RAL-TR-98-005

Subspace-by-Subspace Preconditioners for Structured Linear Systems

Michel J. Daydé[1], Jerome P. Décamps[1] and Nicholas I. M. Gould[2,3]

ABSTRACT

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.

---

[1] ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse CEDEX, France, EU
Email : dayde@enseeiht.fr and decamps@enseeiht.fr

[2] Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU
Email : n.gould@rl.ac.uk

[3] Current reports available by anonymous ftp from joyous-gard.cc.rl.ac.uk (internet 130.246.9.91) in the directory "pub/reports".

Department for Computation and Information
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

January 20, 1998.

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

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

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

1. Normal equations for least squares (see, for instance, Björck, 1996).
2. The Schur complement following partial elimination in augmented systems (see, for example, Duff, 1994).
3. Newton equations for partially separable optimization of unary functions (see Goldfarb and Wang, 1993).
4. More general partially separable optimization (see Griewank and Toint, 1982).

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, Hestenes and Stiefel, 1952, and Concus, Golub and O'Leary, 1976) 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, $a_i a_i^T$. 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.

SBS preconditioners will never be well-suited to all problems, primarily because of their design aims. In particular, they are unlikely to be appropriate for those problems whose A_i are (close to) full rank. In Section 4 we consider matrices for which some, but not all, terms are of the form $\boldsymbol{A}_i \boldsymbol{A}_i^T$. We demonstrate that one of the great advantage of SBS preconditioners is that they can efficiently be combined with other Element-by-Element preconditioners to handle substructures of low rank. This then suggests 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 $\boldsymbol{E}_i = \boldsymbol{A}_i \boldsymbol{A}_i^T$, 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 $\boldsymbol{E}_i = \boldsymbol{A}_i \boldsymbol{A}_i^T$. 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, Hughes, Levit and Winget, 1983, and Ortiz, Pinsky and Taylor, 1983). These have been seen to be effective for systems arising from partial differential equations (Hughes, Ferencz and Hallquits, 1987, and Erhel, Traynard and Vidrascu, 1991) and optimization (Daydé, Décamps, L'Excellent and Gould, 1997*a*)

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 individual $\boldsymbol{S}_i$ are small or zero (this is likely to be true if $\boldsymbol{E}_i$ is very diagonal 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 (see eg, Theorem 5.3 in Chapter 5 of L'Excellent, 1995) and thus has a Cholesky factorization

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

The matrix $\boldsymbol{W}_i$ is known as the *Winget decomposition* of $\boldsymbol{E}_i$ (see Hughes et al., 1983). Notice that if $\boldsymbol{E}_i$ has nonzeros 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

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

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

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

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

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

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 $\boldsymbol{P}_{EBE}\boldsymbol{x} = \boldsymbol{y}$ 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 backsolves 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. Daydé, L'Excellent and Gould (1997*b*) demonstrate 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 nonzeros in e_i rows and columns — E_i might 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 nonzero rows and columns of $\boldsymbol{E}_i$ by $\boldsymbol{E}_i^{\mathcal{E}_i}$, and making similar definitions for $\boldsymbol{W}_i$, $\boldsymbol{D}_i$ and $\boldsymbol{D}$, we obtain

$$\boldsymbol{W}_i^{\mathcal{E}_i} = \boldsymbol{I}_{e_i} + (\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}} (\boldsymbol{E}_i^{\mathcal{E}_i} - \boldsymbol{D}_i^{\mathcal{E}_i}) (\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}} = \boldsymbol{\Delta}_i^{\mathcal{E}_i} + (\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}} \boldsymbol{E}_i^{\mathcal{E}_i} (\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}}, \tag{2.9}$$

where $\boldsymbol{\Delta}_i^{\mathcal{E}_i} = \boldsymbol{I}_{e_i} - (\boldsymbol{D}^{\mathcal{E}_i})^{-1} \boldsymbol{D}_i^{\mathcal{E}_i} \geq \boldsymbol{0}$. We suppose, for now, that $\boldsymbol{\Delta}_i^{\mathcal{E}_i} > \boldsymbol{0}$, but will shortly return to the singular case.

Now suppose that $\boldsymbol{E}_i$ is of rank r_i. Then we see immediately that $\boldsymbol{W}_i^{\mathcal{E}_i}$ is a rank-r_i modification of the positive definite diagonal matrix $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$. Thus, if $r_i < e_i$ it would seem to be preferable to *update* the Cholesky factors of $\boldsymbol{W}_i^{\mathcal{E}_i}$ following a sequence of rank-r_i modifications rather than assembling and factoring $\boldsymbol{W}_i^{\mathcal{E}_i}$ directly (see, for example Gill, Golub, Murray and Saunders, 1974). 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 $\boldsymbol{B}_i^{\mathcal{E}_i} = (\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{-\frac{1}{2}}(\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}}\boldsymbol{E}_i^{\mathcal{E}_i}(\boldsymbol{D}^{\mathcal{E}_i})^{-\frac{1}{2}}(\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{-\frac{1}{2}}$ be a rescaling of $\boldsymbol{E}_i^{\mathcal{E}_i}$. Then we may write (2.9) as

$$\boldsymbol{W}_i^{\mathcal{E}_i} = (\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}}\left(\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathcal{E}_i}\right)(\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}}. \tag{2.10}$$

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

$$\boldsymbol{B}_i^{\mathcal{E}_i} = \boldsymbol{Q}_i^{\mathcal{I}_i}\begin{pmatrix} \boldsymbol{B}_i^{\mathcal{I}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}(\boldsymbol{Q}_i^{\mathcal{I}_i})^T = (\boldsymbol{Y}_i^{\mathcal{I}_i} \;\; \boldsymbol{Z}_i^{\mathcal{I}_i})\begin{pmatrix} \boldsymbol{B}_i^{\mathcal{I}_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} (\boldsymbol{Y}_i^{\mathcal{I}_i})^T \\ (\boldsymbol{Z}_i^{\mathcal{I}_i})^T \end{pmatrix} = \boldsymbol{Y}_i^{\mathcal{I}_i}\boldsymbol{B}_i^{\mathcal{I}_i}(\boldsymbol{Y}_i^{\mathcal{I}_i})^T, \tag{2.11}$$

where $\boldsymbol{Q}_i^{\mathcal{I}_i}$ is orthogonal and defines a transformation from the elemental to an *internal* representation of $\boldsymbol{B}_i^{\mathcal{E}_i}$, and this representation, $\boldsymbol{B}_i^{\mathcal{I}_i}$, is an r_i by r_i symmetric positive definite matrix (for instance, but not restricted to, tridiagonal, see Parlett, 1980, Chapter 7). Furthermore, let $\boldsymbol{L}_i^{\mathcal{I}_i}$ be the Cholesky factor of $\boldsymbol{I}_{e_i} + \boldsymbol{B}_i^{\mathcal{I}_i}$. Then

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

where

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

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

$$\boldsymbol{W}_i^{\mathcal{E}_i} = (\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}}\boldsymbol{M}_i^{\mathcal{I}_i}(\boldsymbol{M}_i^{\mathcal{I}_i})^T(\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}}. \tag{2.14}$$

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

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

where $\boldsymbol{\Delta}_i$ and $\boldsymbol{M}_i$ are simply $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$ and $\boldsymbol{M}_i^{\mathcal{I}_i}$ 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 $\boldsymbol{M}_i^{\mathcal{I}_i}$ on the subspaces defined by the matrices $\boldsymbol{Y}_i^{\mathcal{I}_i}$ and $\boldsymbol{Z}_i^{\mathcal{I}_i}$.

At a first glance, we do not appeared 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 $(\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}}\boldsymbol{M}_i^{\mathcal{I}_i}\boldsymbol{x}^{\mathcal{I}_i} = \boldsymbol{y}^{\mathcal{I}_i}$. Using the orthogonality and partitioning of $\boldsymbol{Q}_i^{\mathcal{I}_i}$ and (2.13), we have that

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

The matrix $\boldsymbol{Q}_i^{\mathcal{I}_i}$ is not required, merely its first r_i columns $\boldsymbol{Y}_i^{\mathcal{I}_i}$. 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 $2r_ie_i + \frac{1}{2}r_i^2 \leq \frac{1}{2}e_i^2$, that is whenever

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

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

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

and the principal costs are identical to the previous case.

We now consider how to cope with the possibility that $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$ may be singular. Notice that $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$ 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 known as static condensation in finite element terminology); 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 $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$ 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.

2.3 Subspace-by-Subspace preconditioners for structured problems

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

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

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

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

where $\boldsymbol{Q}_i^{\mathcal{I}_i}$ and $\boldsymbol{V}_i^{\mathcal{I}_i}$ are orthogonal and $\boldsymbol{R}_i^{\mathcal{I}_i}$ is triangular and of rank r_i, be a complete orthogonal decomposition of $\boldsymbol{C}_i^{\mathcal{E}_i}$ (see, for instance, Björck, 1996). Then clearly, $\boldsymbol{B}_i^{\mathcal{I}_i} = \boldsymbol{R}_i^{\mathcal{I}_i} (\boldsymbol{R}_i^{\mathcal{I}_i})^T$, 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 $\boldsymbol{Y}_i^{\mathcal{I}_i}$ or $\boldsymbol{Q}_i^{\mathcal{I}_i}$, 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 $\boldsymbol{C}_i^{\mathcal{E}_i}$, and a singular-value decomposition may be preferred. Again, see, Björck (1996) for details.

A potential difficulty occurs when $\boldsymbol{\Delta}_i^{\mathcal{E}_i}$ 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 $\boldsymbol{E}_i^{\mathcal{E}_i}$. We may then find an orthogonal matrix $\boldsymbol{U}_i^{\mathcal{E}_i}$ so that

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

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

$$\begin{aligned} \boldsymbol{W}_i^{\mathcal{E}_i} &= \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \overline{\boldsymbol{\Delta}}_i^{\mathcal{E}_i} \end{pmatrix} + \begin{pmatrix} \boldsymbol{R}_i^{\mathcal{E}_i} & \boldsymbol{0} \\ \widehat{\boldsymbol{A}}_i^{\mathcal{E}_i} & \overline{\boldsymbol{A}}_i^{\mathcal{E}_i} \end{pmatrix} \begin{pmatrix} (\boldsymbol{R}_i^{\mathcal{E}_i})^T & (\widehat{\boldsymbol{A}}_i^{\mathcal{E}_i})^T \\ \boldsymbol{0} & (\overline{\boldsymbol{A}}_i^{\mathcal{E}_i})^T \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{R}_i^{\mathcal{E}_i} (\boldsymbol{R}_i^{\mathcal{E}_i})^T & \boldsymbol{R}_i^{\mathcal{E}_i} (\widehat{\boldsymbol{A}}_i^{\mathcal{E}_i})^T \\ \widehat{\boldsymbol{A}}_i^{\mathcal{E}_i} (\boldsymbol{R}_i^{\mathcal{E}_i})^T & \overline{\boldsymbol{\Delta}}_i^{\mathcal{E}_i} + \widehat{\boldsymbol{A}}_i^{\mathcal{E}_i} (\widehat{\boldsymbol{A}}_i^{\mathcal{E}_i})^T + \overline{\boldsymbol{A}}_i^{\mathcal{E}_i} (\overline{\boldsymbol{A}}_i^{\mathcal{E}_i})^T \end{pmatrix}. \end{aligned} \tag{2.23}$$

Eliminating the first k elemental variables then leaves the Schur complement $\overline{\boldsymbol{\Delta}}_i^{\mathcal{E}_i} + \overline{\boldsymbol{A}}_i^{\mathcal{E}_i} (\overline{\boldsymbol{A}}_i^{\mathcal{E}_i})^T$, which is of the form (2.9), but now with $\overline{\boldsymbol{\Delta}}_i^{\mathcal{E}_i}$ non singular. Notice, the matrix $\boldsymbol{U}_i^{\mathcal{E}_i}$ 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 $\overline{\boldsymbol{A}}_i^{\mathcal{E}_i}$ contains only zeros. Thus the variable associated with this row is no longer involved in the i-th 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 a m by n rectangular matrix with $m > n$. A solution to (3.1) satisfies the normal equations

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

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

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

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 $\boldsymbol{A}_i = \boldsymbol{a}_i^T$ and $e = m$, where $\boldsymbol{a}_i^T$ is the i-th 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, Björck, 1996, 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 $\boldsymbol{A}_i = \boldsymbol{a}_i^T$ 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 $\mathcal{R}$ and variables $\mathcal{C}$ to be directly eliminated. We now scan the rows and columns of $\boldsymbol{A}$ not in $\mathcal{R}$ and $\mathcal{C}$, respectively, for a column singleton. If one is found, we add its index to $\mathcal{C}$, add the row to $\mathcal{R}$ and repeat the search. If none are found, all remaining columns have two or more nonzero 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 of $\boldsymbol{A}$ so that

$$\boldsymbol{P}\boldsymbol{A}\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{A}_e \\ \boldsymbol{0} & \boldsymbol{A}_r \end{pmatrix},$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are permutation matrices and each column of $\boldsymbol{A}_r$ has at least 2 nonzero entries. Substituting in (3.4) and simplifying reveals that if we solve

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

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

then

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

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 nonzeros and hence choosing $\boldsymbol{A}_i = \boldsymbol{a}_i^T$ 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 3.1 to merge rank-one terms. It basically regroups rank-one terms to ensure that no variable belongs to a single element.

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

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 3.1 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.

3.2 Numerical experiments

We have tested the SBS preconditioner on a number of rectangular matrices from the Harwell-Boeing collection (see Duff, Grimes and Lewis, 1992), and on a number of Jacobian matrices from problems arising from the CUTE collection (see Bongartz, Conn, Gould and Toint, 1995). The characteristics of these problems are summarized in Table 3.1. We include details of how many exposed variables can be trivially removed, and the resulting problem sizes.

Name	Set.	m	n	nz	n_e	n_s	κ
econ1	HWB1	277	207	2909	49	158	7.4×10^5
econ2		260	207	2942	35	172	1.8×10^{16}
econ3		260	207	2948	27	180	7.0×10^{15}
abb313	HWB2	313	176	1557	0	176	1.6×10^1
ash219		219	85	438	0	85	7.8×10^0
ash331		331	104	662	0	104	5.2×10^0
ash608		608	188	1216	0	188	6.3×10^0
ash958		958	292	1916	0	292	6.9×10^0
wl1033		1033	320	4732	12	308	1.7×10^2
wl1850		1850	712	8758	7	705	1.1×10^2
il1033		1033	320	4732	12	308	1.9×10^4
il1850		1850	712	8758	7	705	1.4×10^3
af1252		1252	320	5170	12	308	3.6×10^1
af1641		1641	320	5948	9	311	4.2×10^0
abb313b	HWB3	313	176	1557	0	176	1.7×10^7
ash219b		219	85	438	0	85	7.1×10^6
ash331b		331	104	662	0	104	5.4×10^6
ash608b		608	188	1216	0	188	6.4×10^6
ash958b		958	292	1916	0	292	6.9×10^6
wl1033b		1033	320	4732	12	308	1.1×10^8
wl1850b		1850	712	8758	7	705	2.0×10^7
il1033b		1033	320	4732	12	308	3.1×10^9
il1850b		1850	712	8758	7	705	1.3×10^9
af1252b		1252	320	5170	12	308	2.7×10^7
af1641b		1641	320	5948	9	311	2.4×10^6
BRATU1D	SIF1	4007	3004	6007	1	3003	6.5×10^5
BROYDN3D		1000	1000	2998	0	1000	8.0×10^{15}
HAGER1		2001	1000	3000	0	1000	5.8×10^2
SPMSQRT		1664	1000	2996	0	1000	2.1×10^1
TRIDIA		2000	1000	2998	1	999	7.1×10^3
BROWNBS	SIF2	3997	2997	6993	0	2997	1.4×10^6
DQDRTIC		3994	2994	5988	0	2994	5.6×10^9
DQRTIC		2000	1000	1999	1	999	5.6×10^9
EDENSCH		3998	2998	6994	1	2997	1.8×10^3
EXPFITC		1009	502	2755	0	502	7.3×10^2
FLETCHBV		4001	3001	7001	0	3001	3.0×10^6
GENHS28		1000	998	2994	0	998	3.7×10^0
HYDCAR20		99	99	734	0	99	1.0×10^6
MAXLIKA		243	235	2003	0	235	4.2×10^1
NONDQUAR		2000	1000	3998	0	1000	3.2×10^1
ORTHREGA	SIF3	517	256	1792	0	256	2.8×10^2
ORTHREGC		1005	500	3500	0	500	2.0×10^2
ORTHREGD		1003	500	2498	0	500	5.3×10^1
ORTHREGF		680	225	1770	0	225	5.0×10^2
QUARTC		2000	1000	1999	1	999	5.6×10^9
SEMICON1		1002	1000	3000	0	1000	1.1×10^1
TFI1		2005	1001	5004	0	1001	5.3×10^1
TFI2		2005	1001	5003	0	1001	2.7×10^1
TFI3		2005	1001	5003	0	1001	2.7×10^1

Table 3.1: 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. The column κ gives the condition number of each matrix.

The set HWB1 is made up from Harwell-Boeing matrices. The `econ` problems arise in economic analysis. The SIF1 problems are chosen arbitrarily from the CUTE collection, while the sets SIF2 and SIF3 have been chosen from the same collection to exhibit interesting behavior when using the *SBS* preconditioners. The HWB2 and HWB3 problems were derived from Harwell-Boeing matrices by Matstoms (1994), and subsequently used in tests by Björck (1996). The matrices in the HWB3 set are the same as those in HBW2 except that the rows $n-1$ to m are multiplied by a scaling factor 2^{-20}. This gives matrices with large condition numbers for the HWB3 set. The `abb` and `ash` matrices have random numbers uniformly distributed in $[-1, 1]$. The matrices `af1252` and `af1641` are artificial, constructed, respectively, using `wl1033` and `ash219`, and `wl1033` and `ash608` as

$$A = \begin{pmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{pmatrix}$$

with $\boldsymbol{A}_1 =$ `wlxxx` and $\boldsymbol{A}_2 = \begin{pmatrix} 0 & \texttt{ashyyy} \end{pmatrix}$.

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 $\boldsymbol{x}^* = (1, \ldots, 1)^T$, and setting $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^*$. The origin is taken as the initial estimate of the solution.

In Tables 3.2–3.5, we compare the conjugate gradient (CG) solution of the normal equations without preconditioning, to the same method preconditioned either by a band approximation—in our case *band*(1), a tridiagonal matrix whose nonzeros are those from $\boldsymbol{A}^T\boldsymbol{A}$—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 3.1. We use the variant of the CG method for least-squares problems described by Björck (1996, Section 7.4.1). Convergence is recorded as soon as

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

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 49 problems tested, the unpreconditioned method is best (in terms of total computational time) in 14 cases, band in 15 cases, and SBS in 17 cases. For 4 problems (problems `wl1033b`, `il1033b`, `il1850b` and `af1252b`), all the preconditioners fail to converge within the permitted number of iterations. On one problem (`ash331`), no preconditioning and the band preconditioner give the same computational time.

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 two problems (`HAGER1` and `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 problem `af1641b` is the only one for which no preconditioning requires fewer iterations than with the SBS preconditioner (with $k_{max} = 1$). 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.

The impact of increasing k_{max} depends on the structure of the problems. There is little effect on problems `econ1`, `econ2` and `econ3` for example, while the number of iterations settles down to 2 as k_{max} increases from 1 to 5 (and beyond) on the `ORTHREG` set of problems. In some cases, increasing k_{max} increases the number of iterations, as we see for `HYDCAR20` or less significantly for `wl1033` and `wl1830`. A small value of k_{max}, say $k_{max} = 5$ or 10, seems to be a good choice in our tests.

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.

name		no	band(1)	SBS(k_{max}), k_{max} =						
				1	5	10	20	30	40	50
econ1	#it	194	111	48	47	47	49	49	49	49
	t_{con}	0.00	**0.00**	0.02	0.00	0.03	0.05	0.05	0.07	0.07
	t_{cg}	0.18	**0.13**	0.32	0.18	0.20	0.22	0.27	0.27	0.28
	err	0.1E-10	0.5E-11	0.3E-13	0.1E-11	0.3E-11	0.4E-12	0.5E-12	0.9E-12	0.9E-12
econ2	#it	262	109	46	49	52	52	57	52	51
	t_{con}	0.00	**0.02**	0.02	0.02	0.02	0.05	0.05	0.10	0.10
	t_{cg}	0.25	**0.12**	0.30	0.22	0.23	0.25	0.32	0.35	0.35
	err	0.8E-13	0.2E-13	0.2E-13	0.8E-14	0.2E-13	0.1E-13	0.1E-13	0.1E-13	0.2E-13
econ3	#it	297	109	45	48	49	49	52	48	49
	t_{con}	0.00	**0.00**	0.02	0.03	0.03	0.05	0.07	0.12	0.12
	t_{cg}	0.32	**0.13**	0.30	0.20	0.22	0.27	0.32	0.33	0.37
	err	0.3E-13	0.8E-14	0.6E-14	0.5E-14	0.1E-13	0.6E-14	0.8E-14	0.8E-14	0.5E-14
abb313	#it	123	76	36	36	35	35	35	35	35
	t_{con}	0.00	**0.02**	0.02	0.02	0.02	0.02	0.03	0.03	0.02
	t_{cg}	0.12	**0.07**	0.27	0.15	0.15	0.15	0.13	0.12	0.15
	err	0.4E-14	0.2E-14	0.1E-14	0.1E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14
ash219	#it	70	35	18	18	18	16	16	16	16
	t_{con}	0.00	**0.00**	0.00	0.02	0.02	0.00	0.02	0.03	0.02
	t_{cg}	0.05	**0.03**	0.08	0.03	0.03	0.05	0.03	0.03	0.03
	err	0.1E-14	0.6E-15	0.1E-14	0.1E-14	0.5E-15	0.2E-14	0.2E-14	0.2E-14	0.2E-14
ash331	#it	58	35	17	17	16	17	17	17	17
	t_{con}	**0.00**	**0.00**	0.02	0.02	0.02	0.03	0.03	0.05	0.03
	t_{cg}	**0.03**	**0.03**	0.12	0.05	0.05	0.07	0.05	0.05	0.07
	err	0.2E-14	0.1E-14	0.1E-14	0.5E-15	0.7E-15	0.3E-15	0.4E-15	0.4E-15	0.4E-15
ash608	#it	73	38	18	18	18	17	17	17	18
	t_{con}	0.02	**0.00**	0.02	0.02	0.05	0.05	0.08	0.10	0.10
	t_{cg}	0.05	**0.03**	0.22	0.10	0.10	0.10	0.12	0.13	0.13
	err	0.2E-14	0.2E-14	0.4E-15	0.4E-15	0.1E-14	0.2E-14	0.5E-15	0.1E-14	0.3E-15
ash958	#it	80	39	18	18	17	16	16	16	16
	t_{con}	0.00	**0.00**	0.05	0.03	0.07	0.08	0.12	0.17	0.18
	t_{cg}	0.12	**0.07**	0.35	0.17	0.13	0.15	0.18	0.22	0.25
	err	0.3E-14	0.1E-14	0.1E-14	0.1E-14	0.8E-15	0.7E-15	0.4E-15	0.5E-15	0.6E-15
wl1033	#it	222	222	123	117	117	118	111	114	114
	t_{con}	**0.00**	0.00	0.05	0.05	0.07	0.13	0.18	0.23	0.30
	t_{cg}	**0.47**	0.53	2.78	1.30	1.30	1.62	1.87	2.30	2.47
	err	0.1E-13	0.5E-14	0.3E-13	0.2E-13	0.3E-13	0.8E-14	0.4E-13	0.1E-13	0.1E-13
wl1850	#it	525	521	216	209	197	201	197	197	196
	t_{con}	**0.00**	0.02	0.17	0.15	0.18	0.22	0.33	0.40	0.45
	t_{cg}	**2.00**	2.32	9.37	4.87	4.77	5.57	6.07	7.05	7.68
	err	0.1E-13	0.2E-13	0.1E-13	0.7E-14	0.1E-13	0.1E-13	0.1E-13	0.8E-14	0.1E-13
il1033	#it	3080*	3080*	1835	1827	1736	1739	1698	1711	1640
	t_{con}	0.00	0.00	0.07	0.05	**0.07**	0.13	0.17	0.22	0.32
	t_{cg}	6.70	7.37	41.25	20.30	**19.03**	23.72	28.42	33.73	35.23
	err	0.2E-02	0.2E-02	0.3E-10	0.4E-10	0.3E-09	0.3E-09	0.1E-09	0.4E-10	0.2E-10
il1850	#it	2474	2499	868	837	820	768	791	761	748
	t_{con}	**0.02**	0.02	0.17	0.17	0.17	0.23	0.28	0.40	0.58
	t_{cg}	**9.45**	11.40	37.88	19.28	19.90	21.25	24.42	26.47	30.22
	err	0.2E-11	0.2E-12	0.6E-12	0.4E-12	0.1E-12	0.7E-11	0.3E-11	0.1E-11	0.4E-11

Table 3.2: Results with no, band(1) and SBS(k_{max}) preconditioners.

name		no	band(1)	SBS(k_{max}), k_{max} =						
				1	5	10	20	30	40	50
af1252	#it	140	99	47	47	46	44	44	44	44
	t_{con}	0.00	**0.02**	0.07	0.08	0.08	0.17	0.23	0.33	0.38
	t_{cg}	0.38	**0.27**	1.33	0.63	0.63	0.78	0.92	1.23	1.22
	err	0.2E-14	0.1E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14	0.1E-14	0.2E-14	0.7E-15
af1641	#it	60	39	18	18	19	18	18	19	19
	t_{con}	0.00	**0.00**	0.08	0.08	0.12	0.20	0.32	0.43	0.57
	t_{cg}	0.17	**0.12**	0.63	0.33	0.40	0.48	0.58	0.72	0.73
	err	0.2E-14	0.1E-14	0.1E-14	0.1E-14	0.4E-15	0.2E-14	0.1E-14	0.4E-15	0.4E-15
abb313b	#it	1457	1523	34	34	34	34	34	34	34
	t_{con}	0.00	0.00	0.02	**0.00**	0.02	0.02	**0.02**	**0.02**	0.03
	t_{cg}	1.27	1.60	0.25	**0.15**	0.15	0.15	**0.13**	**0.13**	0.13
	err	0.7E-02	0.1E-04	0.1E-08	0.8E-10	0.3E-08	0.3E-08	0.3E-08	0.3E-08	0.3E-08
ash219b	#it	559	330	18	19	14	14	14	14	14
	t_{con}	0.00	0.00	0.00	**0.00**	**0.02**	**0.02**	**0.02**	**0.02**	**0.00**
	t_{cg}	0.27	0.17	0.08	**0.05**	**0.03**	**0.03**	**0.03**	**0.03**	**0.05**
	err	0.3E-02	0.1E-04	0.1E-11	0.5E-13	0.2E-09	0.8E-10	0.1E-10	0.1E-10	0.1E-10
ash331b	#it	448	254	16	16	15	14	14	14	14
	t_{con}	0.00	0.02	0.00	0.02	0.03	**0.02**	0.03	0.03	0.03
	t_{cg}	0.23	0.17	0.10	0.07	0.03	**0.03**	0.05	0.05	0.05
	err	0.1E-02	0.1E-04	0.2E-09	0.6E-09	0.4E-10	0.5E-09	0.2E-08	0.2E-08	0.2E-08
ash608b	#it	776	470	17	18	17	15	15	15	16
	t_{con}	0.00	0.00	0.02	0.05	**0.05**	0.07	0.07	0.12	0.10
	t_{cg}	0.68	0.50	0.20	0.10	**0.08**	0.10	0.10	0.10	0.13
	err	0.3E-02	0.7E-05	0.2E-10	0.7E-12	0.4E-10	0.1E-09	0.2E-09	0.9E-09	0.2E-09
ash958b	#it	946	510	20	20	17	18	18	19	19
	t_{con}	0.00	0.02	0.05	**0.05**	**0.07**	0.08	0.13	0.15	0.20
	t_{cg}	1.27	0.83	0.37	**0.17**	**0.15**	0.18	0.20	0.27	0.27
	err	0.8E-03	0.6E-05	0.7E-13	0.2E-09	0.8E-10	0.3E-13	0.2E-11	0.9E-13	0.9E-13
well1033b	#it	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*
	t_{con}	0.00	0.00	0.07	0.07	0.05	0.12	0.18	0.25	0.32
	t_{cg}	6.67	8.45	70.42	34.43	33.83	42.75	52.60	63.48	66.48
	err	0.3E00	0.6E00	0.1E00	0.1E00	0.4E00	0.4E-01	0.2E00	0.3E00	0.2E01
well1850b	#it	7050*	7050*	3316	3850	7050*	7050*	7050*	7050*	7050*
	t_{con}	0.00	0.02	0.20	**0.15**	0.18	0.25	0.32	0.38	0.48
	t_{cg}	27.03	31.45	144.40	**90.47**	170.45	194.95	217.50	247.70	294.02
	err	0.5E-01	0.1E01	0.2E-02	0.1E-02	0.1E-01	0.3E00	0.6E00	0.6E00	0.6E00
ill1033b	#it	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*
	t_{con}	0.00	0.02	0.07	0.07	0.08	0.13	0.17	0.23	0.32
	t_{cg}	6.38	7.45	69.50	34.13	33.78	42.33	51.92	62.53	67.27
	err	0.4E00	0.2E01	0.7E01	0.8E02	0.9E02	0.3E03	0.7E02	0.3E02	0.1E04
ill1850b	#it	7050*	7050*	7050*	7050*	7050*	7050*	7050*	7050*	7050*
	t_{con}	0.00	0.02	0.18	0.15	0.18	0.27	0.30	0.40	0.50
	t_{cg}	30.35	36.12	326.92	175.68	189.50	205.83	221.12	249.97	285.65
	err	0.7E-01	0.2E02	0.4E01	0.5E01	0.1E03	0.1E04	0.1E04	0.1E04	0.1E04
af1252b	#it	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*	3080*
	t_{con}	0.00	0.00	0.08	0.05	0.07	0.13	0.23	0.30	0.37
	t_{cg}	7.67	8.43	87.83	44.00	44.75	57.80	68.73	79.80	88.75
	err	0.3E00	0.3E00	0.1E00	0.1E00	0.3E-01	0.2E-01	0.6E-01	0.4E00	0.8E-01

Table 3.3: Results with no, band(1) and SBS(k_{max}) preconditioners.

name		no	band(1)	SBS(k_{max}), k_{max} =						
				1	5	10	20	30	40	50
af1641b	#it	2386	3110*	2605	3110*	3110*	3110*	3110*	3110*	3110*
	t_{con}	**0.00**	0.02	0.08	0.10	0.12	0.18	0.30	0.45	0.55
	t_{cg}	**6.42**	9.25	91.20	55.12	54.63	68.58	85.48	102.93	117.37
	err	0.2E-03	0.6E-01	0.6E-04	0.1E-03	0.2E00	0.1E-01	0.8E00	0.2E01	0.1E01
BRATU1D	#it	25210*	1561	16	10	9	9	9	9	9
	t_{con}	0.00	0.02	0.98	**0.68**	0.73	1.02	1.37	1.78	2.20
	t_{cg}	146.27	14.02	1.30	**0.45**	0.42	0.65	0.82	0.95	1.25
	err	0.5E00	0.4E-10	0.7E-15	0.1E-11	0.7E-12	0.5E-13	0.4E-13	0.3E-13	0.3E-13
BROYDN3D	#it	98	573	36	27	27	27	27	27	27
	t_{con}	**0.00**	0.02	0.13	0.10	0.08	0.10	0.10	0.08	0.10
	t_{cg}	**0.20**	1.67	0.75	0.37	0.38	0.38	0.38	0.42	0.38
	err	0.8E-02	0.6E-01	0.5E-02	0.1E-02	0.1E-02	0.1E-02	0.1E-02	0.1E-02	0.1E-02
HAGER1	#it	758	2	416	192	126	76	56	41	34
	t_{con}	0.00	**0.00**	0.23	0.17	0.20	0.32	0.43	0.62	0.82
	t_{cg}	1.77	**0.02**	16.52	3.48	2.60	2.10	1.98	1.88	1.82
	err	0.2E-12	0.1E-14	0.9E-14	0.2E-13	0.7E-14	0.2E-13	0.5E-13	0.7E-13	0.5E-14
SPMSQRT	#it	251	219	65	58	50	50	50	50	50
	t_{con}	**0.00**	0.00	0.18	0.18	0.18	0.18	0.18	0.17	0.18
	t_{cg}	**0.88**	0.93	2.42	1.22	0.93	0.97	1.08	0.98	0.98
	err	0.2E-13	0.1E-13	0.6E-14	0.7E-14	0.1E-13	0.1E-13	0.1E-13	0.1E-13	0.1E-13
TRIDIA	#it	1810	1	10	8	6	6	5	5	5
	t_{con}	0.00	**0.00**	0.23	0.17	0.20	0.30	0.47	0.58	0.80
	t_{cg}	4.43	**0.00**	0.38	0.17	0.13	0.17	0.18	0.23	0.27
	err	0.2E-14	0.2E-15	0.1E-15	0.4E-15	0.3E-13	0.1E-15	0.1E-15	0.1E-15	0.1E-15
BROWNBS	#it	10	10	14	7	5	3	3	3	3
	t_{con}	**0.02**	0.02	1.25	0.98	1.05	1.22	1.70	2.28	2.90
	t_{cg}	**0.08**	0.12	1.32	0.37	0.32	0.30	0.38	0.48	0.58
	err	0.5E-03	0.1E-03	0.5E-09	0.1E-08	0.5E-10	0.9E-09	0.3E-10	0.4E-10	0.2E-10
DQDRTIC	#it	11	5	1	1	1	1	1	1	1
	t_{con}	0.00	**0.00**	1.27	0.92	1.00	1.20	1.57	2.08	2.63
	t_{cg}	0.10	**0.08**	0.12	0.08	0.12	0.12	0.15	0.20	0.25
	err	0.6E-15	0.2E-13	0.4E-13	0.4E-13	0.4E-13	0.4E-13	0.4E-13	0.4E-13	0.4E-13
DQRTIC	#it	9990*	2834	2	2	2	2	2	2	2
	t_{con}	0.00	0.02	0.22	**0.18**	0.20	0.28	0.47	0.62	0.78
	t_{cg}	21.47	9.02	0.08	**0.05**	0.05	0.08	0.08	0.12	0.15
	err	0.2E00	0.6E-01	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10
EDENSCH	#it	106	42	13	7	5	3	3	3	3
	t_{con}	0.00	**0.02**	1.27	0.87	0.95	1.22	1.58	2.05	2.70
	t_{cg}	0.68	**0.42**	1.12	0.37	0.28	0.28	0.33	0.45	0.57
	err	0.4E-12	0.5E-13	0.2E-13	0.2E-14	0.1E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14
EXPFITC	#it	843	243	26	2	2	2	2	2	2
	t_{con}	0.00	0.03	0.07	**0.07**	**0.07**	0.17	0.28	0.43	0.58
	t_{cg}	1.25	0.48	0.52	**0.03**	**0.03**	0.05	0.05	0.07	0.10
	err	0.3E-10	0.2E-10	0.6E-12	0.6E-14	0.1E-13	0.1E-13	0.3E-13	0.1E-13	0.1E-13
FLETCHBV	#it	4466	24	3	2	2	2	2	2	2
	t_{con}	0.00	**0.02**	1.18	0.95	1.00	1.25	1.78	2.68	4.00
	t_{cg}	35.18	**0.27**	0.28	0.12	0.17	0.22	0.25	0.32	0.53
	err	0.2E-11	0.2E-11	0.2E-11	0.2E-11	0.2E-11	0.1E-11	0.1E-11	0.1E-11	0.2E-11

Table 3.4: Results with no, band(1) and SBS(k_{max}) preconditioners.

name		no	band(1)	SBS(k_{max}), k_{max} =						
				1	5	10	20	30	40	50
GENHS28	#it	52	635	19	16	16	16	16	16	16
	t_{con}	**0.00**	0.02	0.10	0.20	0.10	0.10	0.10	0.12	0.10
	t_{cg}	**0.10**	1.88	0.40	0.22	0.23	0.22	0.27	0.22	0.23
	err	0.4E-14	0.7E-14	0.1E-14	0.1E-14	0.1E-14	0.1E-14	0.1E-14	0.1E-14	0.1E-14
HYDCAR20	#it	990*	990*	716	414	308	285	326	326	326
	t_{con}	0.00	0.00	0.00	0.00	**0.00**	0.02	0.03	0.03	0.03
	t_{cg}	0.45	0.53	1.77	0.65	**0.60**	0.70	0.87	0.83	0.83
	err	0.5E00	0.8E01	0.5E-11	0.5E-10	0.1E-10	0.6E-11	0.3E-10	0.3E-10	0.3E-10
MAXLIKA	#it	30	327	24	13	2	2	2	2	2
	t_{con}	0.00	0.02	0.00	0.02	**0.00**	0.03	0.03	0.08	0.08
	t_{cg}	0.03	0.35	0.17	0.05	**0.02**	0.02	0.02	0.02	0.02
	err	0.3E-13	0.1E-12	0.2E-13	0.8E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14	0.2E-14
NONDQUAR	#it	32	419	18	17	16	16	16	16	18
	t_{con}	**0.02**	0.05	0.22	0.18	0.22	0.32	0.58	0.60	0.83
	t_{cg}	**0.08**	1.58	0.72	0.33	0.35	0.45	0.88	0.75	1.03
	err	0.4E-12	0.5E-12	0.3E-12	0.1E-12	0.2E-12	0.2E-12	0.2E-12	0.2E-12	0.2E-13
ORTHREGA	#it	97	589	78	2	2	2	2	2	2
	t_{con}	0.00	0.03	0.03	**0.03**	**0.03**	0.05	0.08	0.10	0.12
	t_{cg}	0.08	0.72	0.80	**0.02**	**0.02**	0.02	0.02	0.00	0.03
	err	0.9E-11	0.4E-11	0.6E-12	0.2E-14	0.1E-14	0.1E-14	0.1E-14	0.2E-14	0.2E-14
ORTHREGC	#it	290	654	77	2	2	2	2	2	2
	t_{con}	0.00	0.05	0.07	**0.05**	0.07	0.08	0.13	0.20	0.27
	t_{cg}	0.45	1.35	1.58	**0.02**	0.03	0.02	0.03	0.02	0.05
	err	0.1E-11	0.7E-12	0.3E-12	0.2E-14	0.3E-14	0.2E-14	0.1E-14	0.2E-14	0.2E-14
ORTHREGD	#it	108	155	16	3	3	3	3	3	3
	t_{con}	0.00	0.05	0.07	**0.05**	0.08	0.10	0.15	0.18	0.25
	t_{cg}	0.15	0.30	0.32	**0.03**	0.02	0.03	0.05	0.05	0.07
	err	0.4E-12	0.2E-12	0.2E-12	0.7E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15
ORTHREGF	#it	191	492	31	15	15	15	15	15	15
	t_{con}	0.00	0.02	0.02	0.03	**0.03**	0.05	0.08	0.10	0.13
	t_{cg}	0.20	0.62	0.42	0.10	**0.07**	0.08	0.08	0.10	0.13
	err	0.2E-11	0.1E-11	0.1E-11	0.1E-12	0.1E-12	0.1E-12	0.1E-12	0.1E-12	0.1E-12
QUARTC	#it	9990*	2834	2	2	2	2	2	2	2
	t_{con}	0.00	0.02	0.22	**0.18**	0.22	0.28	0.43	0.60	0.78
	t_{cg}	21.22	8.70	0.10	**0.03**	0.05	0.07	0.08	0.10	0.12
	err	0.2E00	0.6E-01	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10	0.2E-10
SEMICON1	#it	189	212	63	50	50	50	50	50	50
	t_{con}	**0.00**	0.00	0.12	0.10	0.10	0.08	0.08	0.10	0.10
	t_{cg}	**0.38**	0.62	1.33	0.68	0.72	0.70	0.72	0.68	0.68
	err	0.5E-15	0.7E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15	0.4E-15
TFI1	#it	33	1205	22	2	2	2	2	2	2
	t_{con}	**0.00**	0.15	0.22	0.18	0.23	0.42	0.67	1.00	1.43
	t_{cg}	**0.10**	5.23	0.87	0.07	0.07	0.10	0.15	0.22	0.27
	err	0.1E-11	0.8E-12	0.4E-13	0.1E-13	0.3E-14	0.1E-13	0.1E-13	0.1E-13	0.6E-14
TFI2	#it	4	293	24	2	2	2	2	2	2
	t_{con}	**0.00**	0.15	0.22	0.20	0.25	0.40	0.67	1.00	1.43
	t_{cg}	**0.02**	1.22	0.95	0.05	0.08	0.12	0.15	0.20	0.23
	err	0.7E-14	0.7E-12	0.2E-13	0.4E-14	0.5E-14	0.1E-14	0.2E-14	0.7E-15	0.9E-15
TFI3	#it	4	293	24	2	2	2	2	2	2
	t_{con}	**0.00**	0.15	0.22	0.18	0.25	0.38	0.65	1.00	1.42
	t_{cg}	**0.02**	1.15	0.97	0.07	0.07	0.10	0.15	0.22	0.23
	err	0.7E-14	0.7E-12	0.2E-13	0.4E-14	0.5E-14	0.1E-14	0.2E-14	0.7E-15	0.9E-15

Table 3.5: Results with no, band(1) and SBS(k_{max}) preconditioners.

3.2.2 Experiments with nonzero 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 nonzero 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 nonzero and satisfies $\boldsymbol{A}^T\boldsymbol{r}_0 = \boldsymbol{0}$. This then ensures that the solution $\boldsymbol{x}^* = (1 \ldots 1)^T$ 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 seven nonzero residual problems in Table 3.6. 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 on, for example, problems `BRATU1D` and `BROWNBS`. The SBS preconditioner is the fastest preconditioner for problems `BRATU1D`, `HYDCAR20`, `QUARTC` and `EXPFITC`.

name		no	band(1)	SBS(k_{max}), k_{max} =		
				1	20	50
`BRATU1D`	#it	25210*	1559	16	9	9
	t_{con}	0.00	0.00	0.90	**0.95**	2.22
$\kappa = 6.5 \times 10^5$	t_{cg}	137.66	14.4	1.90	**0.63**	1.27
	err	0.4	3.7D-11	2.1D-11	2.0D-11	2.0D-11
`DQDRTIC`	#it	11	5	1	1	1
	t_{con}	0.00	**0.00**	0.22	0.33	0.80
$\kappa = 5.6 \times 10^9$	t_{cg}	0.10	**0.08**	0.20	0.13	0.26
	err	3.3D-14	2.2D-14	3.5D-14	3.5D-14	3.5D-14
`HYDCAR20`	#it	990*	990*	716	285	326
	t_{con}	0.00	0.00	0.00	**0.02**	0.03
$\kappa = 1.0 \times 10^6$	t_{cg}	0.22	0.35	2.50	**0.60**	0.75
	err	0.6	7.7	5.0D-12	6.0D-12	2.9D-11
`QUARTC`	#it	9990*	2885	2	2	2
	t_{con}	0.00	0.02	0.22	**0.30**	0.80
$\kappa = 5.6 \times 10^9$	t_{cg}	17.20	7.55	0.18	**0.08**	0.13
	err	0.2	5.9D-2	2.4D-11	2.4D-11	2.4D-11
`FLETCHBV`	#it	4593	24	3	2	2
	t_{con}	0.00	**0.02**	1.20	1.18	3.88
$\kappa = 3.0 \times 10^6$	t_{cg}	35.80	**0.28**	0.47	0.20	0.53
	err	3.0D-12	2.2D-12	1.8D-12	1.5D-12	1.1D-12
`EXPFITC`	#it	901	273	22	2	2
	t_{con}	0.00	0.02	0.07	**0.15**	0.62
$\kappa = 7.3 \times 10^2$	t_{cg}	1.33	0.48	0.68	**0.07**	0.10
	err	7.6D-11	6.3D-11	6.2D-11	6.2D-11	6.2D-11
`BROWNBS`	#it	10	10	14	3	3
	t_{con}	**0.00**	0.00	1.18	1.18	2.63
$\kappa = 1.4 \times 10^6$	t_{cg}	**0.08**	0.17	1.77	0.30	0.53
	err	5.0D-04	1.3D-04	5.1D-10	9.0D-10	9.0D-11

Table 3.6: Results with no, band(1) and SBS(k_{max}) preconditioners on nonzero residual problems. κ is the condition number of each matrix.

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 so as to handle substructures of low rank. We first give two examples.

Suppose we wish to find the least-squares solution to a *nonlinear* set of equations by minimizing $\boldsymbol{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|_2^2$, where $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}) \dots f_m(\boldsymbol{x}))^T$. 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

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

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 nonlinear function $f(\boldsymbol{x})$ subject to a set of (nonlinear) constraints $\boldsymbol{c}_i(\boldsymbol{x}) = 0, i = 1, \dots m$, 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

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

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

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

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 Griewank and Toint, 1982), 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 element-wise (as described earlier) and obtain the mixed preconditioner

$$\boldsymbol{A} \approx \boldsymbol{P}_{mix} = \boldsymbol{D}^{\frac{1}{2}} \left(\prod_{i=1}^{e_e} \boldsymbol{L}_i \prod_{i=1}^{e_s} (\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}} \boldsymbol{M}_i^{\mathcal{I}_i} \right) \left(\prod_{i=e_s}^{1} (\boldsymbol{M}_i^{\mathcal{I}_i})^T (\boldsymbol{\Delta}_i^{\mathcal{E}_i})^{\frac{1}{2}} \prod_{i=e_e}^{1} \boldsymbol{L}_i^T \right) \boldsymbol{D}^{\frac{1}{2}}. \tag{4.5}$$

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

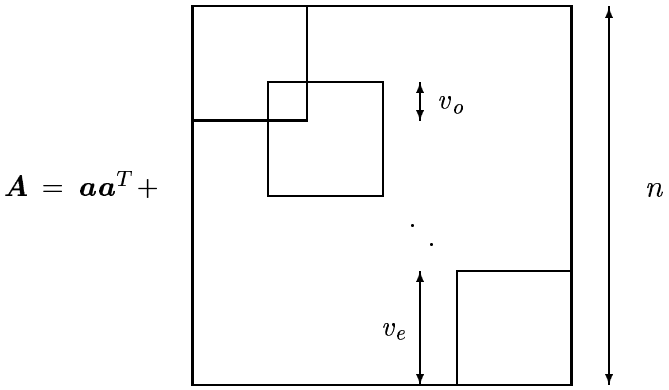

Figure 4.1: The test problem.

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 $\boldsymbol{a}\boldsymbol{a}^T$ involving all the variables. The n_e diagonal blocks are randomly generated to have eigenvalues in the range $[\lambda_{min}, \lambda_{max}]$, while $a_i = 0.1i$ for $1 \leq i \leq n$. The right hand side is chosen so that the exact solution is $(1, \ldots, 1)^T$.

In Tables 4.7 and 4.8, 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)).

In Table 4.7, 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.

Firstly, we see that the mixed preconditioner is effective. When compared to 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 first condition numbers (10 and 10^3) and *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 for the first condition number and 2.7 for the two last condition numbers. 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 the Table 4.8, we compare the effect of varying the degree of overlap between the blocks from 0% to 50%, while keeping the 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).

Prec	t_{con}	t_{pre}	t_{cg}	t_{sol}	#it
$\lambda_{min} = 1$, $\lambda_{max} = 10$					
mixed	0.2	0.1	1.3	**1.2**	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	1412
band(5)	0.1	1.5	50.7	50.8	730
$\lambda_{min} = 1$, $\lambda_{max} = 10^3$					
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	2578
band(5)	0.1	1.7	60.6	60.6	873
$\lambda_{min} = 1$, $\lambda_{max} = 10^5$					
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	1031
band(1)	0.1	1.5	129.5	129.6	1682
band(5)	0.1	1.6	60.9	61.0	886

Table 4.7: Matrix with $n_e = 100$, $v_e = 10$, $v_o = 2$ and $n = 802$. The blocks are generated with eigenvalues in the range $[\lambda_{min}, \lambda_{max}]$. 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.

Overlap	Order	mixed		EBE		diag		band(1)		band(5)	
		#it	t_{sol}	#it	t_{sol}	#it	t_{sol}	#it	t_{sol}	#it	t_{sol}
0	1000	565	**64.2**	865	259.1	1723	179.6	3783	398.5	1726	182.8
1	901	420	**39.7**	634	165.8	1257	107.6	2249	192.2	1183	101.6
2	802	300	**22.7**	409	88.2	1031	80.3	1682	129.6	886	61.0
3	703	207	**11.8**	278	45.0	783	41.2	1010	53.2	602	32.3
4	604	139	**6.2**	183	23.3	483	19.2	743	29.7	410	16.6
5	505	99	**3.3**	126	11.9	345	9.9	454	13.0	304	8.9

Table 4.8: Matrices with $n_e = 100$, $v_e = 10$ and $v_o = 0$ to 5. $\lambda_{min} = 1$ and $\lambda_{max} = 10^5$.

4.2 Tests on some SIF problems

We now turn to less contrived examples which arise from optimization. Given f and $\boldsymbol{c}$, we apply a truncated Newton algorithm (see Daydé et al., 1997*a*) 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 $\sum_{i=1}^{m} \nabla_x c_i(\boldsymbol{x})(\nabla_x c_i(\boldsymbol{x}))^T$ of (4.3) are approximated using the SBS method, while the remainder is handled using EBE factors.

In Tables 4.9–4.11, 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 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 smallest when using the mixed preconditioner. For the problem `STEENBRA` (Table 4.9), the diagonal preconditioner is the 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 4.10), 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 4.11), the mixed preconditioner proves to be fastest for all but the smallest ρ.

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

	$\rho = 1$						$\rho = 10$					
Prec	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
	$\rho = 100$						$\rho = 1000$					
Prec	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

Table 4.9: Solution of the problem `STEENBRA` with different penalty values.

	$\rho = 1$						$\rho = 10$					
Prec	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
	$\rho = 100$						$\rho = 1000$					
Prec	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	1209	105.0	104.2	0.0	104.2	9	1790	153.4	153.4	0.0	153.4
EBE	8	1018	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 4.10: Solution of the problem `GRIDNETA` with different penalty values.

	$\rho = 1$						$\rho = 10$					
Prec	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
	$\rho = 100$						$\rho = 1000$					
Prec	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

Table 4.11: Solution of the problem `GENHS28` with different penalty values.

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.

Acknowledgment

The authors are very grateful to Iain Duff and Daniel Ruiz for their help in reading and improving draft versions of this paper.

References

Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**(1), 123–160, 1995.

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, eds, 'Sparse Matrix Computations', pp. 309–332, Academic Press, London and New York, 1976.

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, pp. 942–953, 1997*a*.

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*b*.

I. S. Duff. The solution of augmented systems. *In* D. F. Griffiths and G. A. Watson, eds, 'Numerical Analysis 1993', pp. 40–55, Longman Scientific and Technical, Harlow, England, 1994.

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.

J. Erhel, A. Traynard, and M. Vidrascu. An element-by-element preconditioned conjugate gradient method implemented on a vector computer. *Parallel Computing*, **17**, 1051–1065, 1991.

P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, **28**, 505–535, 1974.

D. Goldfarb and S. Wang. Partial-update Newton methods for unary, factorable, and partially separable optimization. *SIAM Journal on Optimization*, **3**(2), 382–397, 1993.

A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. *In* M. J. D. Powell, ed., 'Nonlinear Optimization 1981', pp. 301–312, Academic Press, London and New York, 1982.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**, 409–436, 1952.

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. *Computational Methods in Applied Mechanics and Engineering*, **61**, 215–248, 1987.

T. J. R Hughes, I. Levit, and J. Winget. An element-by-element solution algorithm for problems of structural and solid mechanics. *Computational Methods in Applied Mechanics and Engineering*, **36**, 241–254, 1983.

J.-Y. L'Excellent. Utilisation de préconditionneurs élément-par-élément pour la résolution de problèmes d'optimisationde grande taille. *PhD Thesis, Institut National Polytechnique de Toulouse*, 1995.

P. Matstoms. Sparse QR factorization in MATLAB. *ACM Transactions on Mathematical Software*, **20**(1), 136–159, 1994.

M. Ortiz, P. M. Pinsky, and R. L. Taylor. Unconditionally stable element-by-element algorithms for dynamic problems. *Computational Methods in Applied Mechanics and Engineering*, **36**, 223–239, 1983.

B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, New Jersey, 1980.