathscr{E}_i} &= \boldsymbol{Q}_i^{\mathscr{I}_i}\begin{pmatrix}\boldsymbol{B}_i^{\mathscr{I}_i} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}(\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} = (\boldsymbol{Y}_i^{\mathscr{I}_i}\ \ \boldsymbol{Z}_i^{\mathscr{I}_i})\begin{pmatrix}\boldsymbol{B}_i^{\mathscr{I}_i} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\begin{pmatrix}(\boldsymbol{Y}_i^{\mathscr{I}_i})^{\mathrm{T}}\\ (\boldsymbol{Z}_i^{\mathscr{I}_i})^{\mathrm{T}}\end{pmatrix}\\ &= \boldsymbol{Y}_i^{\mathscr{I}_i}\boldsymbol{B}_i^{\mathscr{I}_i}(\boldsymbol{Y}_i^{\mathscr{I}_i})^{\mathrm{T}}\end{aligned} \quad (2.11)$$

where $\boldsymbol{Q}_i^{\mathscr{I}_i}$ is orthogonal and defines a transformation from the elemental to an *internal*

representation of $\boldsymbol{B}_i^{\mathscr{E}_i}$, and this representation, $\boldsymbol{B}_i^{\mathscr{I}_i}$, is an r_i by r_i symmetric positive definite matrix (for instance, but not restricted to, tridiagonal, see [20], Chapter 7). Furthermore, let $\boldsymbol{L}_i^{\mathscr{I}_i}$ be the Cholesky factor of $\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathscr{I}_i}$. Then

$$\begin{aligned}
\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathscr{E}_i} &= \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} \boldsymbol{I}_{r_i} + \boldsymbol{B}_i^{\mathscr{I}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} \\
&= \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} \boldsymbol{L}_i^{\mathscr{I}_i} (\boldsymbol{L}_i^{\mathscr{I}_i})^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} \\
&= \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} \boldsymbol{L}_i^{\mathscr{I}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} (\boldsymbol{L}_i^{\mathscr{I}_i})^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} \\
&= \boldsymbol{M}_i^{\mathscr{I}_i} (\boldsymbol{M}_i^{\mathscr{I}_i})^{\mathrm{T}}
\end{aligned} \tag{2.12}$$

where

$$\boldsymbol{M}_i^{\mathscr{I}_i} \stackrel{\text{def}}{=} \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} \boldsymbol{L}_i^{\mathscr{I}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} \tag{2.13}$$

Hence, we have obtained a factorization of the Winget decomposition of the form

$$\boldsymbol{W}_i^{\mathscr{E}_i} = (\mathbf{1}_i^{\mathscr{E}_i})^{\frac{1}{2}} \boldsymbol{M}_i^{\mathscr{I}_i} (\boldsymbol{M}_i^{\mathscr{I}_i})^{\mathrm{T}} (\mathbf{1}_i^{\mathscr{E}_i})^{\frac{1}{2}} \tag{2.14}$$

We may now use this as the basis of an EBE-like method. In particular, the resulting preconditioner is of the form

$$\boldsymbol{A} \approx \boldsymbol{P}_{SBS} = \boldsymbol{D}^{\frac{1}{2}} \left(\prod_{i=1}^{e} (\mathbf{1}_i)^{\frac{1}{2}} \boldsymbol{M}_i \right) \left(\prod_{i=e}^{1} (\boldsymbol{M}_i)^{\mathrm{T}} (\mathbf{1}_i)^{\frac{1}{2}} \right) \boldsymbol{D}^{\frac{1}{2}} \tag{2.15}$$

where $\mathbf{1}_i$ and $\boldsymbol{M}_i$ are simply $\mathbf{1}_i^{\mathscr{E}_i}$ and $\boldsymbol{M}_i^{\mathscr{I}_i}$ appropriately embedded (in their elemental row and column positions) within $\boldsymbol{I}_n$. We refer to this as a *subspace-by-subspace* (SBS) preconditioner because of the dependence of $\boldsymbol{M}_i^{\mathscr{I}_i}$ on the subspaces defined by the matrices $\boldsymbol{Y}_i^{\mathscr{I}_i}$ and $\boldsymbol{Z}_i^{\mathscr{I}_i}$.

At first glance we do not appear to have gained anything by this. In particular, the forward and back substitutions required when using (2.15) appear to be at least as expensive as using (2.8). However, more careful consideration reveals that this may not be so. Consider, for instance, the single step $(\mathbf{1}_i^{\mathscr{E}_i})^{\frac{1}{2}} \boldsymbol{M}_i^{\mathscr{I}_i} \boldsymbol{x}^{\mathscr{I}_i} = \boldsymbol{y}^{\mathscr{I}_i}$. Using the orthogonality and partitioning of $\boldsymbol{Q}_i^{\mathscr{I}_i}$ and (2.13), we have that

$$\begin{aligned}
\boldsymbol{x}^{\mathscr{I}_i} &= \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} (\boldsymbol{L}_i^{\mathscr{I}_i})^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix} (\boldsymbol{Q}_i^{\mathscr{I}_i})^{\mathrm{T}} (\mathbf{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}} \boldsymbol{y}^{\mathscr{I}_i} \\
&= \left(\boldsymbol{Y}_i^{\mathscr{I}_i} (\boldsymbol{L}_i^{\mathscr{I}_i})^{-1} (\boldsymbol{Y}_i^{\mathscr{I}_i})^{\mathrm{T}} + \boldsymbol{Z}_i^{\mathscr{I}_i} (\boldsymbol{Z}_i^{\mathscr{I}_i})^{\mathrm{T}} \right) (\mathbf{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}} \boldsymbol{y}^{\mathscr{I}_i} \\
&= \left(\boldsymbol{I}_{e_i} + \boldsymbol{Y}_i^{\mathscr{I}_i} \left((\boldsymbol{L}_i^{\mathscr{I}_i})^{-1} - \boldsymbol{I}_{r_i} \right) (\boldsymbol{Y}_i^{\mathscr{I}_i})^{\mathrm{T}} \right) (\mathbf{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}} \boldsymbol{y}^{\mathscr{I}_i}
\end{aligned} \tag{2.16}$$

The matrix $\boldsymbol{Q}_i^{\mathscr{I}_i}$ is not required, merely its first r_i columns $\boldsymbol{Y}_i^{\mathscr{I}_i}$. Thus we see that the principal costs are two matrix vector products with matrices of dimensions r_i by e_i and e_i by r_i, respectively, and a single triangular solve with a matrix of order r_i. The comparative cost of a forward substitution in (2.8) is for a triangular solve with a matrix of order e_i.

Neglecting lower-order terms, this indicates that the SBS approach is seen to be more efficient whenever $2r_i e_i + \frac{1}{2}r_i^2 \leq \frac{1}{2}e_i^2$, that is whenever

$$r_i \leq (\sqrt{5} - 2)e_i \approx 0.236e_i \tag{2.17}$$

The other relevant step $(\boldsymbol{M}_i^{\mathcal{J}_i})^{\mathrm{T}}(\mathbf{1}_i^{\mathcal{E}_i})^{\frac{1}{2}}\boldsymbol{x}^{\mathcal{J}_i} = \boldsymbol{y}^{\mathcal{J}_i}$ is very similar. For in this case, we have that

$$\begin{aligned}
\boldsymbol{x}^{\mathcal{J}_i} &= (\mathbf{1}_i^{\mathcal{E}_i})^{-\frac{1}{2}}\boldsymbol{Q}_i^{\mathcal{J}_i}\begin{pmatrix} (\boldsymbol{L}_i^{\mathcal{J}_i})^{-\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{e_i - r_i} \end{pmatrix}(\boldsymbol{Q}_i^{\mathcal{J}_i})^{\mathrm{T}}\boldsymbol{y}^{\mathcal{J}_i} \\
&= (\mathbf{1}_i^{\mathcal{E}_i})^{-\frac{1}{2}}\left(\boldsymbol{Y}_i^{\mathcal{J}_i}(\boldsymbol{L}_i^{\mathcal{J}_i})^{-T}(\boldsymbol{Y}_i^{\mathcal{J}_i})^{\mathrm{T}} + \boldsymbol{Z}_i^{\mathcal{J}_i}(\boldsymbol{Z}_i^{\mathcal{J}_i})^{\mathrm{T}}\right)\boldsymbol{y}^{\mathcal{J}_i} \\
&= (\mathbf{1}_i^{\mathcal{E}_i})^{-\frac{1}{2}}\left(\boldsymbol{I}_{e_i} + \boldsymbol{Y}_i^{\mathcal{J}_i}\left((\boldsymbol{L}_i^{\mathcal{J}_i})^{-\mathrm{T}} - \boldsymbol{I}_{r_i}\right)(\boldsymbol{Y}_i^{\mathcal{J}_i})^{\mathrm{T}}\right)\boldsymbol{y}^{\mathcal{J}_i}
\end{aligned} \tag{2.18}$$

and the principal costs are identical to the previous case.

We now consider how to cope with the possibility that $\mathbf{1}_i^{\mathcal{E}_i}$ may be singular. Notice that $\mathbf{1}_i^{\mathcal{E}_i}$ will actually be positive definite if and only if each elemental variable occurs in at least one other element. Any variable which occurs in a single element is said to be *exposed*. An exposed variable may be directly eliminated *within its element* (this is also called static condensation in finite element techniques); the resulting smaller element, formed from the Schur complement following this elimination, will itself be positive semi-definite. At first sight, it might then appear that it suffices to directly eliminate all exposed variables. However this is not so, as these eliminations may expose more variables *in the reduced problem*. For example, suppose element i involves variables 1, 2, 3, 4 and $\boldsymbol{E}_i^{\mathcal{E}_i}$ is of the form

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 3 \end{pmatrix} \tag{2.19}$$

and that variable 1 occurs in no other elements, while variable 2 appears in precisely one other element, element j. If we eliminate variable 1 within element i, we obtain the Schur complement

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

But now variable 2 does not occur in the reduced $\boldsymbol{E}_i^{\mathcal{E}_i}$,

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

and only occurs in element j. Thus, in the reduced problem variable 2 is exposed. This has happened simply because the original element $\boldsymbol{E}_i^{\mathcal{E}_i}$ was singular, but not all singular elements will automatically expose new variables when current exposes are eliminated.

Fortunately, a simple scheme for removing all exposed variables is obvious. At each stage, eliminate all currently exposed variables. Now check if extra variables have been exposed. If so, start the next stage. If not, the reduced problem has no exposed variables, and thus the resulting $\mathbf{1}_i^{\mathcal{E}_i}$ are all positive definite. Notice that, as all eliminations take place within

elements, no communication is required during the elimination. At the end of each stage, the list of elements containing each variable may be simply revised, and newly exposed variables detected. This process ends in at most n steps.

2.3. *Subspace-by-subspace preconditioners for structured problems*

We now return to our original problem, that is the case for which $\boldsymbol{E}_i = \boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}}$. In this case, it is straightforward to compute the required matrices $\boldsymbol{Q}_i^{\mathscr{I}_i}$ (or $\boldsymbol{Y}_i^{\mathscr{I}_i}$) and $\boldsymbol{B}_i^{\mathscr{I}_i}$ in (2.11). Suppose that $\boldsymbol{A}_i$ has non-zeros in e_i rows. Denoting these rows by $\boldsymbol{A}_i^{\mathscr{E}_i}$, we obtain

$$\boldsymbol{B}_i^{\mathscr{E}_i} = \boldsymbol{C}_i^{\mathscr{E}_i} (\boldsymbol{C}_i^{\mathscr{E}_i})^{\mathrm{T}} \tag{2.20}$$

where $\boldsymbol{C}_i^{\mathscr{E}_i} = (\mathbf{1}_i^{\mathscr{E}_i})^{-\frac{1}{2}} (\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}} \boldsymbol{A}_i^{\mathscr{E}_i}$. Now, let

$$\boldsymbol{C}_i^{\mathscr{E}_i} = \boldsymbol{Q}_i^{\mathscr{I}_i} \begin{pmatrix} \boldsymbol{R}_i^{\mathscr{I}_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \boldsymbol{V}_i^{\mathscr{I}_i} \tag{2.21}$$

where $\boldsymbol{Q}_i^{\mathscr{I}_i}$ and $\boldsymbol{V}_i^{\mathscr{I}_i}$ are orthogonal and $\boldsymbol{R}_i^{\mathscr{I}_i}$ is triangular and of rank r_i, be a complete orthogonal decomposition of $\boldsymbol{C}_i^{\mathscr{E}_i}$ (see, for instance, [1]). Then clearly, $\boldsymbol{B}_i^{\mathscr{I}_i} = \boldsymbol{R}_i^{\mathscr{I}_i} (\boldsymbol{R}_i^{\mathscr{I}_i})^{\mathrm{T}}$, and we have the ingredients of (2.11). The decomposition (2.21) may be determined by a QR factorization with column pivoting, while the actual form we require, involving only the first r_i columns $\boldsymbol{Y}_i^{\mathscr{I}_i}$ or $\boldsymbol{Q}_i^{\mathscr{I}_i}$, may be obtained using the modified Gram–Schmidt process (again with column pivoting). We should caution the reader that under exceptional circumstances these methods may incorrectly estimate the rank of $\boldsymbol{C}_i^{\mathscr{E}_i}$, and a singular-value decomposition may be preferred. Again, see [1] for details.

A potential difficulty occurs when $\mathbf{1}_i^{\mathscr{E}_i}$ is singular. As we have already mentioned, this can only happen if one or more elemental variables are restricted to this single element. We mentioned that in this case we may directly eliminate these exposed variables, and the resulting smaller element is still positive semi-definite. However, unless we are careful, it may not inherit the structure (2.20). We now show that, in fact, we can still arrange the computation so that the smaller element is of the form (2.20).

To see this, suppose, without loss of generality, that the first k elemental variables only occur in $\boldsymbol{E}_i^{\mathscr{E}_i}$. We may then find an orthogonal matrix $\boldsymbol{U}_i^{\mathscr{E}_i}$ so that

$$(\boldsymbol{D}^{\mathscr{E}_i})^{-\frac{1}{2}} \boldsymbol{A}_i^{\mathscr{E}_i} \boldsymbol{U}_i^{\mathscr{E}_i} = \begin{pmatrix} \boldsymbol{R}_i^{\mathscr{E}_i} & \mathbf{0} \\ \widehat{\boldsymbol{A}}_i^{\mathscr{E}_i} & \overline{\boldsymbol{A}}_i^{\mathscr{E}_i} \end{pmatrix} \tag{2.22}$$

where $\boldsymbol{R}_i^{\mathscr{E}_i}$ is k by k, non-singular and upper triangular—the matrix $\boldsymbol{U}_i^{\mathscr{E}_i}$ may, for instance be formed as a product of plane rotations. We may then write (2.9) as

$$\begin{aligned} \boldsymbol{W}_i^{\mathscr{E}_i} &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{1}}_i^{\mathscr{E}_i} \end{pmatrix} + \begin{pmatrix} \boldsymbol{R}_i^{\mathscr{E}_i} & \mathbf{0} \\ \widehat{\boldsymbol{A}}_i^{\mathscr{E}_i} & \overline{\boldsymbol{A}}_i^{\mathscr{E}_i} \end{pmatrix} \begin{pmatrix} (\boldsymbol{R}_i^{\mathscr{E}_i})^{\mathrm{T}} & (\widehat{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}} \\ \mathbf{0} & (\overline{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{R}_i^{\mathscr{E}_i} (\boldsymbol{R}_i^{\mathscr{E}_i})^{\mathrm{T}} & \boldsymbol{R}_i^{\mathscr{E}_i} (\widehat{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}} \\ \widehat{\boldsymbol{A}}_i^{\mathscr{E}_i} (\boldsymbol{R}_i^{\mathscr{E}_i})^{\mathrm{T}} & \overline{\mathbf{1}}_i^{\mathscr{E}_i} + \widehat{\boldsymbol{A}}_i^{\mathscr{E}_i} (\widehat{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}} + \overline{\boldsymbol{A}}_i^{\mathscr{E}_i} (\overline{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}} \end{pmatrix} \end{aligned} \tag{2.23}$$

Eliminating the first k elemental variables then leaves the Schur complement $\overline{\mathbf{1}}_i^{\mathscr{E}_i} + \overline{\boldsymbol{A}}_i^{\mathscr{E}_i}(\overline{\boldsymbol{A}}_i^{\mathscr{E}_i})^{\mathrm{T}}$, which is of the form (2.9), but now with $\overline{\mathbf{1}}_i^{\mathscr{E}_i}$ non-singular. Notice, the matrix $\boldsymbol{U}_i^{\mathscr{E}_i}$ need not be stored.

Unfortunately, this does not completely remove the problem because, although the Schur complement is of the correct form, it may happen that one or more of the rows of the reduced matrix $\overline{\boldsymbol{A}}_i^{\mathscr{E}_i}$ contains only zeros. Thus, the variable associated with this row is no longer involved in the ith reduced element, and this may expose the variable within another element. As in the more general case considered at the end of Section 2.2, a number of stages may be required, each eliminating exposed variables and marking any further exposed variables for elimination at the next stage.

3. Least squares problems

3.1. Development

A rich source of systems of the form (1.1)–(1.2) are least-squares problems,

$$\underset{x\in\Re^n}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2 \tag{3.1}$$

where $\boldsymbol{A}$ is an m by n rectangular matrix with $m > n$. A solution to (3.1) satisfies the normal equations

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}. \tag{3.2}$$

If we group rows of $\boldsymbol{A}$ so that

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 \\ \cdots \\ \boldsymbol{A}_e \end{pmatrix} \tag{3.3}$$

then (3.2) is simply

$$\sum_{i=1}^{e} \boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{x} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} \tag{3.4}$$

which is of the form (1.1)–(1.2). Clearly there is considerable freedom in the partitioning of $\boldsymbol{A}$ into (3.3). Extreme examples are $\boldsymbol{A}_i = \boldsymbol{a}_i^{\mathrm{T}}$ and $e = m$, where $\boldsymbol{a}_i^{\mathrm{T}}$ is the ith row of $\boldsymbol{A}$, or $\boldsymbol{A}_1 = \boldsymbol{A}$ and $e = 1$. We wish to solve (3.2) using a suitably preconditioned variant of conjugate gradients appropriate for least-squares problems (see, [1], Sections 7.4 and 7.5). We also wish to use the flexibility of the form (3.4) to construct suitable subspace-by-subspace preconditioners.

The SBS(1) preconditioner simply chooses $\boldsymbol{A}_i = \boldsymbol{a}_i^{\mathrm{T}}$ and $e = m$ and follows the construction in Sections 2.2 and 2.3. It is, moreover, easy to detect and eliminate exposed variables before constructing the preconditioner. To do this, we first initialize an empty list of rows $\mathscr{R}$ and variables $\mathscr{C}$ to be directly eliminated. We now scan the rows and columns of $\boldsymbol{A}$ not in $\mathscr{R}$ and $\mathscr{C}$, respectively, for a column singleton. If one is found, we add its index to $\mathscr{C}$, add the row to $\mathscr{R}$ and repeat the search. If none are found, all remaining columns have two or more non-zero entries (or are null which implies that $\boldsymbol{A}$ is rank deficient). If we suppose that $\boldsymbol{A}$ is of full rank, this implies that we can permute the rows and columns

Algorith *Let $\boldsymbol{a}_1 \ldots \boldsymbol{a}_m$ be given vectors.*
Compute $Occ(i)$, $i = 1, \ldots, n$, the number of $\boldsymbol{a}_j$ containing variable i.
Let S_l denote the current set of the vectors to be merged, and let
$Occs(i)$, $i = 1, \ldots, n$, be the number of occurrences of variable i within S_l
Set $l = 1$, $S_l = \emptyset$, $Occs(i) = 0$, $i = 1, \ldots, n$
For $k = 1, \ldots, m$
  $S_l = S_l + \{\boldsymbol{a}_k\}$
  Update Occs
  If $\exists i$ such that $Occs(i) = Occ(i)$, reset
    $S_l = S_l - \{\boldsymbol{a}_k\}$
    $l = l + 1$, $S_l = \emptyset$, $Occs(i) = 0$, $i = 1, \ldots, n$
  End If
End For

Figure 1. Construction of the sets of rank-one terms to be amalgamated

of $\boldsymbol{A}$ so that

$$\boldsymbol{PAQ} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{A}_e \\ \boldsymbol{0} & \boldsymbol{A}_r \end{pmatrix}$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are permutation matrices and each column of $\boldsymbol{A}_r$ has at least two non-zero entries. Substituting in (3.4) and simplifying reveals that if we solve

$$\boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{A}_r \boldsymbol{x}_r = \boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{b}_r \quad \text{and} \tag{3.5}$$

$$\boldsymbol{R}\boldsymbol{x}_e = \boldsymbol{b}_e - \boldsymbol{A}_e \boldsymbol{x}_r \tag{3.6}$$

then

$$\boldsymbol{x} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{x}_e \\ \boldsymbol{x}_r \end{pmatrix}, \quad \text{where} \quad \begin{pmatrix} \boldsymbol{b}_e \\ \boldsymbol{b}_r \end{pmatrix} = \boldsymbol{Pb}$$

Of course, (3.5) are the normal equations for the reduced least-squares problem

$$\underset{x_r \in \Re^{n_r}}{\text{minimize}} \quad \|\boldsymbol{A}_r \boldsymbol{x}_r - \boldsymbol{b}_r\|_2$$

The advantage is that each column of this problem has at least two non-zeros and hence choosing $\boldsymbol{A}_i = \boldsymbol{a}_i^{\mathrm{T}}$ for this problem reveals no exposed variables.

A second possibility is to merge groups of rows of $\boldsymbol{A}$ to form the $\boldsymbol{A}_i$. We can then use the amalgamation algorithm described in Figure 1 to merge rank-one terms. It basically regroups rank-one terms to ensure that no variable belongs to a single element.

We also impose a threshold, $k_{\max}$, on the maximum number of rows allowed in an amalgamated element (i.e., the maximum size of the set S_l). In practice, in view of (2.17), $k_{\max}$ should be no larger than $(\sqrt{5} - 2) \times n$.

We recognize that the algorithm described in Figure 1 is quite naive. However, it has proved effective in practice and attempts to design more sophisticated algorithms—for instance, to try to group terms which have a large overlap together—have not proved significantly better.

Table 1. Characteristics of the test matrices. For each matrix, m gives the number of rows, n the number of columns, and nz the number of non-zeros. The number of variables directly eliminated is n_e, while n_s gives the resulting number of columns in the Schur complement. Columns 7–10 give the degree of overlap defined as the average number of groups sharing each variable and the average number of rank-one terms within each group (which is also the average rank of the groups). The column headed κ gives the condition number of each problem

						Degree of overlap/average group size				
						$k_{\max} =$				
Name	m	n	nz	n_e	n_s	1	5	20	50	κ
il1033	1 033	320	4 732	12	308	15.2/1.0	5.9/5.0	3.8/17.3	3.2/30.0	1.9E+4
BRATU1D	4 007	3 004	6 007	1	3 003	2.6/1.0	2.1/5.0	2.0/19.9	2.0/49.6	6.5E+5
TRIDIA	2 000	1 000	2 998	1	999	3.0/1.0	2.2/5.0	2.0/20.0	2.0/50.0	7.1E+3
BROWNBS	3 997	2 997	6 993	0	2 997	2.3/1.0	2.1/5.0	2.0/20.0	2.0/50.0	1.4E+6
DQDRTIC	3 994	2 994	5 988	0	2 994	2.0/1.0	2.0/5.0	2.0/20.0	2.0/49.9	5.6E+9
EXPFITC	1 009	502	2 755	0	502	5.5/1.0	3.0/5.0	3.0/19.8	2.9/48.0	7.3E+2
HYDCAR20	99	99	734	0	99	7.4/1.0	4.3/4.9	3.5/19.8	3.3/19.8	1.0E+6
MAXLIKA	243	235	2 003	0	235	8.5/1.0	3.0/5.0	2.0/18.7	2.0/40.5	4.2E+1
ORTHREGC	1 005	500	3 500	0	500	7.0/1.0	2.2/5.0	2.1/19.7	2.0/47.8	2.0E+2

3.2. Numerical experiments

We have tested the SBS preconditioner on a number of rectangular matrices from the Harwell–Boeing collection (see [9]), and on a number of Jacobian matrices from problems arising from the CUTE collection (see [2]). The characteristics of these problems are summarized in Table 1. We include details of how many exposed variables can be trivially removed, and the resulting problem sizes. Further experiments are reported in [7].

The matrix il1033 was derived from an Harwell–Boeing matrix in [18], and subsequently used in tests in [1]. The remaining matrices were chosen arbitrarily from the CUTE collection.

3.2.1. Experiments with zero residual problems

For our first set of experiments, we consider the least-squares solution of overdetermined, consistent sets of equations. The zero-residual problems $\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ are defined by requiring the exact solution to be $\boldsymbol{x}^* = (1, \ldots, 1)^{\mathrm{T}}$, and setting $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^*$. The origin is taken as the initial estimate of the solution.

In Table 2, we compare the conjugate gradient (CG) solution of the normal equations without preconditioning, to the same method preconditioned by a diagonal preconditioner, a band approximation—in our case band(k), a band matrix with semi-bandwidth k whose non-zeros are those from $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$—or by an SBS($k_{\max}$) preconditioner, whose elements are composed of at most $k_{\max}$ rank-one terms using the algorithm described in Figure 1. For the band preconditioner, once we have the approximation to the band of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, we use the LDL^{T} band solver from the LANCELOT package (see [4]). We note that, in this case, the factors may be modified to guarantee that the preconditioner is safely positive definite. The diagonal preconditioner is simply band(0).

We use the variant of the CG method for least-squares problems described in [1], Section 7.4.1). Convergence is recorded as soon as

$$\|\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b})\|_2 \leq 10^{-15}\|\boldsymbol{b}\|_2 \tag{3.7}$$

We report #it, the number of iterations needed for convergence, t_{con}, the construction time (in CPU seconds) for the preconditioner, t_{cg}, the convergence time for the CG method neglecting the construction time, and *err*, the relative error $\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|_2/\|\boldsymbol{x}^*\|_2$ in the computed solution $\bar{\boldsymbol{x}}$. A star indicates that the method reached a maximum number of iterations without satisfying (3.7)—in our experiments, a maximum of $10n$ iterations were permitted. The entries marked in bold correspond to the method(s) which performed best in terms of total time required. All of the experiments reported in this paper were performed on a SUN workstation with a 125 MHz HyperSPARC processor.

For the nine problems tested, the unpreconditioned method performed best (in terms of total computational time) in two cases, the diagonal preconditioner proved to be best in two cases, the band in two cases, the SBS in two cases, while the diagonal and SBS preconditioners tied on problem `EPXFITC`. For the problems `il1033` only the SBS preconditioner converges within the permitted number of iterations.

In most cases, the SBS preconditioner converges in fewer iterations than its competitors. However, this efficiency in number of iterations is not systematically reflected in the computational time as one SBS iteration is typically more costly than one band iteration. For one problem (`TRIDIA`), the band preconditioner requires significantly fewer iterations than the SBS preconditioner. However, this is not surprising since these matrices are tridiagonal, and the CG method acts as a direct method with the band preconditioner. The problems `BRATU1D`, `BROWNBS` and `DQDRTIC` also exhibit a band structure with semi-bandwidths, respectively, of 5, 4 and 7. As we have already noted, the factors of the band are sometimes modified to ensure that the preconditioner is safely positive definite. The perturbations introduced on problems `BRATU1D` and `BROWNBS` explain why the band preconditioner is not exact even when k exceeds the semi-bandwidth of the problem. This is not the case on `DQDRTIC` and convergence is achieved in one iteration when k is larger than 7. It also helps to explain why the band preconditioner may occasionally be worse than no preconditioning.

In Figure 2, we show the eigenvalue spectrum of the unpreconditioned system $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and the eigenvalue spectrum of the preconditioned system using SBS(1) for the problem `DQDRTIC`.

The structure of the problems `il1033` and `HYDCAR20` is very irregular, the non-zeros being spread throughout the matrix. SBS preconditioners seem more able to deal with such irregular structure than its competitors.

The problems `EXPFITC`, `MAXLIKA` and `ORTHREGC` have a very similar structure, in that the first few columns of $\boldsymbol{A}$ are (almost or completely) dense leading to a dense $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. We consider the problem `EXPFITC` in detail. Columns 1 to 3 of the matrix $\boldsymbol{A}$ involve roughly half the variables, while columns 4 and 5 are completely dense. The main diagonal of the matrix is also dense. As a consequence, the matrix $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ is dense, and clearly a band preconditioner has no little chance of success. SBS proves to be effective as soon as $k_{\max}$ is large enough to capture the first five columns in a single group, which happens whenever $k_{\max} \geq 5$. We show in Figure 3 the eigenvalue spectrum of the unpreconditioned problem and the eigenvalue spectrum of the preconditioned system using SBS(1) and SBS(5). The eigenvalue spectrum of the preconditioned system is slightly improved compared with the unpreconditioned one. SBS(5) is able to capture the set of dense columns into a single

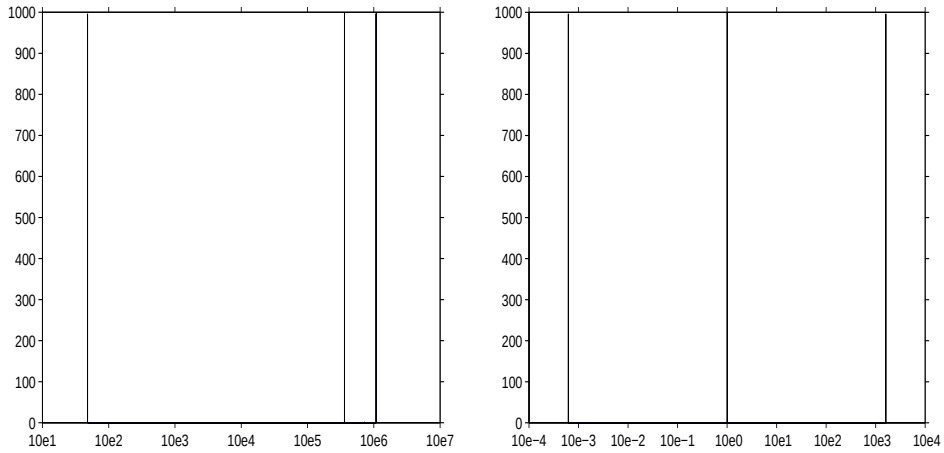

Figure 2. Eigenvalues of the problem DQDRTIC: unpreconditioned (left) and preconditioned using SBS(1) (right)

group, and the eigenvalues of the preconditioned system are significantly more clustered, which helps to explain the convergence in two iterations.

The impact of increasing $k_{\max}$ depends on the structure of the problems. There is little effect on problem il1033 for example, while the number of iterations settles down to two as $k_{\max}$ increases from 1 to 5 (and beyond) on the ORTHREGC problems. In some cases, increasing $k_{\max}$ increases the number of iterations, as we see for HYDCAR20. A small value of $k_{\max}$, say $k_{\max} = 5$ or 10, seems to be a good choice in our tests.

In general, the relative error is smaller when a SBS preconditioner is used. It is also evident that SBS usually (but not always) performs better than its competitors when the problem is ill-conditioned. For the well-conditioned examples, the cost of forming and applying the preconditioner does not, in general, pay off.

In summary, the SBS preconditioner appears to be an attractive alternative to less sophisticated possibilities when using the CG method to solve least-squares problems, particularly when the problem is ill-conditioned and when it does not have a kind of band structure. We have observed that SBS—as do other element-by-element preconditioners—takes advantage of a wider range of sparsity patterns than band preconditioners.

3.2.2. Experiments with non-zero residual problems

In addition to the consistent sets of equations we considered in the last section, we also performed tests on least-squares problems with non-zero residuals. The problems are constructed as follows. We build a random vector $\boldsymbol{b}_0$ in the interval [-1,1], and solve the least-squares problem $\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}_0\|_2$ using the LAPACK library. The residual $\boldsymbol{r}_0$ corresponding to this solution is almost surely non-zero and satisfies $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{r}_0 = \boldsymbol{0}$. This then ensures that the solution $\boldsymbol{x}^* = (1 \ldots 1)^{\mathrm{T}}$ we seek solves the least-squares problem $\min \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$, where $\boldsymbol{b} = \boldsymbol{r}_0 + \boldsymbol{A}\boldsymbol{x}^*$.

We report on experiments using six problems with non-zero residual in Table 3. These results are similar to those obtained with zero residual problems. Again the relative error in the solution is seen to be much smaller using SBS in many cases (see, for example,

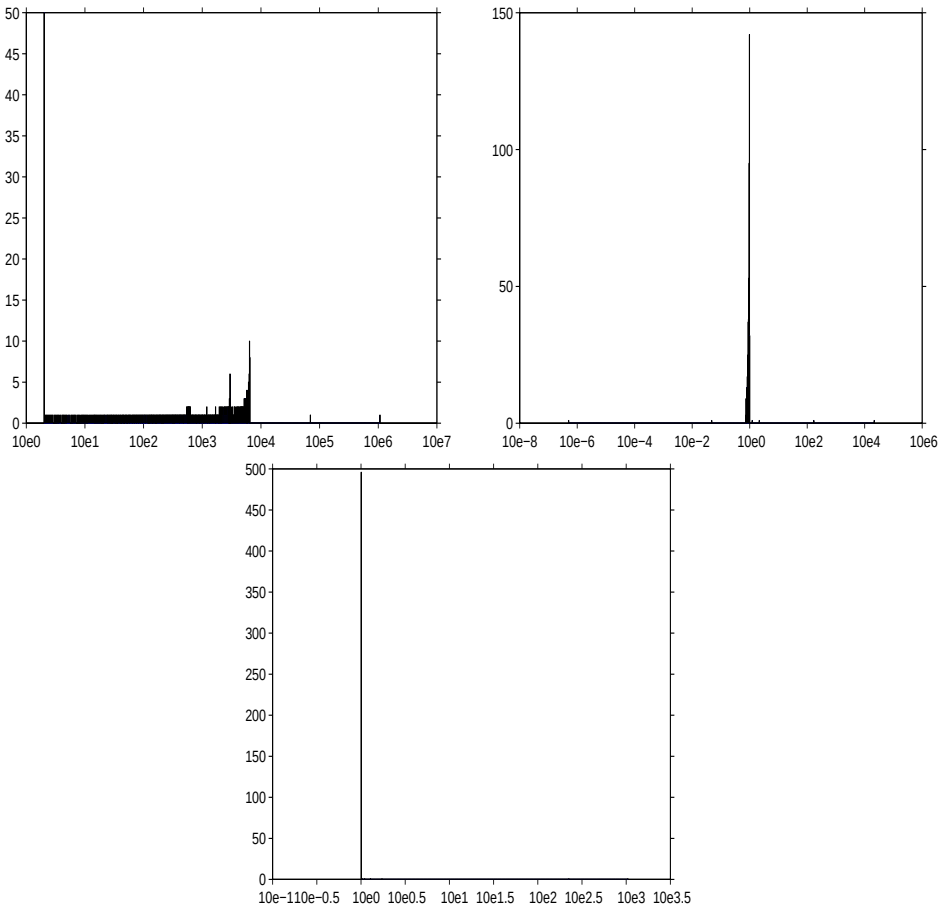

Figure 3. Eigenvalues of the problem EXFITC: unpreconditioned (top left), preconditioned using SBS(1) (top right) and preconditioned using SBS(5) (down centre)

Table 2. Results with no, diag, band(k), and SBS($k_{\max}$) preconditioners on zero residual problems

				band(k), k =				SBS($k_{\max}$), $k_{\max}$ =			
Name	no	diag		1	2	5	20	1	5	20	50
il1033	#it	3 080*	3 080*	3 080*	3 080*	3 080*	3 080*	1 835	1 827	1 739	1 640
	t_{con}	0.00	0.00	0.00	0.00	0.00	0.02	0.07	**0.05**	0.13	0.32
	t_{cg}	6.70	5.68	6.05	7.02	7.82	10.45	41.25	**20.30**	23.72	35.23
	err	2E-03	1E-03	2E-03	1E-03	5E-04	2E-02	3E-11	4E-11	3E-10	2E-11
BRATU1D	#it	25 210*	36	1 549	2 179	805	805	16	10	9	9
	t_{con}	0.00	**0.02**	0.02	0.02	0.03	0.12	0.98	0.68	0.65	2.20
	t_{cg}	146.27	**0.22**	11.58	18.43	9.18	22.05	1.30	0.45	0.65	1.25
	err	5E-01	4E-11	3E-11	8E-11	4E-07	4E-07	7E-16	1E-12	5E-14	3E-14
TRIDIA	#it	1 810	23	1	1	1	1	10	8	6	5
	t_{con}	0.00	0.00	**0.00**	0.00	0.02	0.03	0.23	0.17	0.30	0.80
	t_{cg}	4.43	0.07	**0.00**	0.02	0.02	0.03	0.38	0.17	0.17	0.27
	err	2E-15	2E-16	3E-16	3E-16	3E-16	3E-16	1E-16	4E-16	1E-16	1E-16
BROWNBS	#it	10	33	10	9	9	9	14	7	3	3
	t_{con}	**0.02**	0.02	0.00	0.00	0.03	0.15	1.25	0.98	1.22	2.90
	t_{cg}	**0.08**	0.30	0.12	0.13	0.15	0.32	1.32	0.37	0.30	0.58
	err	5E-04	2E-09	1E-04	2E-04	2E-04	2E-04	5E-10	1E-09	9E-10	2E-11
DQDRTIC	#it	11	5	5	5	5	1	1	1	1	1
	t_{con}	0.00	**0.00**	0.02	0.02	0.03	0.15	1.27	0.92	1.20	2.63
	t_{cg}	0.10	**0.07**	0.05	0.07	0.08	0.07	0.12	0.08	0.12	0.25
	err	6E-16	2E-14	2E-14	2E-13	5E-14	4E-16	4E-14	4E-14	4E-14	4E-14
EXPFITC	#it	843	33	242	298	265	215	26	2	2	2
	t_{con}	0.00	**0.05**	0.05	0.05	0.03	0.05	0.07	**0.07**	0.17	0.58
	t_{cg}	1.25	**0.05**	0.37	0.50	0.55	0.83	0.52	**0.03**	0.05	0.10
	err	3E-11	6E-12	1E-11	3E-12	9E-12	2E-11	6E-13	6E-15	1E-14	1E-14
HYDCAR20	#it	990*	990*	990*	990*	990*	397	716	414	285	326
	t_{con}	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.02	0.03
	t_{cg}	0.45	0.30	0.32	0.35	0.43	**0.32**	1.77	0.65	0.70	0.83
	err	5E-01	7E-02	8E00	9E-01	1E00	6E-10	5E-12	5E-11	6E-12	3E-11
MAXLIKA	#it	30	25	326	127	100	103	24	13	2	2
	t_{con}	**0.00**	0.02	0.03	0.02	0.02	0.05	0.00	0.02	0.03	0.08
	t_{cg}	**0.03**	0.02	0.25	0.12	0.10	0.18	0.17	0.05	0.02	0.02
	err	3E-14	5E-14	1E-13	9E-14	1E-13	1E-13	2E-14	8E-15	2E-15	2E-15
ORTHREGC	#it	290	135	654	337	296	297	77	2	2	2
	t_{con}	0.00	0.07	0.07	0.08	0.07	0.08	0.07	**0.05**	0.08	0.27
	t_{cg}	0.45	0.20	1.15	0.62	0.65	1.17	1.58	**0.02**	0.02	0.05
	err	1E-12	8E-13	7E-13	7E-13	6E-13	1E-12	3E-13	2E-15	2E-15	2E-15

Table 3. Results with no, diag, band(k), and SBS($_{\text{kmax}}$) preconditioners on non-zero residual problems. κ is the condition number of each matrix

				band(k), k =				SBS($k_{\max}$), $k_{\max}$ =			
Name		no	diag	1	2	5	20	1	5	20	50
BRATU1D	#it	25 210*	36	1 567	2 250	798	798	16	10	9	9
	t_{con}	0.00	**0.00**	0.02	0.00	0.03	0.10	0.92	0.70	0.95	2.13
κ=6.5E5	t_{mcg}	137.66	**0.27**	11.95	19.43	9.10	23.27	1.17	0.45	0.63	1.23
	err	0.4	5E-11	5E-11	5E-11	4E-07	4E-07	2E-11	2E-11	2E-11	2E-11
DQDRTIC	#it	11	5	5	5	5	1	1	1	1	1
	t_{con}	0.00	0.02	**0.00**	0.00	0.03	0.17	0.22	0.88	1.27	2.80
κ=5.6E9	t_{cg}	0.10	0.07	**0.07**	0.10	0.10	0.05	0.20	0.08	0.13	0.26
	err	3E-14	2E-14	2E-14	2E-13	3E-14	7E-15	3E-14	3E-14	3E-14	3E-14
HYDCAR20	#it	990*	990*	990*	990*	990*	397	716	414	285	326
	t_{con}	0.00	0.00	0.00	0.00	0.00	**0.00**	0.02	0.00	0.02	0.03
κ=1.0E6	t_{cg}	0.22	0.28	0.35	0.35	0.43	**0.28**	1.77	0.67	0.60	0.75
	err	0.5	7E-02	7.7	0.8	1.5	6E-10	5E-12	5E-11	6E-12	3E-11
BROWNBS	#it	10	33	10	7	9	9	14	7	3	3
	t_{con}	**0.00**	0.00	0.02	0.02	0.02	0.15	1.22	0.87	1.18	2.63
κ=1.4E6	t_{cg}	**0.08**	0.30	0.12	0.10	0.17	0.32	1.23	0.37	0.30	0.53
	err	5E-04	2E-09	1E-04	5E-04	1E-04	1E-04	5E-10	1E-09	9E-10	9E-11
EXPFITC	#it	901	34	270	339	279	227	22	2	2	2
	t_{con}	0.00	0.03	0.05	0.05	0.07	0.07	0.07	**0.05**	0.15	0.62
κ=7.3E2	t_{cg}	1.33	0.07	0.50	0.67	0.65	1.00	0.48	**0.02**	0.05	0.10
	err	8E-11	6E-11	6E-11	6E-11	6E-11	6E-11	6E-11	6E-11	6E-11	6E-11
ORTHREGC	#it	320	151	778	364	316	327	76	2	2	2
	t_{con}	0.00	0.07	0.07	0.07	0.08	0.10	0.08	**0.05**	0.10	0.27
κ=2.0E2	t_{cg}	0.42	0.23	1.53	0.73	0.78	1.48	1.63	**0.03**	0.03	0.05
	err	1E-10	1E-10	1E-10	1E-10	1E-10	1E-10	1E-10	1E-10	1E-10	1E-10

problems BRATU1D, BROWNBS and HYDCAR20). The SBS preconditioner is the fastest preconditioner for problems EXPFITC and ORTHREGC.

4. Mixing SBS with other element-by-element preconditioners

Quite clearly, SBS preconditioners will never be well suited to all problems, most particularly to those problems with elements of (close to) full rank. In this section we consider a second possibility, namely, to combine SBS preconditioners with other element-by-element preconditioners to handle substructures of low rank. We first give two examples.

Suppose we wish to find the least-squares solution to a *non-linear* set of equations by minimizing $\boldsymbol{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|_2^2$, where $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}) \dots f_m(\boldsymbol{x}))^{\mathrm{T}}$. Then the coefficient matrix associated with the Newton equations $\nabla_{xx}\boldsymbol{F}(\boldsymbol{x})\Delta x = -\nabla_x \boldsymbol{F}(\boldsymbol{x})$ is of the form

$$\nabla_{xx}\boldsymbol{F}(\boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{x})\nabla_{xx} f_i(\boldsymbol{x}) + \sum_{i=1}^{m} \nabla_x f_i(\boldsymbol{x})(\nabla_x f_i(\boldsymbol{x}))^{\mathrm{T}}. \tag{4.1}$$

The first summation in (4.1) is a generic 'sum of weighted elements', and would typically be treated using an EBE method. The second summation is of rank-one terms, and could be treated using the SBS methods developed in this paper.

A second example relates to the minimization of a non-linear function $f(\boldsymbol{x})$ subject to a set of (non-linear) constraints $\boldsymbol{c}_i(\boldsymbol{x}) = 0, i = 1, \ldots m$, using penalty or augmented Lagrangian methods (very similar arguments hold for inequality constraints using barrier methods). In this case, a penalty function of the form

$$\phi(\boldsymbol{x}, \boldsymbol{\lambda}, \rho) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{c}(\boldsymbol{x}) + \tfrac{1}{2}\rho\|\boldsymbol{c}(\boldsymbol{x})\|_2^2 \tag{4.2}$$

will be (approximately) minimized for a sequence of Lagrange multiplier estimates $\boldsymbol{\lambda}$ and penalty parameters ρ. Once again, Newton's method requires the solution of a system of linear equations whose coefficient matrix, $\nabla_{xx}\phi(\boldsymbol{x}, \boldsymbol{\lambda}, \rho)$ is of the form

$$\nabla_{xx}\phi(\boldsymbol{x}, \boldsymbol{\lambda}, \rho) = \nabla_{xx} f(\boldsymbol{x}) + \sum_{i=1}^{m}(\lambda_i + \rho c_i(\boldsymbol{x}))\nabla_{xx}c_i(\boldsymbol{x}) + \rho\sum_{i=1}^{m}\nabla_x c_i(\boldsymbol{x})(\nabla_x c_i(\boldsymbol{x}))^{\mathrm{T}}. \tag{4.3}$$

The first summation in (4.3) is again a generic 'sum of weighted elements', and can be treated using an EBE method, while the second summation, being a sum of rank-one terms, can be handled using an SBS method. Any additional structure within the remaining term $\nabla_{xx} f(\boldsymbol{x})$, such as might result if f is partially separable (see [13]), can be treated using EBE or SBS preconditioners, as appropriate.

In general, element-by-element and subspace-by-subspace preconditioners can be mixed in an obvious way. Suppose the generic matrix $\boldsymbol{A}$ is of the form

$$\boldsymbol{A} = \sum_{i=1}^{e_e}\boldsymbol{E}_i^e + \sum_{i=1}^{e_s}\boldsymbol{E}_i^s \tag{4.4}$$

where the elements in the first sum are of general 'finite element' type, while those in the second sum are of low rank. Then we can treat the $\boldsymbol{E}_i^e$ exactly as we described in Section 2.1, and the remaining terms as described in Section 2.2. We compute the preconditioners elementwise (as described earlier) and obtain the mixed preconditioner

$$\boldsymbol{A} \approx \boldsymbol{P}_{mix} = \boldsymbol{D}^{\frac{1}{2}}\left(\prod_{i=1}^{e_e}\boldsymbol{L}_i\prod_{i=1}^{e_s}(\mathbf{1}_i^{\mathscr{C}_i})^{\frac{1}{2}}\boldsymbol{M}_i^{\mathscr{I}_i}\right)\left(\prod_{i=e_s}^{1}(\boldsymbol{M}_i^{\mathscr{I}_i})^{\mathrm{T}}(\mathbf{1}_i^{\mathscr{C}_i})^{\frac{1}{2}}\prod_{i=e_e}^{1}\boldsymbol{L}_i^{\mathrm{T}}\right)\boldsymbol{D}^{\frac{1}{2}} \tag{4.5}$$

The order of the terms in (4.5) is arbitrary, and in practice the EBE and SBS terms in (4.5) might be interleaved to encourage parallelism. Preliminary tests do not, in general, reveal significant differences in performance when different orderings are used.

4.1. Tests on an artificial problem

We first illustrate the efficacy of such an approach on the artificial problem shown in Figure 4.1

This problem is formed by adding n_e elements, each having v_e variables and overlapping its neighbours by v_o variables, to a single rank-one element $\boldsymbol{a}\boldsymbol{a}^{\mathrm{T}}$ involving all the variables. The n_e diagonal blocks are randomly generated to have eigenvalues in the range $[1, \lambda_{\max}]$,

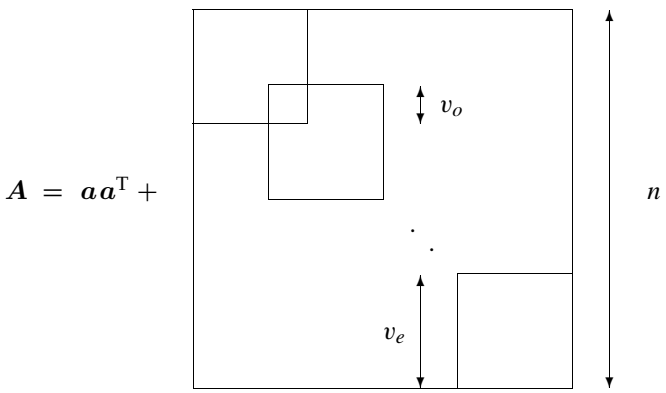

Figure 4. The test problem

while $a_i = 0.1i$ for $1 \leq i \leq n$. The right-hand side is chosen so that the exact solution is $(1, \ldots, 1)^{\mathrm{T}}$.

In Tables 4 and 5, we compare the following preconditioners on this test problem:

- *mixed*, the $\boldsymbol{P}_{mix}$ preconditioner (4.5) where we apply the EBE preconditioner to the n_e 'finite' elements and SBS to the rank-one term,
- the EBE preconditioner applied to the complete set of $n_e + 1$ elements,
- diag, a diagonal preconditioner, and
- band preconditioner with semi-bandwidths of 1 (band(1)) and 5 (band(5)). Increasing the semi-bandwidth to larger values does not help significantly with such problems.

In Table 4, we study the effect of varying the condition number of the matrix when there is a 20 % overlap between the blocks (that is, $v_e = 10$ and $v_o = 2$)—we let $n_e = 100$, and thus $n = 802$. We report #it, the number of CG iterations, t_{con}, the time to construct the preconditioner, t_{pre}, the time spent applying the preconditioner, t_{cg}, the convergence time for the CG iteration, and t_{sol}, the total time to solve the linear system ($t_{\mathrm{sol}} = t_{\mathrm{con}} + t_{\mathrm{cg}}$).

When increasing $\lambda_{\max}$ from 10 up to 10^5, even if the condition number of the matrix slightly decreases, the eigenvalue spectrum is less clustered which makes the problem harder to solve and thus explains why the number of iterations increases for all the preconditioners. For example, when $\lambda_{\max} = 10$, there is a cluster of eigenvalues around 10^6 and all other eigenvalues are spread between 10^0 and 10^2, while when $\lambda_{\max} = 10^5$, there is still a cluster of eigenvalues around 10^6 but the other eigenvalues are spread between 10^0 and 10^5.

Firstly, we see that the mixed preconditioner is effective. When compared with the other preconditioners, it requires the smallest number of CG iterations for the three condition numbers studied. The construction time required by EBE is dramatically larger than that of the mixed preconditioner, purely because EBE has to factorize the dense matrix arising from the rank-one element using a (modified) Cholesky algorithm. Moreover, this has a significant effect on the time to apply the preconditioner, and the total computational time.

We now compare mixed with the second best preconditioner for each of the three condition numbers studied (the diagonal preconditioner for the two values of $\lambda_{\max}$ (10 and 10^3) and

Table 4. Matrix with $n_e = 100$, $v_e = 10$, $v_o = 2$ and $n = 802$. The blocks are generated with eigenvalues in the range $[1, \lambda_{\max}]$. t_{con} is the time of construction of the preconditioner, t_{pre} the time spent applying the preconditioner, t_{cg} the time of convergence of the CG method, t_{sol} the total time of the linear solver and #it the number of iterations needed for converging. κ is the condition number of the matrix

Preconditioner	t_{con}	t_{pre}	t_{cg}	t_{sol}	#it
$\lambda_{\max} = 10, \kappa = 1.6 \times 10^6$					
mixed	0.2	0.1	1.3	**1.5**	13
EBE	16.7	2.9	4.9	21.6	27
diag	0.0	0.1	18.6	18.6	244
band(1)	0.1	1.4	108.7	108.9	1 412
band(5)	0.1	1.5	50.7	50.8	730
$\lambda_{\max} = 10^3, \kappa = 1.3 \times 10^6$					
mixed	0.2	0.5	8.4	**8.7**	113
EBE	14.9	16.9	29.2	44.1	180
diag	0.0	0.1	23.9	23.9	354
band(1)	0.1	2.3	179.3	179.3	2 578
band(5)	0.1	1.7	60.6	60.6	873
$\lambda_{\max} = 10^5, \kappa = 1.0 \times 10^6$					
mixed	0.2	1.3	22.5	**22.7**	300
EBE	16.2	41.9	72.0	88.2	409
diag	0.0	0.2	80.3	80.3	1 031
band(1)	0.1	1.5	129.5	129.6	1 682
band(5)	0.1	1.6	60.9	61.0	886

band(5) for the remaining one (10^5)). The gain in total computational time due to the use of a mixed preconditioner is 15.5 when $\lambda_{\max} = 10$, and 2.7 for the other two values of $\lambda_{\max}$. We should note that the number of iterations for the mixed preconditioner increases as the conditioning worsens, so we cannot claim that the method is perfect.

In Table 5, we compare the effect of varying the degree of overlap between the blocks from 0% to 50%, while keeping the condition number of the blocks fixed at 10^5. Once again, the mixed preconditioner proves to be the best in terms of number of iterations and total computational time. The mixed preconditioner appears to be consistently about 2.7 times faster than band(5).

4.2. Tests on some optimization problems

We now turn to less contrived examples which arise from optimization. Given f and $\boldsymbol{c}$, we apply a truncated Newton algorithm (see [5]) to minimize an augmented Lagrangian function of the form (4.2) for fixed values of the Lagrange multiplier estimates $\boldsymbol{\lambda} = 0$ and increasing values of ρ. The truncated Newton equations are solved using preconditioned conjugate gradients, and we consider diagonal, EBE and mixed preconditioners; in the mixed preconditioner, the terms $\sum_{i=1}^{m} \nabla_x c_i(\boldsymbol{x})(\nabla_x c_i(\boldsymbol{x}))^{\mathrm{T}}$ of (4.3) are approximated using the SBS method, while the remainder is handled using EBE factors.

In Tables 6–8, we show results for three different problems from the CUTE collection. For each preconditioner, we report i_{new} the number of iterations of the truncated Newton algorithm, i_{cg} the total number of CG iterations required for the solution, t_{new} the total

Table 5. Matrices with $n_e = 100$, $v_e = 10$ and $v_o = 0$ to 5. $\lambda_{max} = 10^5$ and κ gives the condition number of the problem

Over-lap	order	κ	mixed		EBE		diag		band (1)		band (5)	
			#it	t_{sol}	#it	t_{sol}	#it	t_{sol}	#it	t_{sol}	#it	t_{sol}
0	1 000	3.3E6	565	**64.2**	865	259.1	1723	179.6	3 783	398.5	1 726	182.8
1	901	2.1E6	420	**39.7**	634	165.8	1 257	107.6	2 249	192.2	1 183	101.6
2	802	1.0E6	300	**22.7**	409	88.2	1 031	80.3	1 682	129.6	886	61.0
3	703	3.9E5	207	**11.8**	278	45.0	783	41.2	1 010	53.2	602	32.3
4	604	9.9E4	139	**6.2**	183	23.3	483	19.2	743	29.7	410	16.6
5	505	2.4E4	99	**3.3**	126	11.9	345	9.9	454	13.0	304	8.9

Table 6. Solution of the problem STEENBRA with different penalty values

Preconditioner	$\rho = 1$						$\rho = 10$					
	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	7	86	**2.0**	1.7	0.0	1.7	8	96	**2.2**	2.0	0.0	2.0
EBE	8	154	8.5	8.3	1.0	7.3	9	194	10.6	10.4	1.1	9.3
mixed	7	38	3.9	2.8	0.8	2.0	8	37	4.0	2.9	1.0	1.9
Preconditioner	$\rho = 100$						$\rho = 1000$					
	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	8	89	**2.1**	1.8	0.0	1.8	9	115	**2.6**	2.4	0.0	2.4
EBE	9	159	8.9	8.7	1.1	7.6	10	219	12.0	11.8	1.2	10.5
mixed	8	38	4.1	3.0	0.9	2.0	9	40	4.5	3.1	1.0	2.1

computational time for the Newton process, t_{sol} the total time required by the linear solver, t_{con} the construction time of the preconditioner and t_{cg} the convergence time for the CG method.

The penalty parameter ρ is varied from 1 to 10^3, which increases the condition number of the Hessian. As before, the number of CG iterations is always least when using the mixed preconditioner. For the problem STEENBRA (Table 6), the diagonal preconditioner is marginally the most efficient in terms of computational time for all four penalty parameters considered, while the mixed preconditioner is significantly faster than EBE. For GRIDNETA (Table 7), the mixed preconditioner is always the best preconditioner when considering the time spent in the conjugate gradient iteration and requires slightly fewer Newton iterations than other preconditioners when ρ is greater than 10. The gains arising from the use of the mixed preconditioner increase with the penalty parameter value. Finally, for GENHS28 (Table 8), the mixed preconditioner proves to be fastest for all but the smallest ρ.

Table 7. Solution of the problem GRIDNETA with different penalty values

	$\rho = 1$						$\rho = 10$					
Preconditioner	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	6	194	**17.2**	16.6	0.0	16.6	7	592	51.6	50.9	0.0	50.9
EBE	6	153	35.1	34.5	2.1	32.4	7	446	98.4	97.7	2.4	95.2
mixed	6	58	18.1	15.3	2.1	13.3	7	173	**44.8**	41.6	2.4	39.2
	$\rho = 100$						$\rho = 1000$					
Preconditioner	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	8	1 209	105.0	104.2	0.0	104.2	9	1790	153.4	153.4	0.0	153.4
EBE	8	1 018	223.7	222.9	2.7	220.2	9	1680	368.7	364.8	3.1	364.8
mixed	7	271	**67.1**	63.9	2.4	61.5	8	536	**130.0**	126.4	2.7	123.7

Table 8. Solution of the problem GENHS28 with different penalty values

	$\rho = 1$						$\rho = 10$					
Preconditioner	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	6	46	**12.7**	10.9	0.0	10.9	8	103	26.0	23.8	0.0	23.8
EBE	6	19	19.3	17.5	6.1	11.4	7	25	24.0	22.0	7.1	14.9
mixed	5	12	17.1	12.4	5.1	7.2	5	9	**15.6**	10.6	5.2	5.4
	$\rho = 100$						$\rho = 1000$					
Preconditioner	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}	i_{new}	i_{cg}	t_{new}	t_{sol}	t_{con}	t_{cg}
diag	8	116	29.2	27.0	0.0	27.0	10	143	35.3	32.7	0.0	32.7
EBE	8	37	31.7	29.5	8.0	21.4	9	42	35.8	33.4	8.9	24.5
mixed	6	13	**19.2**	13.7	6.0	7.7	7	12	**21.0**	14.4	7.1	7.4

5. Conclusions

The subspace-by-subspace preconditioner we have described allows us to take advantage of low-rank terms in an unassembled matrix. The preliminary results we have reported demonstrate that the approach has some promise in comparison with a number of currently popular preconditioners such as band or EBE methods. Mixing SBS and EBE preconditioners proves to be attractive for problems where some, but not all, elements have low rank.

However, we recognize that the tests are at best an attempt to demonstrate the viability of the approach, and that there are a number of unresolved issues, such as improvements to the amalgamation algorithm and a complete understanding of the effects that reordering the elements has on the quality of the preconditioner. More importantly, it is not known what effect such preconditioners have on the spectrum of the preconditioned matrix. We intend to consider these and other issues in due course.

Acknowledgements

The authors are very grateful to Iain Duff, Daniel Ruiz and three anonymous referees for their help in reading and improving early versions of this paper.

REFERENCES

1. Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.
2. I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: constrained and unconstrained testing environment. *ACM Trans. Math. Soft.*, **21**(1), 123–160, 1995.
3. P. Concus, G. H. Golub and D. P. O'Leary. Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method. In J. Bunch and D. Rose, editors. *Sparse Matrix Computations*, pages 309–332. Academic Press, London and New York, 1976.
4. A. R. Conn, N. I. M. Gould and Ph. L. Toint. LANCELOT*: a Fortran Package for Large-scale Nonlinear Optimization (Release A)*. Springer Series in Computational Mathematics, **17**. Springer Verlag, Heidelberg, Berlin, New York, 1992.
5. M. Daydé, J. P. Décamps, J.-Y. L'Excellent and N. I. M. Gould. Solution of large scale partially separable unconstrained optimization problems using element-by-element preconditioners. In *Proceedings of NAFEMS World Congress 97*, Vol. 2, NAFEMS Ltd, Glasgow, UK, pages 942–953, 1997.
6. M. J. Daydé, J.-Y. L'Excellent and N. I. M. Gould. Element-by-element preconditioners for large partially separable optimization problems. *SIAM Journal on Scientific Computing*, **18**(6), 1767–1787, 1997.
7. M. J. Daydé, J. P. Décamps and N. I. M. Gould. Subspace-by-subspace preconditioners for structured linear systems. ENSEEIHT-IRIT Technical Report, RT/APO/98/2, 1998.
8. I. S. Duff. The solution of augmented systems. In D. F. Griffiths and G. A. Watson, editors. *Numerical Analysis 1993*, pages 40–55, Longman Scientific and Technical, Harlow, England, 1994.
9. I. S. Duff, R. G. Grimes and J. G. Lewis. Users' guide for the Harwell–Boeing sparse matrix collection (Release 1). Technical Report RAL-92-086, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1992.
10. J. Erhel, A. Traynard and M. Vidrascu. An element-by-element preconditioned conjugate gradient method implemented on a vector computer. *Parallel Comput.*, **17**, 1051–1065, 1991.
11. P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comput.*, **28**, 505–535, 1974.

12. D. Goldfarb and S. Wang. Partial-update Newton methods for unary, factorable, and partially separable optimization. *SIAM J. Optimiz.*, **3**(2), 382–397, 1993.
13. A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor. *Nonlinear Optimization 1981*, pages 301–312. Academic Press, London and New York, 1982.
14. M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bureau Stand.*, **49**, 409–436, 1952.
15. T. J. R. Hughes, R. M. Ferencz and J. O. Hallquits. Large-scale vectorized implicit calculations in solid mechanics on a CRAY X-MP/48 utilizing EBE preconditioned conjugate gradients. *Comput. Meth. Appl. Mech. Eng.*, **61**, 215–248, 1987.
16. T. J. R Hughes, I. Levit and J. Winget. An element-by-element solution algorithm for problems of structural and solid mechanics. *Comput. Meth. App. Mech. Eng.*, **36**, 241–254, 1983.
17. J.-Y. L'Excellent. Utilisationde préconditionneurs élément-par-élément pour la résolution de problèmes d'optimisationde grande taille. *PhD Thesis, Institut National Polytechnique de Toulouse*, 1995.
18. P. Matstoms. Sparse QR factorization in MATLAB. *ACM Trans. Math. Soft.*, **20**(1), 136–159, 1994.
19. M. Ortiz, P. M. Pinsky and R. L. Taylor. Unconditionally stable element-by-element algorithms for dynamic problems. *Comput. Meth. Appl. Mech. Eng.*, **36**, 223–239, 1983.
20. B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

