MULTI-PRECONDITIONED GMRES

CHEN GREIF[*], TYRONE REES[*], AND DANIEL B. SZYLD[†]

Abstract. Standard Krylov subspace methods only allow the user to choose a single preconditioner, although in many situations there may be a number of possibilities. Here we describe an extension of GMRES, multi-preconditioned GMRES, which allows the use of more than one preconditioner. We give some theoretical results, propose a practical algorithm, and present numerical results from problems in domain decomposition and PDE-constrained optimization. These numerical experiments illustrate the applicability and potential of the multi-preconditioned approach.

Key words. Krylov subspaces, iterative methods, preconditioning, linear systems, GMRES

AMS subject classifications. 65F10, 65N22, 15A06

1. Introduction. Let $\mathcal{A} \in \mathbb{R}^{n \times n}$ be a matrix, possibly indefinite and/or non-symmetric, and $\mathbf{b} \in \mathbb{R}^n$. Suppose we wish to solve the linear system

$$\mathcal{A}\mathbf{x} = \mathbf{b}, \tag{1.1}$$

using a modern iterative method, such as those associated with a Krylov subspace [14]. One ingredient in the successful application of these methods is the use of a preconditioner, $\mathcal{P}$. This is an easily invertible matrix (or, more generally, a linear operation which implicitly defines the inverse of a matrix) which, when applied to the system, transforms it into an equivalent one for which the algorithm converges more rapidly. One can apply a preconditioner on the left, the right, or a combination of both. Here we consider only right preconditioning, which means that we consider the equivalent system

$$\mathcal{A}\mathcal{P}^{-1}\mathbf{u} = \mathbf{b}, \tag{1.2}$$

where $\mathbf{u} = \mathcal{P}\mathbf{x}$. At each step of a Krylov subspace method one needs to perform a matrix-vector product, which in the case of (1.2) implies first the solution of a system, say $\mathcal{P}\mathbf{z} = \mathbf{v}$, and the product $\mathcal{A}\mathbf{z} = \mathcal{A}\mathcal{P}^{-1}\mathbf{v}$.

In this paper we derive a method in which one can use more than one (right) preconditioner with a minimal residual method for nonsymmetric matrices such as GMRES [15]. It is not uncommon to have two or more different preconditioners for the same problem, but with very different properties. What we propose is a way to use all preconditioners in an optimal manner. As we shall see, the optimal may be too expensive; but some affordable approximation to the optimal might provide significantly more rapid convergence by combining the preconditioners than by using a method which forces the user to pick just one.

We were inspired in part by [2] where such multiple preconditioning is used for the conjugate gradient method (CG) for symmetric positive definite linear systems. The algorithm from [2] is called multi-preconditioned CG (MPCG). One issue with

this algorithm is that, despite being designed for symmetric positive definite matrices, MPCG does not in general retain the short-term recurrence of CG. It does, however, combine the preconditioners in an optimal way.

We mention also two other approaches to multi-preconditioning in the literature. In [12], it is proposed to use the flexible GMRES (FGMRES) algorithm [13] with cycling preconditioners, while in [8] the preconditioner is based on multi-splitting methods. In both cases preconditioners are combined, but not optimally; we describe these in sections 4.1 and 4.2 respectively.

The rest of the paper is organized as follows. In Section 2 we derive and prove some results about an extension of the GMRES algorithm which allows us to use more than one preconditioner simultaneously. We present both an ideal or optimal algorithm, which is of theoretical interest but is not practical, and a practical truncated version. We give some further theoretical details and discuss issues related to the implementation of the method in Section 3. In Section 4 we discuss some algorithms in the literature which can be thought of an non-optimal truncations of the new algorithm. In Section 5 we present some numerical experiments illustrating the effectiveness of the proposed method.

Notation. Throughout this manuscript scalars are denoted by lower case letters, vectors are denoted by lower case bold type and matrices are denoted by upper case letters. A quantity (vector or matrix) which is the result of an iteration is denoted by a bracketed superscript numeral. A single numeral subscript appended after a matrix denotes the column, and a pair of numerals denotes a single entry. A subscript of two numbers separated by a colon following a matrix denotes a submatrix of the original matrix. We use $\mathcal{R}(A)$ to denote the range of A, i.e., the space spanned by the columns of a matrix A.

2. MPGMRES. We start this section with a brief description of the standard GMRES algorithm, and some comments on right and left preconditioning.

The Generalized Minimal Residual (GMRES) algorithm of Saad and Schultz [15] is the solution method of choice for nonsymmetric linear systems (1.1). Starting from some initial guess $\mathbf{x}^{(0)}$, the kth step GMRES (or any other minimal residual method) consists of computing the vector $\mathbf{x}^{(k)}$ such that

$$\mathbf{x}^{(k)} := \arg \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_k(\mathcal{A}, \mathbf{r}^{(0)})} \|\mathbf{b} - \mathcal{A}\hat{\mathbf{x}}\|_2,$$

where $\mathbf{r}^{(k)} := \mathbf{b} - \mathcal{A}\mathbf{x}^{(k)}$ and

$$\mathcal{K}_k(\mathcal{A}, \mathbf{r}^{(0)}) := \text{span}\{\mathbf{r}^{(0)}, \mathcal{A}\mathbf{r}^{(0)}, \dots, \mathcal{A}^{k-1}\mathbf{r}^{(0)}\},$$

a Krylov subspace. The strength of GMRES, as opposed to other minimal residual methods relates to its implementation details; see, e.g., [14, 19].

When we consider the preconditioned system (1.2), then GMRES finds $\mathbf{u}^{(k)}$ which minimizes the 2-norm of the residual over $\mathbf{u}^{(0)} + \mathcal{K}_k(\mathcal{A}\mathcal{P}^{-1}, \mathbf{r}^{(0)})$. Written in terms of $\mathbf{x}$, since $\mathbf{x} = \mathcal{P}^{-1}\mathbf{u}$, we therefore find

$$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathcal{P}^{-1}\mathcal{K}_k(\mathcal{A}\mathcal{P}^{-1}, \mathbf{r}^{(0)})$$
$$= \mathbf{x}^{(0)} + \mathcal{K}_k(\mathcal{P}^{-1}\mathcal{A}, \mathcal{P}^{-1}\mathbf{r}^{(0)}). \tag{2.1}$$

Note that this space is the same as in left-preconditioned GMRES. The difference between the two approaches is the functional which is minimized, which depends on

the preconditioner in left-preconditioned GMRES, but not with right preconditioning; see, e.g., [14, p. 272], [16].

An orthonormal basis for the Krylov subspace $\mathcal{K}_k(\mathcal{A}\mathcal{P}^{-1}, \mathbf{r}^{(0)})$ is found using the Arnoldi algorithm. This generates a decomposition of the form

$$(\mathcal{A}\mathcal{P}^{-1})V_k = V_{k+1}\widetilde{H}_k, \tag{2.2}$$

where $V_k \in \mathbb{R}^{n \times k}$ is orthogonal with the first column being $\mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$, and $\widetilde{H}_k \in \mathbb{R}^{(k+1) \times k}$ is upper Hessenberg.

Note that the columns of V_k span the space $\mathcal{K}_k(\mathcal{A}\mathcal{P}^{-1}, \mathbf{r}^{(0)})$. The iterate $\mathbf{x}^{(k)}$ therefore must have the form

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathcal{P}^{-1}V_k\mathbf{y}^{(k)} \tag{2.3}$$

for some vector $\mathbf{y}^{(k)} \in \mathbb{R}^k$. Therefore

$$\begin{aligned}
\|\mathbf{b} - \mathcal{A}\mathbf{x}^{(k)}\|_2 &= \|\mathbf{b} - \mathcal{A}(\mathbf{x}^{(0)} + \mathcal{P}^{-1}V_k\mathbf{y}^{(k)})\|_2 \\
&= \|\mathbf{r}^{(0)} - \mathcal{A}\mathcal{P}^{-1}V_k\mathbf{y}^{(k)})\|_2 \\
&= \|\mathbf{r}^{(0)} - V_{k+1}\widetilde{H}_k\mathbf{y}^{(k)})\|_2 \\
&= \|\|\mathbf{r}^{(0)}\|_2\mathbf{e}_1 - \widetilde{H}_k\mathbf{y}^{(k)}\|_2,
\end{aligned} \tag{2.4}$$

where we have used (2.3), (2.2) and the fact that V_{k+1} has orthonormal columns. The problem of finding the vector which minimizes the residual over the Krylov subspace is therefore equivalent to solving the least-squares problem (2.4). Due to the structure of $\widetilde{H}_k$, this is an easy task.

Note that we can write the Arnoldi relation (2.2) in the equivalent form

$$\mathcal{A}Z_k = V_{k+1}H_k, \tag{2.5}$$

where $Z_k = \mathcal{P}^{-1}V_k$. Equation (2.5) allows us to extend the concept of right preconditioning beyond that described above – there is no need for V_k be included on the left of (2.5) at all. The matrix Z_k could be any matrix whose range is the space in which we wish to find our approximation. This is the basis of flexible GMRES (FGMRES) [13], an extension of GMRES which allows the preconditioner to change at each iteration; see Section 4.1.

2.1. Derivation of MPGMRES. In many applications the ideal choice of preconditioner is not clear, and we may have many candidate preconditioners; see, e.g., [1]. Suppose that instead of just a single preconditioner we have t preconditioners, $\mathcal{P}_1, \ldots, \mathcal{P}_t$. Take any one of these, $\mathcal{P}_i$ say, and then apply the standard GMRES algorithm described above. Consider the first iteration, which we denote by $\mathbf{x}_i^{(1)}$. By (2.1) GMRES finds a scalar – y_i, say – such that $\mathbf{x}_i^{(1)} = \mathbf{x}^{(0)} + (\mathcal{P}_i^{-1}\mathbf{r}^{(0)})y_i$ has the minimal residual in the 2-norm.

Now consider all t first iterates $\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_t^{(1)}$ generated by GMRES applied with the t preconditioners. Starting from $\mathbf{x}^{(0)}$, each instance of GMRES individually finds the optimal step length in the direction of the vector $\mathcal{P}_i^{-1}\mathbf{r}^{(0)}$. If we wish to incorporate information from all preconditioners in one method, it makes sense to – at the first step – minimize the residual of all vectors of the form

$$\mathbf{x}^{(0)} + \text{span}\{\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \ldots, \mathcal{P}_t^{-1}\mathbf{r}^{(0)}\}.$$

3

Define

$$Z^{(1)} = [\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \ldots, \mathcal{P}_t^{-1}\mathbf{r}^{(0)}] \in \mathbb{R}^{n \times t}. \tag{2.6}$$

The t columns of $Z^{(1)}$ will – excepting some obvious degeneracies – almost always be linearly independent. We can therefore define the first iterate of a new method as

$$\mathbf{x}^{(1)} := \mathbf{x}^{(0)} + Z^{(1)}\mathbf{y}^{(1)},$$

where the vector $\mathbf{y}^{(1)} \in \mathbb{R}^t$ is chosen to minimize the residual in the 2-norm.

In order to extend the first step described above to a GMRES-style algorithm we need an analog of the Arnoldi decomposition (2.5). This can be obtained by orthogonalizing $\mathcal{A}Z^{(1)}$ against $V^{(1)} := \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$ by a block Gram-Schmidt procedure, and then performing a skinny QR factorization on the resulting matrix. We can then iterate this process, resulting in the following loop:

> **for** $i = 1, \ldots k$ **do**
> $\quad W = \mathcal{A}Z^{(i)}$
> $\quad$ **for** $j = 1, \ldots, i$ **do**
> $\quad\quad H^{(j,i)} = (V^{(j)})^T W$
> $\quad\quad W = W - V^{(j)}H^{(j,i)}$
> $\quad$ **end for**
> $\quad W = V^{(i+1)}H^{(i+1,i)}$ $\quad\quad$ (skinny QR factorization)
> $\quad Z^{(i+1)} = [\mathcal{P}_1^{-1}V^{(i+1)} \cdots \mathcal{P}_t^{-1}V^{(i+1)}]$
> **end for**

This construction leads to the Arnoldi-type equality

$$\mathcal{A}\widetilde{Z}_k = \widetilde{V}_{k+1}\widetilde{H}_k, \tag{2.7}$$

where

$$\widetilde{Z}_k = \begin{bmatrix} \vdots & & \vdots \\ Z^{(1)} & \cdots & Z^{(k)} \\ \vdots & & \vdots \end{bmatrix}, \qquad \widetilde{V}_{k+1} = \begin{bmatrix} \vdots & & \vdots \\ V^{(1)} & \cdots & V^{(k+1)} \\ \vdots & & \vdots \end{bmatrix}$$

and

$$\widetilde{H}_k = \begin{bmatrix} H^{(1,1)} & H^{(1,2)} & \cdots & & H^{(1,k)} \\ H^{(1,2)} & H^{(2,2)} & & & H^{(2,k)} \\ & & \ddots & & \vdots \\ & & & H^{(k,k-1)} & H^{(k,k)} \\ & & & & H^{(k+1,k)} \end{bmatrix}.$$

We have implicitly assumed that $Z^{(k)}$ and $\widetilde{Z}_k$ have full rank. We keep this assumption throughout this section and address the situation when this assumption fails to hold in Section 3.

Observe that $Z^{(1)}$ and $V^{(2)}$ have t columns, while $Z^{(2)}$ and $V^{(3)}$ have t^2 columns, and in general $Z^{(i)}$ and $V^{(i+1)}$ have t^i columns. Therefore $\widetilde{V}_{k+1}$ has

$$\tau_k := \sum_{i=0}^{k} t^i = \frac{t^{k+1} - 1}{t - 1}$$

4

columns, while $\widetilde{Z}_k$ has $\tau_k - 1 = (t^{k+1} - t)/(t-1)$ columns. Note also that for all $H^{(1,i)} \in \mathbb{R}^{1 \times t}$ for all i, and since the blocks on the subdiagonal come from the QR factorization, they are all upper triangular. Therefore, as in the standard Arnoldi algorithm, the matrix $\widetilde{H}_k$ above is upper Hessenberg, here of order $(\tau_k - 1) \times \tau_k$. Figure 2.1 is a schematic diagram of the Arnoldi decomposition (2.7) showing the dimensions of the matrices involved.
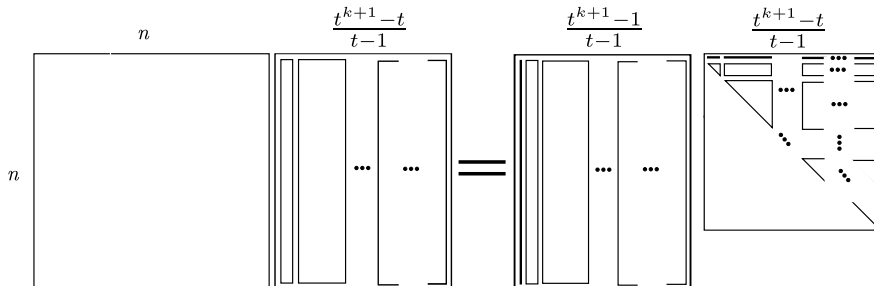

FIG. 2.1. *Schematic of Arnoldi decomposition for full multi-preconditioned Arnoldi*

In the following, we refer to the columns of $Z^{(k)}$ as *search directions*, the matrix $Z^{(k)}$ as the kth, *or current, matrix of search directions*, and the matrix $\widetilde{Z}_k$ as the *search matrix*. The equivalent terms for the columns of $V^{(k)}$, the matrix $V^{(k)}$ and the matrix $\widetilde{V}_k$ are *basis vectors, matrix of basis vectors*, and *basis matrix*, respectively.

Now consider a vector $\mathbf{z} = \mathbf{x}^{(0)} + \widetilde{Z}_k \mathbf{y}$, where $\mathbf{y}$ is a column vector. Then

$$\begin{aligned}
\mathbf{b} - \mathcal{A}\mathbf{z} &= \mathbf{b} - \mathcal{A}(\mathbf{x}^{(0)} + \widetilde{Z}_k \mathbf{y}) \\
&= \mathbf{r}^{(0)} - \mathcal{A}\widetilde{Z}_k \mathbf{y} \\
&= \beta V^{(1)} - \widetilde{V}_{k+1}\widetilde{H}_k \mathbf{y} \\
&= \widetilde{V}_{k+1}(\beta \mathbf{e}_1 - \widetilde{H}_k \mathbf{y}),
\end{aligned}$$

where $\beta = \|\mathbf{r}^{(0)}\|_2$. Since the basis matrix $\widetilde{V}_{k+1}$ is an orthogonal matrix, we have

$$\arg \min_{\mathbf{z} \in \mathbf{x}^{(0)} + \mathcal{R}(\widetilde{Z}_k)} \|\mathbf{b} - \mathcal{A}\mathbf{z}\|_2 = \arg \min_{\mathbf{y}} \|\beta \mathbf{e}_1 - \widetilde{H}_k \mathbf{y}\|_2. \tag{2.8}$$

The method described above is given as Algorithm 1, and we call it *full multi-preconditioned GMRES* (MPGMRES).

Note that Algorithm 1 contains both block-Arnoldi and QR factorizations. Both of these algorithms can be thought of in terms of Gram-Schmidt orthogonalization, and we can perform the QR step explicitly in an Arnoldi-style algorithm. This was first proposed (for the block-Lanczos case) by Ruhe [11] and is described by Saad [14, p. 209] for the block-Arnoldi case. We give this mathematically equivalent alternative method as Algorithm 3.

Algorithm 3 requires only matrix-vector operations, and so it bears more similarity to the standard GMRES algorithm. This version has the drawback that it relies heavily on level 1 BLAS routines, as opposed to Algorithm 1 which makes use of level 3 BLAS. However, in a parallel implementation a processor could start on the Arnoldi step as soon as each search direction has been calculated, whereas in Algorithm 1 this

---

Algorithm 1 MPGMRES

  calls Algorithm 2
  Choose $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
  $\beta = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
  $Z^{(1)} = \texttt{fullmultiprecondition}(\mathbf{r}^{(0)})$
  for $k = 1, \ldots,$ **until** convergence **do**
    $W = \mathcal{A}Z^{(k)}$
    for $j = 1, \ldots, k$ **do**
      $H^{(j,k)} = (V^{(j)})^T W$
      $W = W - V^{(j)} H^{(j,k)}$
    end for
    $W = V^{(k+1)} H^{(k+1,k)}$ (skinny QR factorization)
    $\mathbf{y}^{(k)} = \text{argmin}\|\beta \mathbf{e}_1 - \widetilde{H}_k \mathbf{y}\|_2$
    $\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + [Z^{(1)} \cdots Z^{(k)}]\mathbf{y}^{(k)}$
    $Z^{(k+1)} = \texttt{fullmultiprecondition}(V^{(k+1)})$
  end for

Algorithm 2 Subroutine: full multi-preconditioning step

  function $Z = \texttt{fullmultiprecondition}(V)$
  $Z = [\mathcal{P}_1^{-1}V \cdots \mathcal{P}_t^{-1}V]$
  end function

---

would have to wait until the entire matrix of search directions has been calculated. Also, due to the vectorized nature it is straightforward to adapt the version presented in Algorithm 3 to deal with technicalities related to rank-deficiency that arise in a practical implementation – see the subsequent section for details. The version of the algorithm which should be used is therefore highly dependent on the needs of the implementation, and so we give both here.

2.2. Characterizing the search space. Recall that GMRES preconditioned with a right preconditioner $\mathcal{P}$ finds the vector that minimizes the 2-norm of the residual over all vectors of the form

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathcal{P}^{-1}\mathbf{w}_k,$$

where $\mathbf{w}_k$ is a member of the Krylov subspace

$$\mathcal{K}_k(\mathcal{A}\mathcal{P}^{-1}, \mathbf{r}^{(0)}) = \text{span}(\mathbf{r}^{(0)}, \mathcal{A}\mathcal{P}^{-1}\mathbf{r}^{(0)}, \ldots, (\mathcal{A}\mathcal{P}^{-1})^{k-1}\mathbf{r}^{(0)}).$$

For ease of illustration, consider first the case where two preconditioners are used. The obvious (theoretical) extension of a Krylov subspace method now would be a method where the first two iterates satisfy

$$\mathbf{x}^{(1)} - \mathbf{x}^{(0)} \in \text{span}\{\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathbf{r}^{(0)}\}$$
$$\mathbf{x}^{(2)} - \mathbf{x}^{(0)} \in \text{span}\{\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathbf{r}^{(0)}, \mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathbf{r}^{(0)}, \qquad (2.9)$$
$$\mathcal{P}_2^{-1}\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathbf{r}^{(0)}\},$$

and the rest follow the same pattern.

Let us define $\mathbb{P}_{k-1} = \mathbb{P}_{k-1}(t)$ to be the space of all possible (noncommuting) polynomials of matrices in t variables of degree $k - 1$ or less. Returning to the case

---

Algorithm 3 MPGMRES: vectorized version

---

calls Algorithm 2
Choose $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
$\beta = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
$Z^{(1)} = \texttt{fullmultiprecondition}(\mathbf{r}_0)$
$\widetilde{V} = \mathbf{v}^{(1)}$
$n_{\widetilde{V}} = 1$ (no. of columns in $\widetilde{V}$)
$n_{Z^{(1)}} =$ no. of columns in $Z^{(1)}$, $n_Z = n_{Z^{(1)}}$
for $k = 1, \ldots,$ **until** convergence **do**
   for $\ell = 1, \ldots, n_{Z^{(k)}}$ **do**
     $\mathbf{w} = \mathcal{A}Z^{(k)}{}_\ell$
     for $j = 1, \ldots, n_{\widetilde{V}}$ **do**
       $\widetilde{H}_k(j, n_{\widetilde{V}}) = \mathbf{w}^T \widetilde{V}_j$
       $\mathbf{w} = \mathbf{w} - \widetilde{V}_j \widetilde{H}_k(j, n_{\widetilde{V}})$
     end for
     $\widetilde{H}_k(n_{\widetilde{V}} + 1, n_{\widetilde{V}}) = \|\mathbf{w}\|_2$
     $\widetilde{V} = [\widetilde{V} \quad \mathbf{w}/\widetilde{H}_k(n_{\widetilde{V}} + 1, n_{\widetilde{V}})]$
     $n_{\widetilde{V}} = n_{\widetilde{V}} + 1$
   end for
   $\mathbf{y}^{(k)} = \text{argmin}\|\beta \mathbf{e}_1 - \widetilde{H}_k \mathbf{y}\|_2$
   $\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + [Z^{(1)} \cdots Z^{(k)}]\mathbf{y}^{(k)}$
   $Z^{(k+1)} = \texttt{fullmultiprecondition}(\widetilde{V}_{n_Z+1:n_Z+n_{Z^{(k)}}})$
   $n_{Z^{(k)}} =$ no. of columns in $Z^{(k)}$, $n_Z = n_Z + n_{Z^{(k)}}$
end for

---

where we have t preconditioners, in generality the space in which our idealized method chooses iterates can be characterized by

$$\mathbf{x}^{(k)} - \mathbf{x}^{(0)} = \sum_{i=1}^{t} p_k^i(\mathcal{P}_1^{-1}\mathcal{A}, \ldots, \mathcal{P}_t^{-1}\mathcal{A})\mathcal{P}_i^{-1}\mathbf{r}^{(0)},$$

where $p_k^i(X_1, \ldots, X_t) \in \mathbb{P}_{k-1}$.

DEFINITION 2.1. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_t \in \mathbb{R}^{n \times n}$ be preconditioners, $\mathcal{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{w} \in \mathbb{R}^n$. We define the* multi-Krylov *subspace*

$$\mathcal{K}_k^{\mathcal{P}_1, \ldots, \mathcal{P}_t}(\mathcal{A}, \mathbf{w}) := \{p(\mathcal{P}_1^{-1}\mathcal{A}, \ldots, \mathcal{P}_t^{-1}\mathcal{A})\mathbf{w} \; : \; p(X_1, \ldots, X_t) \in \mathbb{P}_{k-1}\}.$$

We note that $\dim \mathcal{K}_k^{\mathcal{P}_1, \ldots, \mathcal{P}_t}(\mathcal{A}, \mathbf{w}) = \dim \mathbb{P}_{k-1} = \tau_{k-1}$.

Using Definition 2.1, we can now characterize the space in which we find the kth iterate by

$$\mathbf{x}^{(k)} - \mathbf{x}^{(0)} \in \mathcal{S}_k^{\mathcal{P}_1, \ldots, \mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}) := \sum_{i=1}^{t} \mathcal{K}_k^{\mathcal{P}_1, \ldots, \mathcal{P}_t}(\mathcal{A}, \mathcal{P}_i^{-1}\mathbf{r}^{(0)}), \qquad (2.10)$$

and we note that

$$\dim \mathcal{S}_k^{\mathcal{P}_1, \ldots, \mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}) = t\tau_{k-1} = \tau_k - 1. \qquad (2.11)$$

For example, $\mathbf{x}^{(3)}$ will be formed from spaces of matrix polynomials in

$$\mathbb{P}_2 = \mathrm{span}\left\{I, X_1, X_2, X_1^2, X_1 X_2, X_2 X_1, X_2^2\right\}.$$

The following result shows formally that the space (2.10) is precisely where MPGMRES looks for the next iterate.

PROPOSITION 2.2. *MPGMRES as described in Algorithm 1 selects, at the kth step, the approximate solution*

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathcal{S}_k^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}),$$

and in fact $\mathbf{x}^{(k)}$ is the vector which minimizes the 2-norm of the residual $\mathbf{b} - A\mathbf{x}^{(k)}$ over this space.

Proof. It is clear that Algorithm 1 is an implementation of (2.8), and therefore all we need to show is that $\mathcal{R}(\widetilde{Z}_k) = \mathcal{S}_k^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)})$. We begin by observing that from (2.11) it follows that these two subspaces have the same dimension. Therefore, it suffices to show that

$$\mathcal{R}(\widetilde{Z}_k) \subseteq \mathcal{S}_k^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}). \tag{2.12}$$

We do so by induction on k. For $k = 1$, this is precisely what we have in line 3 of Algorithm 1, i.e., $\widetilde{Z}_1 = Z^{(1)} = [\mathcal{P}_1^{-1}\mathbf{r}^{(0)} \cdots \mathcal{P}_t^{-1}\mathbf{r}^{(0)}]$, and $\mathcal{R}(Z^{(1)}) = \sum_{i=1}^t \mathcal{R}(\mathcal{P}_i^{-1}\mathbf{r}^{(0)}) = \mathcal{S}_1(\mathcal{A}, \mathbf{r}^{(0)})$. Assume now that (2.12) holds for some k. Since $\widetilde{Z}_{k+1} = [\widetilde{Z}_k\ Z^{(k+1)}]$, and $\mathcal{S}_k^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}) \subseteq \mathcal{S}_{k+1}^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)})$, all that remains to be shown is that $\mathcal{R}(Z^{(k+1)}) \subseteq \mathcal{S}_{k+1}^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)})$. To that end, we have by construction, e.g., from (2.7), and from the fact that $V^{(1)} = \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$, that

$$V^{(k+1)} = \mathcal{A}Z^{(k)} - \sum_{i=1}^k V^{(i)} H^{(i,k)}, \quad \text{and thus}$$

$$\begin{aligned}
\mathcal{R}(V^{(k+1)}) &\subseteq \mathcal{R}(\mathcal{A}Z^{(k)}) + \cdots + \mathcal{R}(\mathcal{A}Z^{(1)}) + \mathrm{span}(\mathbf{r}^{(0)}) \\
&= \mathrm{span}(\mathbf{r}^{(0)}) + \mathcal{A}\left(\mathcal{R}(Z^{(1)}) + \cdots + \mathcal{R}(Z^{(k-1)}) + \mathcal{R}(Z^{(k)})\right) \\
&= \mathrm{span}(\mathbf{r}^{(0)}) + \mathcal{A}\left(\mathcal{R}(\widetilde{Z}_k)\right) = \mathrm{span}(\mathbf{r}^{(0)}) + \mathcal{A}\,\mathcal{S}_k^{\mathcal{P}_1,\dots,\mathcal{P}_t}(\mathcal{A}, \mathbf{r}^{(0)}).
\end{aligned}$$

Since $Z^{(k+1)} = [\mathcal{P}_1^{-1}V^{(k+1)} \cdots \mathcal{P}_t^{-1}V^{(k+1)}]$, the result follows. $\square$

REMARK 2.3. *If the preconditioned systems $\mathcal{P}_i^{-1}\mathcal{A}$ commute – as would be the case with, say, polynomial preconditioners – then the space spanned would be isomorphic to the space of scalar polynomials. In this case error bounds analogous to those in classical GMRES can be derived.*

In light of Proposition 2.2, using this method significantly increases the scope for choosing effective preconditioners compared with the standard preconditioned GMRES; if $\mathbf{x} - \mathbf{x}^{(0)}$ is, say, $\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathbf{r}^{(0)}$, then we will converge in two steps. As with standard GMRES, in general, the eigenvalues alone will not predict entirely the convergence [6, 18], however if we pick preconditioners so that the eigenvalues of, e.g., $\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathcal{A}$ are clustered away from zero, then we will often see rapid convergence.

We now further show that the space over which MPGMRES picks its iterates is significantly richer than in standard preconditioned GMRES. We give an idealized situation which is constructed to show that MPGMRES can – at least in theory – perform significantly better than either of the preconditioners taken separately. Recall

that MPGMRES minimizes the 2-norm of the residual over all vectors in the space (2.10). Therefore – given any preconditioners – it is possible to pick a right-hand side vector $\mathbf{b}$ such that $\mathbf{x}^{(k)}$ lies in this space, but the iterates for standard preconditioned GMRES does not.

Suppose we have two preconditioners, $\mathcal{P}_1$ and $\mathcal{P}_2$, and we take $\mathbf{x}^{(0)} = \mathbf{0}$. Then we know from (2.10) that, for example,

$$\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathbf{b} \in \mathcal{K}_2^{\mathcal{P}_1,\mathcal{P}_2}(\mathcal{A}, \mathcal{P}_1^{-1}\mathbf{r}^{(0)}) + \mathcal{K}_2^{\mathcal{P}_1,\mathcal{P}_2}(\mathcal{A}, \mathcal{P}_2^{-1}\mathbf{r}^{(0)}).$$

Hence, if the solution $\mathbf{x}$ satisfies

$$\mathbf{x} = \lambda \mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}\mathbf{b},$$

for some scalar λ, then MPGMRES will converge after 2 iterations. It is easy to see that is equivalent to $\mathbf{b}$ being an eigenvector of $\mathcal{A}\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1}$ with λ the corresponding eigenvalue. This generalizes in an obvious fashion to a large number of preconditioners. The first example in Section 5 confirms our observations here, for this special right-hand side case.

2.3. Truncated MPGMRES. As we saw in the previous section, the dimension of the space in which we look for the solution in MPGMRES grows exponentially fast, since $Z^{(k)} \in \mathbb{R}^{n \times t^k}$. This means that – unless we have very rapid convergence – the full algorithm will be almost always impractical. However, we can find a good approximation to the solution without using the whole space, but instead choosing an appropriate subspace whose dimension would grow only linearly. We develop this method, which we call truncated MPGMRES, in the following.

We remark that the MPGMRES algorithm, and its truncated version described below, can also be adapted to use less storage in a way analogous to that in restarted GMRES [14, Algorithm 6.11] or GMRES(m) for a single preconditioner. We do not consider these options further here.

One way to truncate full MPGMRES to get a practical algorithm is, at step k, to only apply the preconditioners certain columns of $V^{(k)}$. For example, we could apply $\mathcal{P}_1$ to the first column of $V^{(k)}$, $\mathcal{P}_2$ to the second, and so on. Using this method the matrix $Z^{(k)}$ is in $\mathbb{R}^{n \times t}$ for all k, and we just have to perform a QR factorization of the size of the (often small) number of preconditioners. We call this algorithm truncated multi-preconditioned GMRES (truncated MPGMRES), and it is obtained by applying the function described in Algorithm 4 in place of all but the first call to `fullmultiprecondition` in Algorithm 1 or 3.

Algorithm 4 Subroutine: truncated multi-preconditioning step

function $Z = $ `truncmultiprecondition`(V)
$Z = [\mathcal{P}_1^{-1}V_1 \cdots \mathcal{P}_t^{-1}V_t]$
end function

We repeat that Algorithm 4 represents just one example of a possible truncation strategy. Other choices are possible, and we mention a few of them below. For instance, more generally than in Algorithm 4, given some permutation π (which may or may not change with each iteration), we can compute the ith column of Z as $\mathcal{P}_i^{-1}V_{\pi(i)}$.

Another possible truncation is to use all the columns of $V^{(k)}$ simultaneously by applying the multi-preconditioning step 2 to the vector $V^{(k)}\mathbf{1}$, where $\mathbf{1}$ denotes the

vector of ones. Note that is method will also keep $Z^{(k)} \in \mathbb{R}^{n \times t}$ for all k. Again, this method can be generalized by applying Algorithm 2 to $V^{(k)}\boldsymbol{\alpha}$ for any vector $\boldsymbol{\alpha}$ of appropriate size.

Applying any of the truncated schemes described above produces an Arnoldi-type decomposition, which we show schematically in Figure 2.3. For the rest of this section, and for most of the paper, we will use the strategy developed in Algorithm 4 as the premier example of the truncated version of MPGMRES.
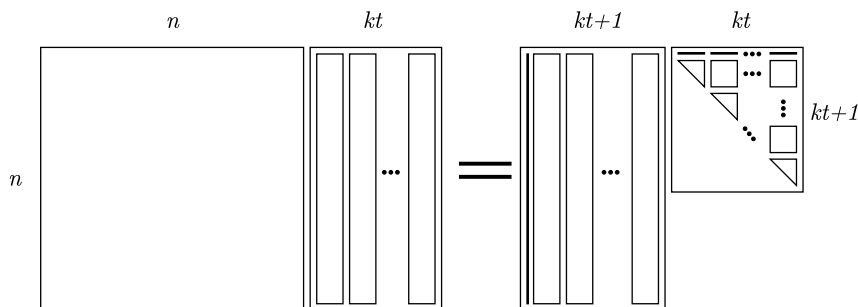

FIG. 2.2. *Schematic of Arnoldi decomposition for truncated multi-preconditioned Arnoldi*

Truncated MPGMRES finds iterates in a particular subspace of the multi-Krylov subspace (2.10). Specifically, while the first iterate $\mathbf{x}^{(1)}$ will be the same for both full and truncated MPGMRES, $\mathbf{x}^{(2)}$ will be chosen from a $(t + t^2)$-dimensional space in the full MPGMRES algorithm, but only a $2t$-dimensional space in the truncated algorithm described above. Recall that for $t = 2$, full GMRES picks an iterate which satisfies (2.9). Truncated MPGMRES will look for an approximate solution in a four-dimensional subspace of this six-dimensional space.

We can say a little more about the structure of this subspace. First, we have $\mathcal{R}(Z^{(1)}) = \operatorname{span}\{\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathbf{r}^{(0)}\}$, and after orthogonalization the first vector of $V^{(2)}$ is parallel to $\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}$, and the second has components of both $\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}$ and $\mathcal{A}\mathcal{P}_1^{-2}\mathbf{r}^{(0)}$. We generate $Z^{(2)}$ by applying $\mathcal{P}_1^{-1}$ to the first of these, and $\mathcal{P}_2^{-1}$ to the second, and so our space includes $\operatorname{span}\{\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}\}$ and

$$\operatorname{span}\{\mathcal{P}_2^{-1}\mathcal{A}(\mathcal{P}_2^{-1} + \alpha\mathcal{P}_1^{-1})\mathbf{r}^{(0)})\}$$

for some fixed α (which comes from the orthogonalization step), but contains no component of $\mathcal{P}_2^{-1}\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}$. In summary,

$$\mathbf{x}^{(2)} - \mathbf{x}^{(0)} \in \operatorname{span}\{\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathbf{r}^{(0)}, \mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_1^{-1}\mathbf{r}^{(0)}, \mathcal{P}_2^{-1}\mathcal{A}(\mathcal{P}_1^{-1} + \alpha\mathcal{P}_2^{-1})\mathbf{r}^{(0)}\},$$

for some constant α. Since α comes from the orthogonalization step, it will be different for each matrix and each $\mathbf{r}^{(0)}$. This makes a general characterization of this space unobtainable. We mention that if different columns of V are chosen for the truncation strategy, we will still have a subspace of dimension four in the above analysis, but a different subspace than the one just described.

As already mentioned, there are many other possibilities for truncation. As can be seen from the above, simply changing the order of the preconditioners will dramatically change the space in which the approximation is found. In addition, we could pick a

truncation so that $Z^{(k)} \in \mathbb{R}^{n \times t^2}$ for $k > 2$, say, and this will give us a much richer space for relatively little increase in cost. We do not explore these possibilities here, but simply observe that the truncation scheme described above and exemplified in Algorithm 4 is often very good in practice; see Section 5.

3. Further aspects of MPGMRES. In this section we describe in more detail some aspects of the MPGMRES algorithm, both in the full and truncated versions. In particular, we address the question of whether the algorithm can break down, and also describe some issues related to implementation.

3.1. Breakdowns. It is well known that all breakdowns in standard GMRES – i.e., cases where the last subdiagonal entry of the matrix $\widetilde{H}_k$ is zero – are 'lucky' in that they occur only when the algorithm has converged to the exact solution. This is not the case with a multi-preconditioned algorithm, as there are cases – e.g., if $\mathcal{P}_1 = \mathcal{P}_2$, $\mathcal{P}_1^{-1} A \mathcal{P}_2^{-1} = \mathcal{P}_2^{-1} A \mathcal{P}_1^{-1}$ – where the multi-Krylov matrix formed by the elements of (2.10) will not be of full rank, and hence we may get a zero on the subdiagonal of $\widetilde{H}_k$ before reaching the exact solution. Here, however, we have the following result.

PROPOSITION 3.1. Consider Algorithm 3, and assume that after $s = n_{\widetilde{V}} = n_Z + \ell$ steps it has successfully generated a set of s linearly independent columns of $\widetilde{Z}_k$, i.e., we have a set of linearly independent search directions. In other words, assume that the search matrix $Z = (\widetilde{Z}_k)_{1:s}$ is full rank. Let $\mathbf{z}$ be the next search direction (either the next column of the kth matrix of search directions, $\widetilde{Z}_k$, or the first column of the next matrix of search directions, $\widetilde{Z}_{k+1}$). Then, $\mathbf{z} \in \mathcal{R}(Z)$ if and only if the matrix $\widehat{H}_{s+1}$ formed by the first $s+2$ columns and the first $s+1$ rows of $\widetilde{H}_k$ is rank deficient.

Proof: Define $V = (\widetilde{V}_{k+1})_{1:s+1}$ and $H = (\widetilde{H}_k)_{1:s+1,1:s}$. Then we have

$$\mathcal{A}Z = VH,$$

where V has orthonormal columns and H has full rank. Now suppose that $\mathbf{z} \in \mathcal{R}(Z)$. Then

$$\mathcal{A}\mathbf{z} \in \mathcal{R}(AZ) = \mathcal{R}(V).$$

Therefore, from Algorithm 3, we must have that the last subdiagonal entry of $H^{(k+1,k)}$ vanishes. Now pick any $\hat{\mathbf{v}}$ which is orthogonal to the columns of V. Then we have

$$A[Z \ \mathbf{z}] = [V \ \hat{\mathbf{v}}] \begin{bmatrix} H & \mathbf{h} \\ \mathbf{0}^T & 0 \end{bmatrix},$$

and hence

$$[V \ \hat{\mathbf{v}}]^T A[Z \ \mathbf{z}] = \begin{bmatrix} H & \mathbf{h} \\ \mathbf{0}^T & 0 \end{bmatrix}.$$

Since $[Z \ \mathbf{z}]$ has rank s and V, A have full rank, then $\widehat{H}_{s+1}$ must also have rank s, so $\mathbf{h} \in \mathcal{R}(H)$.

Conversely, if $\mathbf{z} \notin \mathcal{R}(Z)$, then the last subdiagonal entry of $H^{(k+1,k)}$ is nonzero, and hence if H is of full rank, then $\widehat{H}_{s+1}$ must also be. $\square$

In light of Proposition 3.2, the possible redundancy of vectors in $Z^{(k)}$ is not a problem, as we can simply monitor $H^{(k+1,k)}{}_{m+1,m}$, the subdiagonal entries of $H^{(k+1,k)}$, in Algorithm 3. If $|H^{(k+1,k)}{}_{m+1,m}|$ is smaller than some pre-defined tolerance and the updated part of $H^{(k+1,k)}$ is not full rank, then the current vector adds nothing

11

to the multi-Krylov subspace (2.10) and so it, along with the corresponding column of $\widetilde{H}_k$, can be discarded. Algorithm 5 is an updated version of Algorithm 3 which incorporates this update. We note that this situation is only likely to occur in certain circumstances where the preconditioners have special structure, and in general such issues will not occur.

Algorithm 5 MPGMRES: vectorized version with elimination of redundant search directions

---

   Choose $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
   $\beta = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
   $Z^{(1)} = \texttt{fullmultiprecondition}(\mathbf{r}_0)$
   $\widetilde{V} = \mathbf{v}^{(1)}$
   $n_{\widetilde{V}} = 1$ (no. of columns in $\widetilde{V}$)
   $n_{Z^{(1)}} =$ no. of columns in $Z^{(1)}$, $n_Z = n_{Z^{(1)}}$
   for $k = 1, \ldots,$ **until** convergence **do**
     for $\ell = 1 \ldots n_{Z^{(k)}}$ **do**
       $\mathbf{w} = \mathcal{A}{Z^{(k)}}_\ell$
       for $j = 1, \ldots, n_{\widetilde{V}}$ **do**
         $(\widetilde{H}_k)_{j,n_{\widetilde{V}}} = \mathbf{w}^T \widetilde{V}_j$
         $\mathbf{w} = \mathbf{w} - \widetilde{V}_j (\widetilde{H}_k)_{j,n_{\widetilde{V}}}$
       end for
       $(\widetilde{H}_k)_{n_{\widetilde{V}}+1,n_{\widetilde{V}}} = \|\mathbf{w}\|_2$
       $\widetilde{V} = [\widetilde{V} \; \mathbf{w}/(\widetilde{H}_k)_{n_{\widetilde{V}}+1,n_{\widetilde{V}}}]$
       $n_{\widetilde{V}} = n_{\widetilde{V}} + 1$
     end for
     $\mathbf{y}^{(k)} = \mathrm{argmin}\|\beta\mathbf{e}_1 - \widetilde{H}_k\mathbf{y}\|_2$
     $\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + [Z^{(1)} \cdots Z^{(i)}]\mathbf{y}^{(k)}$
     if $(\widetilde{H}_k)_{n_{\widetilde{V}}+1,n_{\widetilde{V}}} = 0$ **then**
       if residual small enough **then**
         lucky breakdown
       else
         remove this column of $Z^{(k)}$, $n_{Z^{(k)}} = n_{Z^{(k)}} - 1$
         return to top of loop
       end if
     else
       $Z^{(k+1)} = \begin{cases} \texttt{fullmultiprecondition}(\widetilde{V}_{n_Z+1:n_Z+n_{Z^{(k)}}}), \text{ or} \\ \texttt{truncmultiprecondition}(\widetilde{V}_{n_Z+1:n_Z+n_{Z^{(k)}}}) \end{cases}$
       $n_{Z^{(i+1)}} =$ no. of columns in $Z^{(i+1)}$, $n_Z = n_Z + n_{Z^{(i)}}$
     end if
   end for

The question arises: is it possible that the algorithm would break down by having a section of $\widetilde{H}_{k+1}$ full rank, but ${H^{(k+1,k)}}_{m+1,m} = 0$ for some m? The following proposition confirms that the answer is no.

PROPOSITION 3.2. *All breakdowns in Algorithm 5 (which always generates a full rank $\widetilde{H}_k$) are 'lucky', in the sense that if ${H^{(k+1,k)}}_{m+1,m} = 0$ then we have converged to the exact solution.*

Proof: We follow the arguments of [13, Proposition 2.2]. Suppose that $H^{(k+1,k)}{}_{m+1,m} = 0$ for some k, m, but the matrix

$$H := \left[\begin{array}{cc} (\widetilde{H}_k)_{1:k,1:k-1} & H^{(1:k,k)}{}_{1:k,1:m} \\ \mathbf{0}^T & H^{(k+1,k)}{}_{1:m+1,1:m} \end{array} \right]$$

has full rank. Say that $H \in \mathbb{R}^{s+1 \times s}$ for some s. Then the square matrix $\bar{H}$ formed by removing the last row of H is invertible. Then from (2.7) we have an Arnoldi-style decomposition

$$\mathcal{A}Z = \bar{V}\bar{H},$$

where $Z, \bar{V} \in \mathbb{R}^{n \times s}$ for some s. Then

$$\|\mathbf{r}^{(0)} - \mathcal{A}Z\mathbf{y}\|_2 = \|\beta\mathbf{e}_1 - \bar{H}\mathbf{y}\|_2,$$

where $\beta = \|\mathbf{r}^{(0)}\|_2$. Since $\bar{H}$ is nonsingular, this is minimized for $\mathbf{y} = \bar{H}^{-1}(\beta\mathbf{e}_1)$ with no error, meaning $\mathbf{x} = \mathbf{x}_0 + \hat{Z}\mathbf{y}$ is the solution of (1.1).

For the converse, suppose that $\mathbf{x}_k$ is exact and $(\widetilde{H}_k)_{i+1,i} \neq 0$ for $1 \leq i \leq s-1$. If $(\widetilde{V}_k)_s$ denotes the sth column of $\widetilde{V}_k$ in (2.7), then this can be re-written as

$$A(\widetilde{Z}_k)_{1:s} = (\widetilde{V}_k)_{1:s}H + (\widetilde{V}_k)_s\mathbf{e}_s^T.$$

Then we have that

$$\mathbf{0} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(k)} = (\widetilde{V}_k)_{1:s-1}[\beta\mathbf{e}_1 - \bar{H}\mathbf{y}] - (\widetilde{V}_k)_s\mathbf{e}_s^T\mathbf{y} \tag{3.1}$$

for some vector $\mathbf{y}$.

If $\mathbf{e}_s^T\mathbf{y} = 0$ (i.e., the last component of $\mathbf{y}$ is zero) then $\bar{H}\mathbf{y} = \beta\mathbf{e}_1$, and so back substitution give us that $\mathbf{y} = \mathbf{0}$, and hence $\mathbf{r}^{(0)} = \mathbf{0}$, so we must have started with the exact solution, which is not an interesting case. Therefore we can assume $\mathbf{e}_s^T\mathbf{y} \neq 0$.

Since $(\widetilde{V}_k)_s$ is orthogonal to the columns of $(\widetilde{V}_k)_{1:s-1}$, multiplying (3.1) on the left by $((\widetilde{V}_k)_{1:s})^T$ gives $\beta\mathbf{e}_1 - \bar{H}\mathbf{y} = \mathbf{0}$, and hence $(\widetilde{V}_k)_s = \mathbf{0}$. The only way this can happen is if $(\widetilde{H}_k)_{s+1,s} = 0$. $\square$

In the next section we see how this result helps us in the implementation of a practical algorithm.

3.2. Implementation issues. As with standard GMRES the least-squares problem (2.8) can be solved efficiently with Givens rotations. Using this technique we transform the least-squares problem into an equivalent one of the form

$$\mathbf{y}^{(k)} = \text{argmin}\|\hat{\mathbf{b}} - \widetilde{R}_k\mathbf{y}\|,$$

where $\hat{\mathbf{b}} \in \mathbb{R}^{k+1}$, and $\widetilde{R}_k \in \mathbb{R}^{k+1 \times k}$ is upper triangular; see, e.g., [14, Section 6.5.3]. If we use this method the residual at step k is simply the $(k+1)$st entry of $\hat{\mathbf{b}}$, and so there is no need to explicitly form the current approximation of the solution at each step in order to test convergence.

Since the matrix $\widetilde{H}_k$ in MPGMRES is upper Hessenberg, the Givens rotations are applied in the same way as standard GMRES. The main difference in the implementation for MPGMRES is that – as described in the last section – it is possible to get a subdiagonal entry of $\widetilde{H}_k$ which is zero while the algorithm has not converged to the exact solution. We can use the result of Proposition 3.2 to test the rank in

13

Algorithm 5; if $H^{(k+1,k)}{}_{m+1,m} = 0$ for some k, m, then we can look at the size of the residual. If this is smaller than the supplied tolerance then we have converged, and so there is nothing to prove. If not, then by Proposition 3.2 the updated part of $\widetilde{H}_k$ must be rank deficient, and therefore we do not need to form the next column of $V^{(k)}$. In this case we can simply remove the current column of $Z^{(k)}$, which will be linearly dependent on the other columns, and carry on with the algorithm. Note that this process can be done *without explicitly computing the rank*.

An alternative approach would be to use a block algorithm, as in Algorithm 1; in this case we need to use a rank-revealing QR factorization [5, Section 5.4.1] to detect the rank of the current $V^{(k)}$. This form of the algorithm could be advantageous in practice, since it can be implemented using Level 3 BLAS routines, as opposed to Algorithm 5 which heavily uses Level 1 BLAS. The details are somewhat technical, and we do not expand on this approach here. We found that the vectorized version is more efficient in a MATLAB implementation.

In a truncated algorithm a column of $V^{(k)}$ may be linearly dependent on the others, and subsequently removed. Now the $V^{(k)}$ we pass to the multi-preconditioning routine (e.g., Algorithm 4) may have $t_0 < t$ columns. It is advantageous to keep $Z^{(k+1)} \in \mathbb{R}^{n \times t}$, and to use all t preconditioners at each step, so in this case we simply re-apply the chosen method of choosing columns of $V^{(k)}$, starting from $\mathcal{P}_{t_0+1}$ instead of $\mathcal{P}_1$, until we have generated t columns for $Z^{(k+1)}$. In particular, in the case of the standard truncation described in Algorithm 4 we have

$$Z_i^{(k+1)} = \begin{cases} \mathcal{P}_i^{-1}(V^{(k)})_i, & i = 1, \ldots, t_0, \\ \mathcal{P}_i^{-1}(V^{(k)})_{i-t_0}, & i = t_0 + 1, \ldots, 2t_0, \text{ and so on.} \end{cases}$$

Other truncations can be treated similarly.

In Algorithm 5 we store three potentially large matrices: the basis matrix, $\widetilde{V}_k$; $\widetilde{H}_k$, or in a practical code the upper-triangular matrix $\widetilde{R}_k$; and also the search matrix $\widetilde{Z}_k$, as in FGMRES. If storage cost is an issue it is possible to adapt the algorithm so that only $\widetilde{V}_k$, $\widetilde{R}_k$, and the current matrix of search directions, $Z^{(k)}$, are stored. This can be achieved by also saving the indices of columns of $\widetilde{Z}_k$ which are formed by applying each of the preconditioners. We can then use the fact that

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \sum_{i=1}^{d} (\widetilde{Z}_k)_i \mathbf{y}^{(k)}{}_i$$

$$= \mathbf{x}^{(0)} + \mathcal{P}_1^{-1} \sum_{i=1}^{d_1} (\widetilde{V}_k)_i \mathbf{y}^{(k)}{}_{\pi_1(i)} + \cdots + \mathcal{P}_t^{-1} \sum_{i=1}^{d_t} (\widetilde{V}_k)_i \mathbf{y}^{(k)}{}_{\pi_t(i)},$$

where $\widetilde{Z}_k \in \mathbb{R}^{n \times d}$ contains d_j columns derived from $\mathcal{P}_j$. The $\pi_j(i)$ are permutation operators such that the set $\{(\pi_1(i))_{i=1\ldots d_1}, \ldots, (\pi_t(i))_{i=1\ldots d_t}\}$ contains the integers from 1 to d. This can be evaluated by a simple call to the multi-preconditioning routine, analogous to the standard right preconditioned GMRES algorithm. We note that this approach is slightly more expensive in terms of operation count than when we save $\widetilde{Z}_k$, however the saving of essentially half the storage requirements is substantial.

We also note the obvious potential for parallelization in this code. Each preconditioner solve can be performed on a separate processor, and so significant savings can be made by taking advantage of this.

4. Related algorithms. In this section we briefly describe two algorithms which have appeared in the literature which are related to the MPGMRES algorithm described here: flexible GMRES with cycling preconditioners and Krylov multi-splittings.

Both of these methods can be thought of in terms of non-optimal truncations of the full MPGMRES algorithm.

4.1. Flexible GMRES with cycling multiple preconditioners. The flexible GMRES method of Saad [13] allows us to use a different preconditioner at each iteration. As already mentioned, the key idea for this method is the use of a separate matrix Z_k to store the columns $\mathcal{P}^{-1}V_k$, giving rise to the Arnoldi-type decomposition (2.5). In this manner, the application of different preconditioners $\mathcal{P}_i^{-1}v_j$ can be used and stored in the jth column of Z_k. We note that in this case, $\mathcal{R}(V_k)$ is not strictly speaking a Krylov subspace, but nevertheless is the space where the approximation is sought [18, 19].

In terms of the multi-preconditioning paradigm considered in this paper, one could cycle with FGMRES through the available preconditioners in some prescribed order. This strategy in combination with FGMRES was in fact proposed by Rui, Yong, and Chen in the context of electromagnetic wave scattering problems [12], although the numerical results reported were not as competitive as the use of a single preconditioner. This is consistent with the experiments we report in Section 5.

We use this approach in our experiments for comparison with MPGMRES; for completeness the process is presented in Algorithm 6.

Algorithm 6 FGMRES with cycling preconditioners

   Choose $\mathbf{x}^{(0)}$, $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
   $\beta = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/\beta$
   for $i = 1, \ldots$ **until** convergence **do**
      $s = (i - 1 \bmod t) + 1$
      $\mathbf{z}^{(i)} = \mathcal{P}_s^{-1}\mathbf{r}_k$
      $\mathbf{w} = A\mathbf{z}^{(i)}$
      for $j = 1, \ldots, i$ **do**
         $h_{j,i} = \mathbf{w}^T\mathbf{v}^{(j)}$
         $\mathbf{w} = \mathbf{w} - h_{j,i}\mathbf{v}^{(j)}$
      end for
      $h_{i+1,i} = \|\mathbf{w}\|$
      $\mathbf{v}^{(i+1)} = \mathbf{w}/h_{i+1,i}$
      $\mathbf{y}^{(i)} = \text{argmin}\|\beta\mathbf{e}_1 - H_i\mathbf{y}\|_2$
      $\mathbf{x}^{(i)} = \mathbf{x}^{(0)} + [\mathbf{z}^{(1)} \ldots \mathbf{z}^{(i)}]\mathbf{y}^{(i)}$
   end for

As a way of comparison with MPGMRES and its truncated version, we observe that FGMRES with cycling multiple preconditioners uses a single preconditioner in each step, while MPGMRES uses a (possibly optimal or near-optimal) linear combination of the preconditioners. Numerical comparisons are reported in Section 5.

It is also possible to view FGMRES with cycling multiple preconditioners as a special case of truncated MPGMRES, in which only one choice of preconditioner is taken at each step, and this choice follows a prescribed order.

4.2. Multi-splittings. Given a set of preconditioners $\mathcal{P}_i$, $i = 1, \ldots, t$, and a corresponding set of positive semi-definite diagonal weighting matrices $\mathcal{D}_i$ such that $\sum_i \mathcal{D}_i = I$, the multi-splitting algorithm [9] is defined as the iterative method governed

by the simple (stationary) iteration

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \sum_{i=1}^{t} \mathcal{D}_i \mathcal{P}_i^{-1} \mathbf{r}^{(k)}, \qquad (4.1)$$

where as usual $\mathbf{r}^{(k)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(k)}$. It is immediate to consider now the (nonsingular) multi-splitting preconditioner $\mathcal{B}^{-1} = \sum_{i=1}^{t} \mathcal{D}_i \mathcal{P}_i^{-1}$ and use it for GMRES [8]. One can think of multi-splittings as having been devised with the same philosophy of MPGMRES of combining a number of preconditioners (in this case derived from splittings).

Note that the multi-splitting algorithm – as with MPGMRES – is trivially parallelized, as the solves with $\mathcal{P}_i$ can be carried out simultaneously. This is in contrast to Algorithm 6, FGMRES with cycling multiple preconditioners, in which we require the previous iteration to have finished before solving with the next preconditioner. However, in this method we have to define a weighting of the preconditioners *a priori*, whereas in MPGMRES a weighting which is in some sense optimal is computed as part of the algorithm. Furthermore, O'Leary and White [9] show that one can construct weighting matrices such that the multi-splitting fails to converge, even if the underlying iterations are convergent by themselves. The *a priori* choice of $\mathcal{D}_i$ is therefore important, and might be highly problem-dependent.

We mention that Huang and O'Leary [8] studied further this Krylov multi-splitting (KMS) algorithm and consider more than one (inner) simple stationary iteration in the preconditioning step, say m steps. Their goal was to keep the tasks in each of t processors working. An additional task (or processor) collects the information and performs the residual minimization operation over a space consisting of the sum of t Krylov subspaces of dimension m generated from different initial vectors as the iterations continue; see [8] for more details.

5. Applications and Numerical Experiments. In this section we apply the algorithms in this paper to some numerical examples of varying degrees of realism. We start with the academic example to highlight the fact the the space in MPGMRES is richer, and then look at more realistic problems: the solution of a problem from PDE-constrained optimization and a domain decomposition preconditioner.

In the following examples we compare iteration counts; it should be noted that an iteration is significantly more expensive for MPGMRES than for right-preconditioned GMRES or FGMRES, since we apply more than one preconditioner and have a larger set of vectors to orthogonalize. Table 5 compares the number of matrix-vector products, inner products, and preconditioner solves for Algorithm 3 in its full and truncated versions (i.e., with the two subroutines given by Algorithms 2 and 4), and Algorithm 6. Clearly the full MPGMRES algorithm quickly gets impractical, but the truncated version remains viable for small t, especially when using a parallel implementation. For example, if we have two preconditioners, then we have twice the number of matrix-vector products and preconditioner solves and about four times as many inner products. Our results were run with a sequential code; in a parallel implementation we could solve with each preconditioner on a separate processor. Since the preconditioner solve is usually the most expensive part of the iteration, this is a significant saving.

5.1. A random example with a special right-hand side. Consider a matrix $\mathcal{A}=$ `randn(100,100)` in MATLAB, and let $\mathcal{P}_1$ and $\mathcal{P}_2$ also denote matrices with random, normally distributed entries. Take the right-hand side $\mathbf{b}$ as an eigenvector

	Matrix-vector products	inner products	preconditioner solves[†]
MPGMRES	t^k	$\frac{t^k-1}{t-1} + \frac{t^{2(k-1)}+3t^{k-1}}{2}$	t^k
tMPGMRES	t	$(k-\frac{3}{2})t^2 + \frac{5}{2}t$	t
FGMRES	1	$k+1$	1

† can be easily parallelized.

TABLE 5.1

Number of matrix-vector products, inner products and preconditioner solve at the kth iteration when using t preconditioners. Full and truncated versions of MPGMRES, and FGMRES with cycling multiple preconditioners.

of $\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}^{-1}\mathcal{A}$. Figure 5.1 shows the results for MPGMRES, truncated MPGMRES, standard GMRES with $\mathcal{P}_1$ and $\mathcal{P}_2$ applied on the right, and FGMRES cycling between the preconditioners.
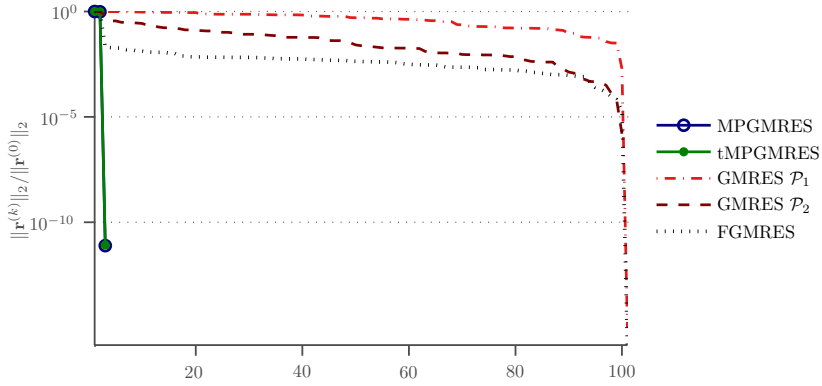

FIG. 5.1. *Convergence curves for a random example where the right-hand side is a specific eigenvector*

As can be seen in Figure 5.1, for this example the multi-preconditioned variants of GMRES performs significantly better than the other methods, and in fact truncating the space makes little difference in the convergence history.

Of course, in practice we do not pick the right-hand side for a given preconditioner, as we have here. However, for a given right-hand side we do have the freedom to pick the preconditioners. In standard Krylov subspace methods we can only pick one preconditioner, and therefore we may be forced to ignore some important part of the matrix to get a preconditioner which is easy to invert. In the proposed method we have more degrees of freedom, which could be useful in real-life applications.

5.2. PDE-constrained optimization. Many real-world problems can be formulated as PDE-constrained optimization problems; see e.g. [21, 7] and the references therein. Here we consider the following model problem.

$$\min_{y,u} \frac{1}{2}||y-\hat{y}||_2^2 + \frac{\beta}{2}||u||_2^2 \tag{5.1}$$

$$\text{s.t.} \quad -\nabla^2 y = u \text{ in } \Omega$$
$$y = f \text{ on } \partial\Omega.$$

17

Here $\hat{y}$ is some pre-determined optimal state, and we want the system to get to a state y as close to this state as possible – in the sense of minimizing the given cost functional – while satisfying Poisson's equation in some domain Ω. The mechanism we have of changing y is by varying the right-hand side of the PDE, u, which is called the control in this context. Note that the norm of the control appears in the cost functional, along with a Tikhonov regularization parameter, β, to ensure that the problem is well-posed.

If we discretize the problem using finite elements, then the minimum of the discretized cost functional is found by solving the linear system of equations [10], [17]

$$
\begin{bmatrix} \beta Q & 0 & -Q \\ 0 & Q & K \\ -Q & K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{d} \end{bmatrix}, \tag{5.2}
$$

where Q is a mass matrix, K is a stiffness matrix and $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{p}$ represent the discretized control, state and Lagrange multipliers respectively. This matrix is typically very large and sparse, and it is generally solved iteratively.

It was shown in [10] that two preconditioners that are optimal in terms of the mesh size taken are

$$
\mathcal{P}_{bd} := \begin{bmatrix} \beta Q & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & KQ^{-1}K \end{bmatrix} \text{ and } \mathcal{P}_{con} := \begin{bmatrix} 0 & 0 & -Q \\ 0 & \beta KQ^{-1}K & K \\ -Q & K & 0 \end{bmatrix}.
$$

Although these preconditioners perform well for moderately small values of β – say $\beta > 10^{-4}$ – the clustering of the generalized eigenvalues of the preconditioned system deteriorates as $\beta \to 0$ [10, Corollary 3.3 and Corollary 4.4].

EXAMPLE 5.1. *We discretize the control problem* (5.1) *on the domain* $\Omega = [0,1]^2$ *using Q_1 finite elements with a uniform mesh size of $h = 2^{-7}$. We take the desired state as*

$$
\hat{y} = \begin{cases} (2x-1)^2(2y-1)^2 & \text{if } (x,y) \in [0,\frac{1}{2}]^2 \\ 0 & \text{otherwise} \end{cases}.
$$

We apply the preconditioners $\mathcal{P}_{bd}$, $\mathcal{P}_{con}$, with multi-preconditioned GMRES (truncated and not), standard GMRES, and FGMRES with cycling. The results are given in Figure 5.2.

Figure 5.2 shows that, although neither of the preconditioners $\mathcal{P}_{bd}$ or $\mathcal{P}_{con}$ is effective for small β, together they have all the components of the large system, and the combination of both of them is an excellent method for the solution of this system at small β. For $\beta > 10^{-4}$ – the range in which the preconditioners were designed to be effective – there is no benefit to using MPGMRES, as the iteration counts are around the same, but one iteration of MPGMRES is much more expensive than one of GMRES. In fact, truncated MPGMRES essentially stagnates for $\beta = 10^{-2}$, whereas for smaller β it behaves essentially as the full MPGMRES algorithm does. Note that FGMRES with cycling preconditioners is not competitive for this example.

5.3. Domain decomposition. One common method of preconditioning linear systems which arise from the solution of partial differential equations (PDEs) is domain decomposition. These methods work by partitioning the domain into small (possibly overlapping) subdomains, and then (approximately) solving the restriction of the PDE to that subdomain; see, e.g., [20]. As pointed out in [2], this is a natural
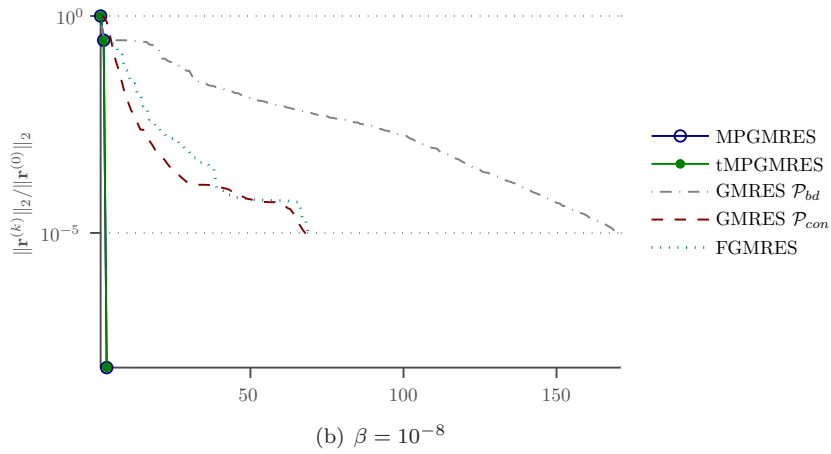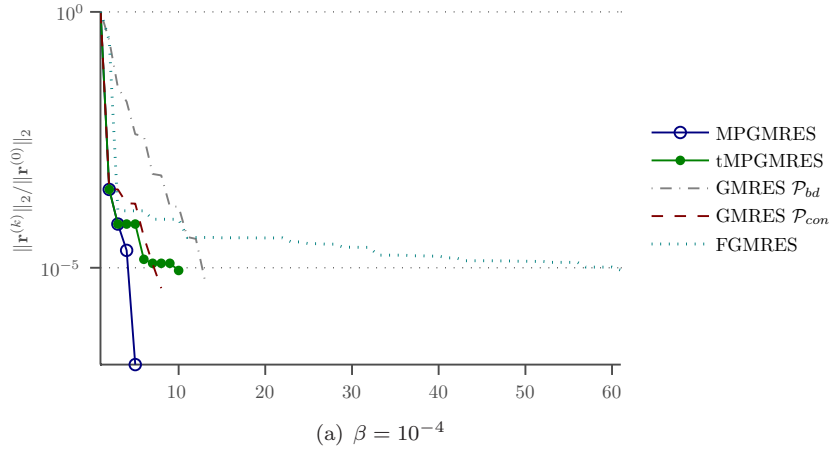
18

(a) $\beta = 10^{-4}$

(b) $\beta = 10^{-8}$

Fig. 5.2. *Convergence curves for solving the optimal control problem with multi-preconditioned GMRES, standard GMRES and FGMRES with cycling*

application for multi-preconditioned algorithms, as we take each solve on a subdomain as a separate preconditioner. These preconditioners will be singular, but when taken together will span the whole space, so singularity does not pose a difficulty.

5.3.1. Additive Schwarz. Consider the following advection-diffusion equation on $\Omega = [0, 1]^2$.

$$-\nabla^2 u + \boldsymbol{\omega} \cdot \nabla u = f \qquad \text{in } \Omega \tag{5.3}$$

$$u = 0 \qquad \text{on } \partial\Omega. \tag{5.4}$$

Upon discretization by finite differences we get the matrix equation

$$A\mathbf{u} = \mathbf{f},$$

where A is a real positive, but nonsymmetric, matrix.

19

The domain Ω can be divided into t (possibly overlapping) subdomains, $\Omega_1, \ldots, \Omega_t$, and we can define restriction operators R_i, $i = 1, \ldots, t$ which restrict the PDE to the ith subdomain. The restriction of A to the ith subdomain is therefore given by $R_i A R_i^T$.

We can use this construction to define the additive Schwarz preconditioner, the inverse of which is defined as

$$M^{-1} = \sum_{i=1}^{t} R_i^T (R_i A R_i^T)^{-1} R_i. \tag{5.5}$$

A simple iteration based on this iteration will, in general, be nonconvergent, but it can be used as a preconditioner for a Krylov subspace method. In practice this ideal preconditioner is generally replaced by

$$M^{-1} = \sum_{i=1}^{t} R_i^T M_i^{-1} R_i,$$

where M_i denotes an approximation to the PDE on the ith subdomain, obtained, for example, by using a multigrid method. This preconditioner is very well suited to solving large problems with a parallel architecture, as each solve on a subdomain can be computed on a separate processor.

This type of preconditioner is well suited to a multi-preconditioned approach. We can take t preconditioners

$$\mathcal{P}_i = R_i^T M_i^{-1} R_i$$

and the multi-preconditioned GMRES algorithm will calculate the ideal weights to assign to each subdomain, giving an effective preconditioner. Moreover, we have the following result.

PROPOSITION 5.2. *Suppose we define a mesh on Ω of, say, $2^k \times 2^k$ points, and partition Ω into subdomains of a constant size, say containing $2^m \times 2^m$, $m < k$, mesh points. Then irrespective of the value of k, we get convergence in a number of iterations dependent only on the number of mesh points in the subdomains, i.e., independent of m.*

Proof. The number of preconditioners we have would be $2^{2(k-m)}$, and so the dimension of the space in which we look for approximate solutions will increase by $2^{2(k-m)}$ vectors at each step. Recall that the order of A is 2^{2k}, hence we will get convergence to the exact solution when the dimension of the space reaches 2^{2k}, i.e., in n iterations, where n satisfies

$$2^{2k} = n 2^{2(k-m)},$$
$$\text{i.e., for} \quad n = 2^{2m}.$$

Therefore we will get convergence in at most 2^{2m} iterations, irrespective of the size of the original domain Ω. $\square$

We remark that in the case of non-overlapping subdomains, the use of multiple preconditioning need not take more storage. Indeed, without loss of generality, we can assume that $R_i A R_i^T$ is the $n_i \times n_i$ principal submatrix of A, with $\sum n_i = n$. Then $\mathcal{P}_1^{-1} \mathbf{r}^{(0)}$ has only nonzeros in the first n_1 entries, $\mathcal{P}_2^{-1} \mathbf{r}^{(0)}$ has only nonzeros in the next n_2 entries, and so on. Thus, we can store $Z^{(1)}$ as a single vector containing these small vectors of length n_i stacked. Similarly, the product $\mathcal{A}Z^{(1)}$ can be stored

in a single vector, and the orthogonalization can be done implicitly by blocks as well, resulting in the fact that $V^{(2)}$ can be stored in one column vector as well (after t normalizations). The process is repeated, and thus $Z^{(2)}$ can be stored in a single column as well, and in general, all matrices $V^{(k)}$, $Z^{(k)}$ can be stored as a column each. When we have overlap, the increase in storage can be contained as not to be larger than the total amount of overlap. More precisely, if n_i is the number of nodes in the ith subdomain, one can store $V^{(k)}$, $Z^{(k)}$ using a single vector of length $\sum n_i$, which is now moderately larger than n.

Note also that in this setup, our proposed multi-preconditioning consists of replacing the Additive Schwarz preconditioning (5.5) with an optimal linear combination of the same summands, and that this linear combination changes from step to step.

EXAMPLE 5.3. *Consider the advection-diffusion equation (5.3) with* $\omega = 10[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$. *We discretize this using finite differences, with central differencing for the advection term, and take the right-hand side as the vector of all ones. We apply a standard additive Schwarz preconditioner to GMRES, with the sub-domain solves performed using Matlab's* `backslash` *command. The multi-preconditioned equivalents, $\mathcal{P}_i$, as described above are applied with full and truncated MPGMRES. Since the preconditioners have low rank, in order to facilitate mixing here for the truncated method we generate $Z^{(k+1)}$ by applying Algorithm 2 to the sum of the columns of $V^{(k)}$, as described in section 2.3. The convergence curves are given in Figure 5.3.*

As we see from Figure 5.3, the multi-preconditioned approach is significantly better in terms of iteration count than standard GMRES with a domain decomposition preconditioner if the domain is split into a large number of subdomains. This confirms our expectation that using optimal linear combinations instead of (5.5) potentially yields significant savings. This approach is competitive since the cost of applying multiple preconditioners is approximately the same as that of applying the additive Schwarz preconditioner, and the storage needs are similar. Depending on the implementation, the only extra cost might come from the increased number of inner products which need to be computed (see Table 5), but if the subdomains are sufficiently large, so that a preconditioner solve has significant cost, then we expect the truncated multi-preconditioned method to outperform the standard additive Schwarz preconditioner.

5.3.2. Restricted Additive Schwarz. The subdomains in the additive Schwarz method will often overlap. We can account for this overlap in the restriction operator by using the notation $R_{i,\delta}$ to denote the restriction to the ith subdomain, which has δ nodes overlapping with the neighboring domains. The method of Restricted Additive Schwarz (RAS), developed by Cai and Sarkis [3], has been shown to be more efficient than Additive Schwarz, both in terms of iteration count and in communication times on a parallel architecture.
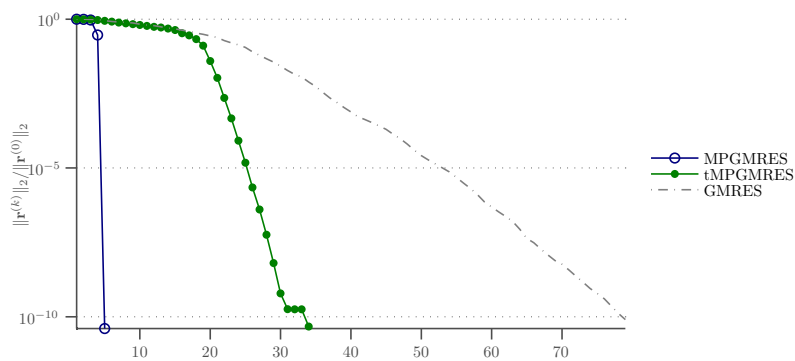
In the notation above, the Additive Schwarz preconditioner with an overlap of size δ can be written as

$$M_{AS,\delta}^{-1} = \sum_{i=1}^{t} R_{i,\delta}^T (R_{i,\delta} A R_{i,\delta}^T)^{-1} R_{i,\delta}.$$

When using the RAS method we use the same restriction operator, but the prolongation operator is that which would would be applied if there was no overlap. We
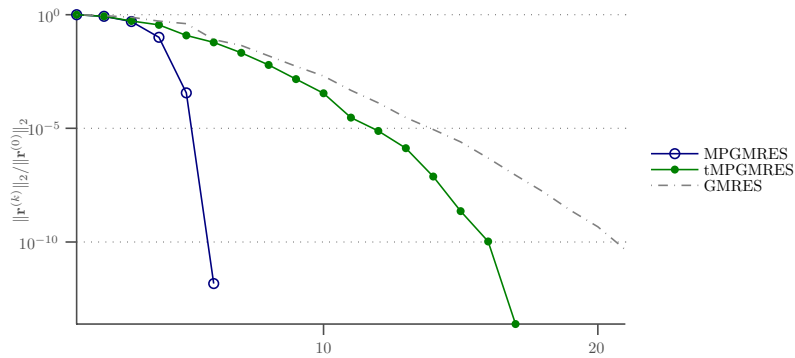
(a) $N = 2^4$

(b) $N = 2^6$

FIG. 5.3. *Convergence curves for solving the domain decomposition problem with multi-preconditioned GMRES and standard GMRES. Domain split into smaller grids of $2^3 \times 2^3$ mesh points.*

therefore get the preconditioner

$$M_{RAS,\delta}^{-1} = \sum_{i=1}^{t} R_{i,0}^{T}(R_{i,\delta}AR_{i,\delta}^{T})^{-1}R_{i,\delta}. \tag{5.6}$$

This method can be thought of as a multisplitting algorithm (see section 4.2), and in this context a convergence theory was given in [4].

Again, this is an ideal candidate for a multi-preconditioned method, as shown in the example below.

EXAMPLE 5.4. *Consider the setup as in Example 5.3. Figure 5.4 shows the plots comparing RAS with an overlap of 2 nodes, additive Schwarz with the same overlap, and the multi-preconditioned equivalents.*

As we remarked earlier, the MPGMRES method can be implemented in such a way that it uses essentially the same storage and number of operations as that of applying the RAS preconditioner. Here, it chooses in each step an optimal linear combination of the summands of (5.6). Figure 5.4 shows that the multi-preconditioned approach is again an improvement over standard GMRES in terms of iteration counts
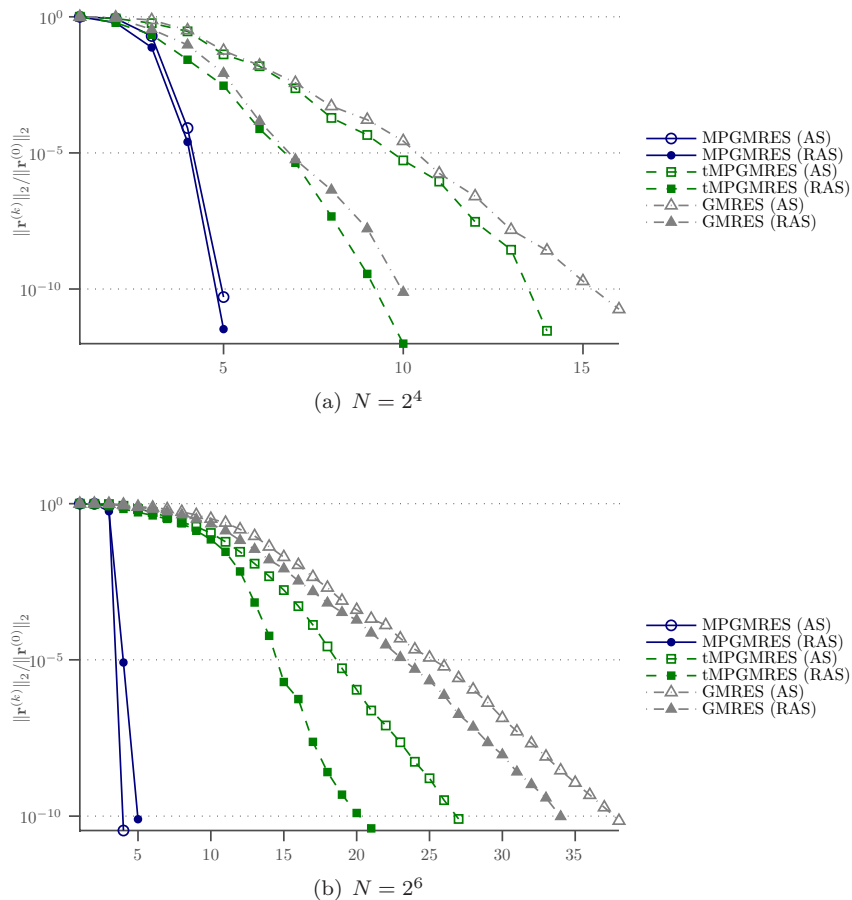
Fig. 5.4. *Convergence curves for solving the domain decomposition problem with multi-preconditioned GMRES and standard GMRES. Domain split into smaller grids of $2^3 \times 2^3$ mesh points.*

with an RAS preconditioner if the domain is split into a large number of subdomains.

6. Conclusions. The multi-preconditioned GMRES algorithm (MPGMRES) is an extension of the standard right-preconditioned GMRES algorithm which allows the application of two or more preconditioners, combining them in an optimal way. We have presented two new algorithms: a theoretical full multi-preconditioned GMRES algorithm, which seeks approximate solutions in a space whose dimension grows exponentially with each iteration number; and a practical truncated MPGMRES algorithm which looks for solutions in a subspace that grows linearly with problem size.

We have presented some theoretical results concerning the new algorithm, and described some issues related to practical implementation. Here we have characterized the generalized Krylov subspace where the iterates are generated, and have discussed the issue of breakdowns and how to handle them.

The numerical examples illustrate that there are certain situations where using two or more preconditioners is significantly better than using just a single one. These experiments indicate the potential of MPGMRES, especially for problems where finding an optimal preconditioner has proved elusive.

REFERENCES

[1] Michele Benzi, *Preconditioning techniques for large linear systems: A survey*, Journal of Computational Physics **182** (2002), 418–477.

[2] Robert Bridson and Chen Greif, *A multipreconditioned conjugate gradient algorithm*, SIAM Journal on Matrix Analysis and Applications **27** (2006), no. 4, 1056–1068.

[3] Xiao-Chuan Cai and Marcus Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM Journal on Scientific Computing **21** (1999), 792.

[4] Andreas Frommer and Daniel B. Szyld, *An algebraic convergence theory for restricted additive Schwarz methods using weighted max norms*, SIAM Journal on Numerical Analysis **39** (2001), 463–479.

[5] Gene H. Golub and Charles F. van Loan, *Matrix computations*, 3rd ed., The Johns Hopkins University Press, 1996.

[6] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM Journal on Matrix Analysis and Applications **17** (1996), no. 3, 465–469.

[7] Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich, *Optimization with pde constraints*, Mathematical Modelling: Theory and Applications, Springer, 2008.

[8] Chiou-Ming Huang and Dianne P. O'Leary, *A Krylov multisplitting algorithm for solving linear systems of equations*, Linear algebra and its applications **194** (1993), 9–29.

[9] Dianne P. O'Leary and Robert E. White, *Multi-splittings of matrices and parallel solution of linear systems.*, SIAM Journal on Algebraic and Discrete Methods **6** (1985), no. 4, 630–640.

[10] Tyrone Rees, H. Sue Dollar, and Andrew J. Wathen, *Optimal solvers for PDE-constrained optimization*, SIAM Journal on Scientific Computing **32** (2010), no. 1, 271–298.

[11] Axel Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Mathematics of Computation **33** (1979), no. 146, 680–687.

[12] Ping-Linag Rui, H. Yong, and Ru-Shan Chen, *Multipreconditioned GMRES method for electromagnetic wave scattering problems*, Microwave and Optical Technology Letters **50** (2008), no. 1, 150–152.

[13] Yousef Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific Computing **14** (1993), 461–469.

[14] _____, *Iterative methods for sparse linear systems*, Society for Industrial Mathematics, 2003.

[15] Yousef Saad and Martin H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing **7** (1986), no. 3, 856–869.

[16] Marcus Sarkis and Daniel B. Szyld, *Optimal left and right additive Schwarz preconditioning for minimal residual methods with Euclidean and energy norms*, Computer Methods in Applied Mechanics and Engineering **196** (2007), 1612–1621.

[17] Joachim Schöberl and Walter Zulehner, *Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems*, SIAM Journal on Matrix Analysis and Applications **29** (2007), no. 3, 752–773.

[18] Valeria Simoncini and Daniel B. Szyld, *On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods*, SIAM Review **47** (2005), 247–272.

[19] _____, *Recent computational developments in Krylov subspace methods for linear systems*, Numerical Linear Algebra with Applications **14** (2007), 1–59.

[20] Andrea Toselli and Olof B. Widlund, *Domain decomposition methods - algorithms and theory*, Springer Series in Computational Mathematics, vol. 34, Springer, Berlin and Heidelberg, 2005.

[21] Fredi Tröltzsch, *Optimal control of partial differential equations: Theory, methods and applications*, American Mathematical Society, Providence, RI, 2010.