A block Recycled GMRES method with investigations into aspects of solver performance

Michael L. Parks, Kirk M. Soodhalter,
and Daniel B. Szyld

Report 16-04-04
April 2016. Revised August 2017 and June 2025

Department of Mathematics
Temple University
Philadelphia, PA 19122

This report is available in the World Wide Web at
`http://faculty.cst.temple.edu/~szyld`
and also in the arxiv at `http://arxiv.org/abs/1603.08970`

A block Recycled GMRES method with investigations into aspects of solver performance

Michael L Parks[1], Kirk M Soodhalter[2*], Daniel B Szyld[3]

[1]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA.
[2]School of Mathematics, Trinity College Dublin, College Green, Dublin 2, Ireland.
[3]Department of Mathematics, Temple University, Philadelphia, PA, 19147, USA.

*Corresponding author(s). E-mail(s): ksoodha@maths.tcd.ie;
Contributing authors: parksml@ornl.gov; szyld@temple.edu;

Abstract

We propose a block Krylov subspace version of the GCRO-DR method proposed in [Parks et al.; SISC 2005], which is an iterative method allowing for the efficient minimization of the the residual over an augmented Krylov subspace. We offer a clean derivation of our proposed method and discuss methods of selecting recycling subspaces at restart as well as implementation decisions in the context of high-performance computing. Numerical experiments are split into those demonstrating convergence properties and those demonstrating the data movement and cache efficiencies of the dominant operations of the method, measured using processor monitoring code from Intel.

Keywords: Krylov subspace methods, deflation, subspace recycling, block Krylov methods, high-performance computing

MSC Classification: 65F10 , 65N12 , 15B57 , 45B05 , 45A05

Funding: This research was supported in part by the U.S. National Science Foundation under grant DMS-1418882.

1 Introduction

We explore the efficient solution of a sequence of linear systems

$$\boldsymbol{A}^{[i]}\left(\boldsymbol{X}_0^{[i]} + \boldsymbol{T}^{[i]}\right) = \boldsymbol{B}^{[i]}, \tag{1}$$

where the coefficient matrices $\boldsymbol{A}^{[i]} \in \mathbb{C}^{n\times n}$ are assumed to be non-Hermitian and the right-hand sides $\boldsymbol{B}^{[i]} \in \mathbb{C}^{n\times p}$ may or may not also change with i. For many applications, the matrices $\boldsymbol{A}^{[i]}$ are large and sparse. The block $\boldsymbol{X}_0^{[i]} \in \mathbb{C}^{n\times p}$ is the initial approximations to the solution of system i and $\boldsymbol{T}^{[i]}$ is the corresponding initial error.

At the core of many problems in the computational sciences is the need to solve large, sparse linear systems. It is often the case that one must solve a sequence of systems, and these systems are somehow related. Examples include: uncertainty quantification [1, 2], Newton-like iterations from, e.g., a density functional theory computations; see [3], topology optimization [4], the modeling of crack propagation in materials [5], and tomography [6]. This challenge often includes solving for more than one right-hand side for each i. For each coefficient matrix, one could simply treat each right-hand side in sequence. However, it is often more efficient to take advantage of the underlying structure of all problems combined. Block Krylov subspace iterative methods [7, 8] were proposed to solve systems with many right-hand sides, and such methods were proposed to accelerate convergence even when there is is only one right-hand side; see, e.g., [7, 9–11]. Subspace recycling techniques [5] were proposed to take advantage of relationships between sequences of coefficient matrices. It is therefore natural to combine these two strategies to take advantage of their individual benefits as well as any synergistic interactions that arise.

It should be noted that this paper is a refinement of a technical report published in 2016 [12], which was derived from work in the PhD thesis [13]. We describe the original development of the combination these two techniques (block Krylov subspace methods and recycling), which led to implementations in Trilinos [14] in 2011 and an accompanying Matlab [15] implementation. We justify the utility of the resulting method not just with steeper convergence curves but also with performance experiments. Specifically, we measure the run-times and cache efficiency of key computational kernels to demonstrate the utility of such block methods.

The rest of this paper is organized as follows. In Section 2, we describe the problem being solved and discuss what has been investigated in the literature. In Section 3, we briefly review Krylov subspace methods and their generalization to the block setting as well as a framework for understanding subspace augmentation methods, including those employing a recycling strategy. In Section 4, we extend the recycled Gmres method to the block Krylov subspace setting. We further describe implementation decisions meant to improve the data movement efficiency of the method. In Section 5, we discuss convergence properties and theory of this method. In Section 6, we present experiments. These are divided into two types. The first includes simple convergence experiments, demonstrating the competitiveness of these methods for large-scale problems. We then present measurements of how data is moved to and used on the processor for the dominant operations of the method.

2 Background

Krylov subspace iterative methods are a standard tool for solving sparse systems such as those arising in (1). In this work, we consider solving (1) in the case that the number of right-hand sides $p > 1$, but we also explore the utility of using block Krylov subspace techniques for the case that $p = 1$. Although block Krylov subspace methods (cf. Section 3.1) generally are proposed for the case that the number of right-hand sides (i.e., the *block size*) $p > 1$ and these techniques can also be used to accelerate convergence in the single-vector case that $p = 1$, first suggested in [7] and elaborated upon in [10] and also used in, e.g., [9, 11]. See, e.g., [16], for a nice introduction to the topic.

We describe the theory behind the implementation of the block GCRO-DR method from [14, 15], which combines block GMRES with the recycling-based augmentation scheme GCRO-DR. In the case $p = 1$, GCRO-DR was introduced [5] for the treatment of sequences of "slowly-changing" linear systems. The expression slowly-changing is intentionally imprecise; it can mean that each matrix is a small Frobenius norm perturbation of its predecessor, e.g., arising from during a Newton iteration or from the modeling of crack propagation [17]. It can also refer to a sequence of systems whose spectral structure has some relationship (though their norm distance from one another is nontrivial), as may be the case when evaluating multiple parameter realizations of some underlying PDE model, e.g., in stochastic PDE applications [1]. This method allows one to retain important approximate invariant subspace information generated during the solution of the ith linear system, and leverage that information to accelerate convergence of the iteration to solve the subsequent $(i + 1)st$ system. The GCRO-DR method is one of many subspace augmentation-based recycling approaches. A general framework can be found in [18], which greatly simplifies the presentation of such approaches and enables more straightforward development of new techniques.

As high-performance computing architectures continue to evolve, the cost of floating point computations has decreased dramatically when compared to the cost of data movement; this effects both algorithm performance and power consumption costs. See, e.g., [19], for further discussion. Metrics such as the amount of data moved and the efficiency of cache reuse have become more important measures of algorithm performance than simply counting floating point operations; see, e.g., [20, 21]. *Arithmetic intensity* (i.e., the amount of computation done per unit of memory accessed) is an effective quantification of how well a particular algorithm can perform in the HPC context.

For example, in the dense linear algebra setting, it was shown that level-3 BLAS matrix-matrix operations (such as multiplying a dense matrix times a block of vectors) demonstrate superior performance over level-2 BLAS matrix-vector operations when measured in terms of arithmetic intensity. In the sparse linear algebra setting, the dominant operation of most Krylov subspace methods for the case $p = 1$ is a sparse matrix-vector multiplication, and in block Krylov subspace methods this is replaced with a sparse matrix-matrix multiplication (which in this context means a sparse matrix multiplied times a block of vectors, which is often dense). It was shown that this sparse block operation also demonstrates superior performance to its non-block counterparts in data-related metrics [20, 22]. It is known from theory and observation that block Krylov methods converge in fewer iterations than their single-vector counterparts (and at a minimum can do no worse). The superiority of performance when applying a sparse matrix to a block of p vectors when compared to p single-vector matrix-vector products (in terms of time and data-movement measurements) is clear, especially on modern and emerging architectures; cf. Section 6.1.

Given this observation, it is reasonable to consider a block version of GMRES with recycling to leverage both the algorithmic and hardware advantages, and to explore their application to block systems and systems with a single right-hand side. The extension of GCRO-DR and other such methods to the block setting is a natural one to make. This paper describes the development of the block GCRO-DR high-performance implementation [14] in the BELOS package of the TRILINOS project [23]. Based upon this code, the authors of [24] have extended this method to the flexible preconditioning setting and treat the issue of inexact block Krylov subspace breakdown thoroughly. Other such methods have also been extended to the block setting; see, e.g., [25, 26]. Based on the original implementation in TRILINOS [14], other authors have refined the block GCRO-DR algorithm to make the HPC implementation more effective; see, e.g., [27, 28].

In this paper we:

- derive the block version of GCRO-DR, as implemented in [14],
- discuss implementation decisions to favor block operations, including increasing the Krylov subspace block size using random vectors,
- demonstrate performance gains of a block Krylov subspace recycling over its single-vector counterpart,
- demonstrate the superiority of block, sparse matrix operations in terms appropriate data metrics relevant in the high-performance computing context,
- and study the efficiency of block operations specific to the block GCRO-DR setting.

It should be noted that the last two goals are achieved through direct measurement of data movement and cache use efficiency on the processor. Through carefully designed experiments, we are able to show that the cost of applying an operator to a block of vectors is often marginally greater than applying the operator to a single vector in terms of data movement and usage cost metrics. Thus, we show that block methods can offer an accelerated convergence rate while reducing the overall data transmission costs by avoiding data movement bottlenecks in modern hardware architectures.

3 Preliminaries

When not necessary for the explanation or derivation of methods, we drop the index i and consider the linear system $\boldsymbol{A}\left(\boldsymbol{X}_0 + \boldsymbol{T}\right) = \boldsymbol{B}$. Krylov subspace methods begin with a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ and a vector $\boldsymbol{u} \in \mathbb{C}^n$ and build a basis for the Krylov subspace

$$\mathcal{K}_m(\boldsymbol{A},\boldsymbol{u}) = \operatorname{span}\left\{\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}, \ldots, \boldsymbol{A}^{j-1}\boldsymbol{u}\right\}. \tag{2}$$

We focus on methods that build an orthonormal basis for the Krylov subspace using with the Arnoldi process. Let $\boldsymbol{V}_m \in \mathbb{C}^{n\times m}$ be the matrix with orthonormal columns generated by the Arnoldi process spanning $\mathcal{K}_m(\boldsymbol{A},\boldsymbol{u})$. Then we have the Arnoldi relation

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}_m} \tag{3}$$

with $\underline{\boldsymbol{H}_m} \in \mathbb{C}^{(m+1)\times m}$ upper Hessenberg; see, e.g., [29, Section 6.3] and [30].

In general, for any $p \geq 1$, let $\boldsymbol{R}_0 = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_0$ denote the initial residual. For the case $p = 1$, at iteration j, we compute $\boldsymbol{X}_m = \boldsymbol{X}_0 + \boldsymbol{T}_m$, where $\boldsymbol{T}_m \in \mathcal{K}_m(\boldsymbol{A},\boldsymbol{R}_0)$. In Gmres [31], we choose

$$\boldsymbol{T}_m = \underset{\boldsymbol{T}\in\mathcal{K}_m(\boldsymbol{A},\boldsymbol{R}_0)}{\arg\min} \left\|\boldsymbol{B} - \boldsymbol{A}(\boldsymbol{X}_0 + \boldsymbol{T})\right\|_2,$$

and this is equivalent to solving the smaller minimization problem

$$\boldsymbol{Y}_m = \underset{\boldsymbol{Y}\in\mathbb{C}^j}{\arg\min} \left\|\underline{\boldsymbol{H}_m}\boldsymbol{Y} - \|\boldsymbol{R}_0\|\,\boldsymbol{e}_1^{(m+1)}\right\|_2, \tag{4}$$

where we use the notation $\boldsymbol{e}_\ell^{(k)}$ to denote the ℓth Cartesian basis vector in $\mathbb{R}^k$, and setting $\boldsymbol{X}_m = \boldsymbol{X}_0 + \boldsymbol{V}_m\boldsymbol{y}_m$. We call $\boldsymbol{T}_m$ a *correction*. In restarted Gmres, i.e., (Gmres (m)), we halt this process at step m, discard the matrix $\boldsymbol{V}_m$, and restart with the new initial residual $\boldsymbol{R}_0 \leftarrow \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_m$. This process is repeated until we achieve convergence.

3.1 Block Krylov subspace methods

The extension of Krylov subspaces and the associated iterative methods to the block Krylov setting has been previously described in, e.g., [7, 8, 16]. Though originally described for solving (1) in the case $p > 1$, such methods have also been proposed for accelerating convergence in the case that $p = 1$. A block Krylov subspace $\mathbb{K}_m(\boldsymbol{A},\boldsymbol{R}_0)$ is a generalization of the definition of a Krylov subspace with more than one starting vector, i.e.,

$$\mathbb{K}_m(\boldsymbol{A},\boldsymbol{R}_0) = \operatorname{blspan}\left\{\boldsymbol{R}_0, \boldsymbol{A}\boldsymbol{R}_0, \boldsymbol{A}^2\boldsymbol{R}_0, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{R}_0\right\},$$

where we note that by *blspan*, we mean that we treat $n \times p$ vectors as a one-sided vector space with linear combinations being constructed using right-multiplication by $p\times p$ matrices. This means that elements of $\mathbb{K}_m(\boldsymbol{A},\boldsymbol{R}_0)$ have the form $\sum_{i=0}^{m-1} \boldsymbol{A}^i\boldsymbol{R}_0\boldsymbol{S}_i$ where $\boldsymbol{S}_i \in \mathbb{C}^{p\times p}$. This one-sided vector space approach is useful for understanding the behavior of these methods since it allows one to maintain the block structure in the analysis by, e.g., considering the iteration in a space over the *-algebra of $p \times p$ matrices. This has been used to great effect in, e.g., [32] which builds on ideas from [33].

It is straightforward to show that this interpretation is equivalent to treating this space as a vector space over $\mathbb{C}$ by observing that

$$\begin{aligned}\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0) =& \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0^{(1)}) + \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0^{(2)}) + \cdots + \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0^{(p)}), \\ =& \textsc{ColSpan}\left\{\boldsymbol{R}_0, \boldsymbol{A}\boldsymbol{R}_0, \boldsymbol{A}^2\boldsymbol{R}_0, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{R}_0\right\}, \end{aligned} \tag{5}$$

where $\boldsymbol{r}_0^{(i)} = \boldsymbol{R}_0\boldsymbol{e}_i$ is the ith column of $\boldsymbol{R}_0$. We mean this in the sense that for any vector $\boldsymbol{c} \in \mathbb{C}^p$, we can express $\left(\sum_{i=0}^{m-1} \boldsymbol{A}^i\boldsymbol{R}_0\boldsymbol{S}_i\right)\boldsymbol{c}$ as a linear combination of elements from the constituent Krylov subspaces in the sum (5).

We denote by L the block size used to generate the block Krylov subspace. We consider two block Krylov subspace use-cases:

- if $p > 1$, and we set $L = p$, we build a block Krylov subspace using $\boldsymbol{R}_0$;
- if $p = 1$ and $L > p$, we build the block Krylov subspace using $\boldsymbol{R}_0 \in \mathbb{C}^n$ and $L - 1$ other vectors, independent from the residual.

Following the description in [29, Section 6.12], we represent $\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0)$ in terms of the block Arnoldi basis $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_m\}$ where $\boldsymbol{V}_i \in \mathbb{C}^{n\times p}$ has orthonormal columns and each column of $\boldsymbol{V}_i$ is orthogonal to all columns of $\boldsymbol{V}_m$ for all $j \neq i$. We obtain $\boldsymbol{V}_1$ via the reduced QR-factorization $\boldsymbol{R}_0 = \boldsymbol{V}_1\boldsymbol{F}_0$ where $\boldsymbol{F}_0 \in \mathbb{C}^{p\times p}$ is upper triangular. We can generate $\boldsymbol{V}_{m+1}$ with the block Arnoldi step, shown in Algorithm 3.1. Let $\boldsymbol{W}_m = \begin{bmatrix}\boldsymbol{V}_1 & \boldsymbol{V}_2 \cdots \boldsymbol{V}_m\end{bmatrix} \in \mathbb{C}^{n\times mp}$. Let $\underline{\boldsymbol{H}_m} = (\boldsymbol{H}_{ij}) \in \mathbb{C}^{(m+1)p\times mp}$. This yields the block Arnoldi relation

$$\boldsymbol{A}\boldsymbol{W}_m = \boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m}, \tag{6}$$

where $\underline{\boldsymbol{H}_m} \in \mathbb{C}^{(m+1)p\times mp}$ is block upper Hessenberg, with $p \times p$ blocks, in which the lower block-subdiagonal is composed of upper-triangular matrices. A straightforward generalization of Gmres for block Krylov subspaces (called block Gmres), first described in [8]; see, e.g., [29, Chapter 6] for more details. For $p > 1$, one solves the

Algorithm 3.1: A step of the block Arnoldi algorithm

```
  Input  : $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_m \in \mathbb{C}^{n\times p}$
  Output: The block Arnoldi vector $\boldsymbol{V}_{m+1}$
1 $\widehat{\boldsymbol{V}}_{m+1} = \boldsymbol{A}\boldsymbol{V}_m$
  for $i = 1$ to $m$ do
2     $\boldsymbol{H}_{i,m} = \boldsymbol{V}_i^*\widehat{\boldsymbol{V}}_{m+1} \in \mathbb{C}^{p\times p}$
      $\widehat{\boldsymbol{V}}_{m+1} \leftarrow \widehat{\boldsymbol{V}}_{m+1} - \boldsymbol{V}_i\boldsymbol{H}_{i,m}$
3 Define $\boldsymbol{V}_{m+1}$ and $\boldsymbol{H}_{m+1,m}$ using reduced QR-factorization
    $\widehat{\boldsymbol{V}}_{m+1} = \boldsymbol{V}_{m+1}\boldsymbol{H}_{m+1,m}$
```

generalization of the single-vector GMRES minimization problem,

$$\boldsymbol{Y}_m = \underset{\boldsymbol{Y}\in\mathbb{C}^{mp\times p}}{\arg\min} \left\| \underline{\boldsymbol{H}_m}\boldsymbol{Y} - \boldsymbol{E}_1^{(m+1)p}\boldsymbol{F}_0 \right\|_F \tag{7}$$

and setting $\boldsymbol{X}_m = \boldsymbol{X}_0 + \boldsymbol{W}_m\boldsymbol{Y}_m$ where $\boldsymbol{E}_1^{(m+1)p\times p}$ is the matrix containing the first p columns of the order $(m+1)p$ identity. It is easy to show this is equivalent to computing the minimum residual 2-norm correction over the block Krylov subspace, one column at-a-time.

In the case that $p = 1$ with the block size having been enlarged to L, the subspace is $\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{V}_1)$ where $\boldsymbol{R}_0 = \|\boldsymbol{R}_0\|\,\boldsymbol{V}_1\boldsymbol{e}_1^{(L)}$. At iteration m, one solves

$$\boldsymbol{Y}_m = \underset{\boldsymbol{Y}\in\mathbb{C}^{mL}}{\arg\min} \left\| \underline{\boldsymbol{H}_m}\boldsymbol{Y} - \boldsymbol{E}_1^{(m+1)L}\boldsymbol{F}_0\boldsymbol{e}_1^{(L)} \right\|_2 \tag{8}$$

where $\boldsymbol{e}_1^{(L)} \in \{0,1\}^L$ is the first column of the identity. One then sets $\boldsymbol{X}_m = \boldsymbol{X}_0 + \boldsymbol{W}_m\boldsymbol{Y}_m$ as before. In this case, one minimizes the single-vector residual over the block Krylov subspace.

The core message is that block GMRES is a residual minimization method generalizing GMRES to the block Krylov subspace setting. This means that by understanding recycled GMRES as being from a class of methods that minimize over the sum of two subspaces (one of which is a Krylov subspace), we are able to chart a clear path forward for extending GCRO-based recycled GMRES to the block Krylov subspace setting.

3.2 Recycled GMRES

Subspace recycling is a type of augmented Krylov subspace method wherein one augments a Krylov subspace with vectors generated by a previous iteration generated by a previous iteration for the same or a previous system,[1] a technique first denoted as *recycling* in [5]. By augmented Krylov subspace, we mean that we compute a correction to the initial approximation not just over a Krylov subspace $\mathcal{K}$ but instead over an *augmented* Krylov subspace of the form $\mathcal{U} + \mathcal{K}$ where $\mathcal{U}$ is available before the start of the iteration. We use the expression recycled GMRES (RGMRES) to encompass all augmented Krylov subspace methods that minimize the residual norm over an augmented Krylov subspace. The most successful implementations are those presented in the GCRO framework introduced in [34]. It is shown that such methods equivalently can be expressed as a GMRES iteration applied to the original linear system, left-multiplied with a specially chosen projector. The survey [18] goes into much more detail and generalizes the idea to augmented/recycling approaches not based on the minimization of an error functional. The correction resulting from this iteration is then further modified to get a minimum residual approximation for the original system, cf. Section 3.2. What differentiates the methods within this class is how the augmenting subspace $\mathcal{U}$ is computed and updated.

We briefly review the method described in [5]. For simplicity, we continue to drop the superscript $\cdot^{[i]}$. We assume there is an augmentation space $\mathcal{U}$ that is available before

[1] or from some other helpful source

the start of the iteration. This algorithm represents the combination of two approaches: those originating from the implicitly restarted Arnoldi method [35], such as Morgan's GMRES-DR [36], and those descending from de Sturler's GCRO method [34]. GMRES-DR is a restarted GMRES-type algorithm, where at the end of each cycle, harmonic Ritz vectors are computed, and a subset of them are used to augment the Krylov subspace generated at the next cycle. The GCRO method allows the user to select the optimal correction over arbitrary subspaces. This concept is extended by de Sturler in [37] (and simplified in [38]), where a framework is provided for selecting the optimal subspace to retain from one cycle to the next so as to minimize the error produced by discarding useful information accumulated in the subspace for candidate solutions before restart. This algorithm is called GCRO-T, and this procedure is referred to as "optimal truncation". Parks et al. in [5] combine the ideas of [36] and [37] and extend them to a sequence of slowly-changing linear systems and recycling with harmonic Ritz vectors. They call their method GCRO-DR.

All methods that minimize the residual over an augmented subspace of the form $\mathcal{U} + \mathfrak{K}_m$ (i.e., a fixed space $\mathcal{U}$ and an iteratively generated space $\mathfrak{K}_m$ that increases in dimension at each iteration, with $d_m := \dim \mathfrak{K}_m$) have common structural characteristics that are exploited when designing an RGMRES algorithm. GCRO-based approaches such as GCRO-DR are demonstrations of this point. We distill the most important aspects of the general theory presented in the survey [18] in the residual minimization setting.

Consider solving $\boldsymbol{A}(\boldsymbol{X}_0 + \boldsymbol{S} + \boldsymbol{T}) = \boldsymbol{B}$ with $\boldsymbol{R}_0 = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_0$ for $p \geq 1$. In [18] the authors express the augmented iterative method as constructing the approximation $\boldsymbol{S} + \boldsymbol{T} \approx \boldsymbol{S}_m + \boldsymbol{T}_m \in \mathcal{U} + \mathfrak{K}_m$. The augmented residual minimization approach selects $\boldsymbol{S}_m \in \mathcal{U}$ and $\boldsymbol{T}_m \in \mathfrak{K}_m$ such that $\|\boldsymbol{B} - \boldsymbol{A}(\boldsymbol{X}_0 + \boldsymbol{S}_m + \boldsymbol{T}_m)\|$ is minimized. Let us express $\boldsymbol{S}_m = \boldsymbol{U}\boldsymbol{Z}_m$ where $\boldsymbol{U} \in \mathbb{C}^{n\times k}$ has columns spanning $\mathcal{U}$ and $\boldsymbol{Z}_m \in \mathbb{C}^{k\times p}$, and $\boldsymbol{T}_m = \boldsymbol{W}_m\boldsymbol{Y}_m$ where $\boldsymbol{W}_m \in \mathbb{C}^{n\times d_m}$ has columns spanning $\mathfrak{K}_m$ and $\boldsymbol{Y}_m \in \mathbb{C}^{d_m\times p}$. A key observation from [18, Section 5.1.1] is that solving

$$\boldsymbol{A}\begin{bmatrix}\boldsymbol{U} & \boldsymbol{W}_m\end{bmatrix}\begin{bmatrix}\boldsymbol{Z}_m \\ \boldsymbol{Y}_m\end{bmatrix} \approx \boldsymbol{R}_0$$

via a least-squares approach using the normal equations is equivalent to approximating the solution to the singular, consistent linear system

$$(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}\boldsymbol{T} = (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{R}_0 \tag{9}$$

with $\boldsymbol{T}_m \in \mathfrak{K}_m$ via residual minimization and then constructing

$$\boldsymbol{X}_m = \boldsymbol{X}_0 - \boldsymbol{\Pi}\boldsymbol{T} + (\boldsymbol{I} - \boldsymbol{\Pi})\,\boldsymbol{T}_m, \tag{10}$$

where $\boldsymbol{\Phi}$ is the orthogonal projector onto $\mathcal{C} := \boldsymbol{A}\mathcal{U}$, $\boldsymbol{\Pi}$ is the $\boldsymbol{A}^*\boldsymbol{A}$-orthogonal projector onto $\mathcal{U}$, and the resulting initial error projection $\boldsymbol{\Pi}\boldsymbol{T}$ is a computable quantity. Furthermore, it was shown that for any such augmented subspace residual minimization, the full residual and the residual for the projected subproblem are the same, i.e.,

$$\boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_m = (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{R}_0 - (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}\boldsymbol{T}_m. \tag{11}$$

Remark 3.1. *We note that the framework developed in [18] is independent of the choices of subspaces $\mathcal{U}$ and $\mathfrak{K}_m$. The projected subproblem (9) comes from doing a Galerkin residual minimization over a sum of two subspaces.*

Indeed, it is the choice of $\mathfrak{K}_m$ that takes a general augmented subspace residual minimization and turns it into a GCRO-based RGMRES algorithm. If we specify $\mathfrak{K}_m = \mathcal{K}_m\left((\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}, (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{R}_0\right)$, then this process becomes equivalent to applying a GMRES iteration directly to (9). It follows then from (11) that the residual convergence behavior of a GCRO-based RGMRES method is governed completely by the behavior of GMRES applied to the projected subproblem (9); cf. Section 5 for further discussion of this fact.

3.2.1 Standard GCRO-DR version of RGMRES

Suppose we are solving (1) with $p = 1$, and we have a k-dimensional subspace $\mathcal{U}$ which is spanned by vectors recycled either from a previous linear system solve or in the previous iteration cycle and whose image under the action of $\boldsymbol{A}$ is $\mathcal{C} = \boldsymbol{A}\mathcal{U}$. Let $\boldsymbol{\Phi}$ be the orthogonal projector onto $\mathcal{C}$. As discussed in Section 3.2, minimizing the residual over an augmented Krylov subspace $\mathcal{U} + \mathfrak{K}_m$ is equivalent to applying a GMRES to (9) if we choose $\mathfrak{K}_m = \mathcal{K}_m\left((\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}, (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{R}_0\right)$. We generate the Krylov subspace using a projected version of the Arnoldi process. After m iterations, GMRES applied to (9) produces the correction $\boldsymbol{T}_m \in \mathcal{K}_m\left((\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}, (\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{R}_0\right)$. At the end of the cycle, an updated $\mathcal{U}$ is constructed, the Krylov subspace basis is discarded, and we restart. At convergence, $\mathcal{U}$ is saved, to be used when solving the next linear system.

Practical construction of (10) is straightforward. A fundamental choice for implementing any recycling method is the choice of bases for $\mathcal{U}$ and $\mathcal{C}$. For GCRO-based RGMRES implementations, the usual choice is to maintain an orthonormal basis for $\mathcal{C}$. This greatly simplifies the representation of the projectors $\boldsymbol{\Pi}$ and $\boldsymbol{\Phi}$ since $\boldsymbol{C}^*\boldsymbol{C} = \boldsymbol{I}$. It follows that

$$\boldsymbol{\Pi} = \boldsymbol{U}\left(\boldsymbol{U}^*\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{U}\right)^{-1}\boldsymbol{U}^*\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{U}\boldsymbol{U}^*\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{U}\boldsymbol{C}^*\boldsymbol{A}, \qquad \text{and} \qquad \boldsymbol{\Phi} = \boldsymbol{C}\boldsymbol{C}^*.$$

It is well documented that all recycling methods that fit into the framework described in [18] have a projected error term of the form $\boldsymbol{\Pi}\boldsymbol{T}$ that is practically computable. In the RGMRES setting, this follows from the structure of $\boldsymbol{\Pi}$ since $\boldsymbol{\Pi}\boldsymbol{T} = \boldsymbol{U}\boldsymbol{C}^*\boldsymbol{R}_0$. When applying GMRES to (9), we obtain the modified (projected) Arnoldi relation

$$(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}_m} \tag{12}$$
$$\iff \boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{V}_m + \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}_m} = \boldsymbol{C}\boldsymbol{B}_m + \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}_m}, \tag{13}$$

where $\boldsymbol{B}_m = \boldsymbol{C}^*\boldsymbol{A}\boldsymbol{V}_m$. In terms of implementation, the action of $(\boldsymbol{I} - \boldsymbol{\Phi})$ is implemented as an orthogonalization away from the orthonormal columns of $\boldsymbol{C}$, with the coefficients being stored in $\boldsymbol{B}_m$. Solving the usual GMRES minimization, we obtain

$\boldsymbol{Y}_m = \arg\min_{\boldsymbol{Y}\in\mathbb{C}^m} \left\| \underline{\boldsymbol{H}_m}\boldsymbol{Y} - \beta \boldsymbol{e}_1 \right\|$ where $\beta = \|(\boldsymbol{I} - \boldsymbol{\Phi})\, \boldsymbol{R}_0\|$, and $\boldsymbol{T}_m = \boldsymbol{V}_m\boldsymbol{Y}_m$. Lastly, we obtain $\boldsymbol{\Pi T}_m$ by observing that

$$\boldsymbol{\Pi T}_m = \boldsymbol{U C}^* \boldsymbol{A V}_m \boldsymbol{Y}_m = \boldsymbol{U B}_m \boldsymbol{Y}_m,$$

which involves already-computed quantities.

Convergence analysis for augmented Krylov subspace methods was previously presented in, e.g., [39, 40]. In the context of rGmres, the thesis of Gaul [41] and the references therein are all excellent sources on this topic. Further analysis of these methods and of the deflated operator $(\boldsymbol{I} - \boldsymbol{\Phi})\, \boldsymbol{A}$ is presented.

Iterating orthogonally to an approximate invariant subspace to accelerate convergence of Gmres can be justified by the theoretical work in [42], wherein it is shown that the widely observed two-stage convergence behavior of Gmres, which has been termed *superlinear convergence*, is governed by how well the Krylov subspace approximates a certain eigenspace. Specifically, when the Krylov subspace contains a good approximation to the eigenspace (call this eigenspace $\mathcal{S}$) associated with eigenvalues hindering convergence, we transition from the slow phase to the fast phase, and convergence will mimic that of Gmres on the projected operator $\boldsymbol{P}_{\mathcal{S}^\perp}\boldsymbol{A}$ where $\boldsymbol{P}_{\mathcal{S}^\perp}$ is the orthogonal projector onto $\mathcal{S}^\perp$. This analysis complements previous discussions of this phenomenon, see e.g., [43, 44].

However, it should be noted the theory describing the effectiveness of rGmres applied to a non-normal system is not fully understood. Explanations that characterize acceleration of convergence in terms of projections onto invariant subspaces do not take into consideration the non-normality of the matrix; and, thus, the effects of ill-conditioning of the eigenbasis are ignored. This has been alluded to [45] but not yet fully explored.

4 Recycled Block Gmres

The beauty of the framework discussed in Section 3.2, is that it is compatible with any minimum residual iterative method over an augmented Krylov subspace. We simply let $\mathfrak{K}_m$ be a block Krylov subspace. We describe the method to accommodate $p \geq 1$ and describe the differences in approach when working with a true block method ($p > 1$) versus when $p = 1$. In that vein, we consider generic block size L.

Given a subspace $\mathcal{U}$ we derive the block recycled Gmres iteration thusly. Using the block Arnoldi process, we generate a basis for the subspace $\mathfrak{K}_m = \mathbb{K}_m\left((\boldsymbol{I} - \boldsymbol{\Phi})\, \boldsymbol{A}, (\boldsymbol{I} - \boldsymbol{\Phi})\, \boldsymbol{R}_0\right)$ where the orthonormal columns of $\boldsymbol{W}_m \in \mathbb{C}^{n\times mL}$ span the subspace. By construction, the columns of $\boldsymbol{W}_{m+1}$ are orthogonal to the columns of $\boldsymbol{C}$, yielding a block version of (13),

$$\boldsymbol{A W}_m = \boldsymbol{C B}_m + \boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m} \tag{14}$$

where $\boldsymbol{B}_m = \boldsymbol{C}^*\boldsymbol{A W}_m \in \mathbb{C}^{k\times mL}$ represents the entries generated by orthogonalizing the columns of the new block Krylov basis vector against $\mathcal{C}$. The derivation proceeds just as in Section 3.2. It still holds that $\boldsymbol{\Pi T} = \boldsymbol{U C}^* \boldsymbol{R}_0$. We obtain $\boldsymbol{T}_m$ as the mth block GMRES approximation to the solution of (9), and $\boldsymbol{\Pi T}_m = \boldsymbol{C B}_m \boldsymbol{Y}_m$. We obtain $\boldsymbol{Y}_m$ from the minimization (7) in the case that $p > 1$ and (8) in the case that $p = 1$.

Proposition 4.1. *The block residual produced by block GCRO-DR applied to* (1) *and that produced by block GMRES applied to* (9) *are equal; i.e.,*

$$\boldsymbol{B} - \boldsymbol{A}\left(\boldsymbol{X}_0 + \boldsymbol{S}_m + \boldsymbol{T}_m\right) = \left(\boldsymbol{I} - \boldsymbol{\Phi}\right)\left(\boldsymbol{R}_0 - \boldsymbol{A T}_m\right).$$

Proof. At iteration m of block GCRO-DR, we have the block residual

$$\boldsymbol{R}_m = \boldsymbol{B} - \boldsymbol{A}\left(\boldsymbol{X}_0 + \boldsymbol{S}_m + \boldsymbol{T}_m\right).$$

Inserting the expressions from earlier for the two corrections, we can write

$$\begin{aligned}
\boldsymbol{R}_m =& \boldsymbol{R}_0 - \boldsymbol{A}\left(-\boldsymbol{U B}_m \boldsymbol{Y}_m + \boldsymbol{W}_m \boldsymbol{Y}_m\right) \\
=& \boldsymbol{R}_0 + \boldsymbol{C B}_m \boldsymbol{Y}_m - \begin{bmatrix}\boldsymbol{C} & \boldsymbol{W}_{m+1}\end{bmatrix} \begin{bmatrix}\boldsymbol{B}_m \\ \underline{\boldsymbol{H}_m}\end{bmatrix} \boldsymbol{Y}_m \\
=& \boldsymbol{R}_0 + \boldsymbol{C B}_m \boldsymbol{Y}_m - \boldsymbol{C B}_m \boldsymbol{Y}_m - \boldsymbol{W}_{m+1} \underline{\boldsymbol{H}_m} \boldsymbol{Y}_m \\
=& \boldsymbol{R}_0 - \boldsymbol{W}_{m+1} \underline{\boldsymbol{H}_m} \boldsymbol{Y}_m.
\end{aligned}$$

This is the block residual produced by block GMRES applied to the projected problem (9), proving the proposition. □

One can, in fact, represent (14) as one large blocked Hessenberg relation. Let

$$\widehat{\boldsymbol{W}}_m = \begin{bmatrix}\boldsymbol{U} & \boldsymbol{W}_m\end{bmatrix}, \qquad \widetilde{\boldsymbol{W}}_{m+1} = \begin{bmatrix}\boldsymbol{C} & \boldsymbol{W}_{m+1}\end{bmatrix}, \quad \text{and} \quad \underline{\boldsymbol{G}_m} = \begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{B}_m \\ \underline{0} & \underline{\boldsymbol{H}_m}\end{bmatrix}, \tag{15}$$

with $\boldsymbol{I}_k$ being the $k \times k$ identity matrix. It follows that we can write

$$\boldsymbol{A}\widehat{\boldsymbol{W}}_m = \widetilde{\boldsymbol{W}}_{m+1} \underline{\boldsymbol{G}_m}. \tag{16}$$

We do not advocate implementing a GCRO-DR method using a compact augmented Arnoldi relation, as it complicates the algorithm and introduces possible stability issues; see, e.g., [5]. However, it is useful to introduce it for the computation of harmonic Ritz vectors, cf., Section 4.1.2.

We use $\underline{\boldsymbol{G}_m}$ to compute a new approximate invariant subspace; and if we have not converged, we begin the next cycle. Algorithm 4.1 gives a complete pseudocode description of the algorithm.

4.1 Implementation considerations

We discuss implementation decisions made in light of the fact that the proposed method is built upon a block Krylov subspace method.

Algorithm 4.1: The Block rGmres Algorithm

```
Input  : $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, $\boldsymbol{B} \in \mathbb{C}^{n\times p}$, $\boldsymbol{X}_0 \in \mathbb{C}^{n\times p}$, $\varepsilon > 0$ the convergence tolerance,
         $m$ the number of block Arnoldi vectors generated, $k$ the desired
         dimension of the space to be recycled, and possibly $\boldsymbol{U}$, whose $k$
         columns span the space to be recycled (optional, if available).
Output: $\boldsymbol{X} \in \mathbb{C}^{n\times p}$ such that $\|\boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}\|_F \le \varepsilon$
1 Set $\boldsymbol{R}_0 = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_0$, Set $i = 0$
   Set $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{e}_1^{(mp)} & \cdots & \boldsymbol{e}_p^{(mp)} \end{bmatrix}$
   if $\boldsymbol{U}$ is available (e.g., defined from solving a previous linear system) then
2     Define $\boldsymbol{C}$ using reduced QR Factorization $\boldsymbol{A}\boldsymbol{U} = \boldsymbol{C}\boldsymbol{S}$, $\boldsymbol{U} \leftarrow \boldsymbol{U}\boldsymbol{S}^{-1}$, and
       $\boldsymbol{\Phi} = \boldsymbol{C}\boldsymbol{C}^*$
       $\boldsymbol{X}_0 \leftarrow \boldsymbol{X}_0 + \boldsymbol{U}\boldsymbol{C}^*\boldsymbol{R}_0$, $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_0 - \boldsymbol{\Phi}\boldsymbol{R}_0$
3 else
4     Define $\boldsymbol{V}_1$ using reduced QR Factorization $\boldsymbol{R}_0 = \boldsymbol{V}_1\boldsymbol{Z}$
       Perform $m$ steps of block Gmres, generating $\boldsymbol{W}_{m+1}$ and $\underline{\boldsymbol{H}_m}$ and solve
       $\boldsymbol{Y}_0 = \arg\min_{\boldsymbol{Y}\in\mathbb{C}^{mp\times p}} \left\|\underline{\boldsymbol{H}_m}\boldsymbol{Y} - \boldsymbol{E}\boldsymbol{Z}\right\|_F$,
       $\boldsymbol{X}_0 \leftarrow \boldsymbol{X}_0 + \boldsymbol{W}_m\boldsymbol{Y}_0$, $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_0 - \boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m}\boldsymbol{Y}_0$
       Select a subspace $\mathcal{U}$ of $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{V}_1)$ to recycle.
       Compute $\boldsymbol{U}$ having basis vectors of $\mathcal{U}$ as columns such that $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{U}$ has
       orthonormal columns, and $\boldsymbol{\Phi} = \boldsymbol{C}\boldsymbol{C}^*$
5 while $\|\boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_i\|_F > \varepsilon$ do
6     $i \leftarrow i + 1$
       Define $\boldsymbol{V}_1$ using reduced QR Factorization $\boldsymbol{R}_{i-1} = \boldsymbol{V}_1\boldsymbol{Z}$
       Compute a basis for $\mathbb{K}_m((\boldsymbol{I} - \boldsymbol{\Phi})\boldsymbol{A}, \boldsymbol{V}_1)$ using the block Arnoldi method,
       generating $\boldsymbol{W}_{m+1}$, $\underline{\boldsymbol{H}_m}$, and $\boldsymbol{F}_m$.
       Solve the block Gmres least-squares subproblem
       $\boldsymbol{Y}_i = \arg\min_{\boldsymbol{Y}\in\mathbb{C}^{mp\times p}} \left\|\underline{\boldsymbol{H}_m}\boldsymbol{Y} - \boldsymbol{E}\boldsymbol{Z}\right\|_F$
       Set $\boldsymbol{X}_i = \boldsymbol{X}_{i-1} + \boldsymbol{W}_m\boldsymbol{Y}_i - \boldsymbol{U}\boldsymbol{F}_m\boldsymbol{Y}_i$, Set $\boldsymbol{R}_i = \boldsymbol{R}_{i-1} - \boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m}\boldsymbol{Y}_i$
       Define $\boldsymbol{D}$ to be the diagonal matrix such that $\widetilde{\boldsymbol{U}} = \boldsymbol{U}\boldsymbol{D}$ has columns of
       unit norm.
       Set $\underline{\boldsymbol{G}_m} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{F}_m \\ \underline{0} & \underline{\boldsymbol{H}_m} \end{bmatrix}$;            // Scaling $\boldsymbol{U}$ with $\boldsymbol{D}$ for stability
7     Compute $\mathcal{U}_{new} \subset \mathcal{U} + \mathbb{K}_m((\boldsymbol{I} - \boldsymbol{\Phi})\boldsymbol{A}, \boldsymbol{V}_1)$, $\mathcal{U} \leftarrow \mathcal{U}_{new}$
       Compute $\boldsymbol{U}$ such that $\boldsymbol{C} \leftarrow \boldsymbol{A}\boldsymbol{U}$ has orthonormal columns, and $\boldsymbol{\Phi} = \boldsymbol{C}\boldsymbol{C}^*$
8 Store $\boldsymbol{U}$ in memory to serve as initial recycle subspace for next function call.
```

4.1.1 Householder reflection storage

Working with a block Hessenberg matrix introduces some additional computational challenges as compared to the non-block case. We elaborate on our approach to block triangularization of the block Hessenberg matrix. In the case of $L = p = 1$ the upper Hessenberg matrix $\underline{\boldsymbol{H}_m}$ has only one subdiagonal entry per column. To compute its QR-factorization at each step of the method, one annihilates the subdiagonal entry of each column in a progressive manner using Givens rotations, which are retained compactly in the form of sines and cosines to be applied to subsequent columns. For $L > 1$, $\boldsymbol{H}_m$ becomes block upper Hessenberg. For a block of columns, newly generated by a step of the block Arnoldi procedure, new Householder reflections are computed column-by-column. However, we must first apply all previously generated reflections to this new block of columns. We employ the strategy of Gutknecht and Schmelzer [46]. One stores the Householder reflections for a block column as a single matrix and applies them all at once. This exchange p applications of previous Householder reflections for one dense matrix-matrix multiplication. This dense matrix-matrix multiplication can be performed as a level-3 Blas operation. It has been noted that for certain approaches to understanding the behavior of block Gmres, the block Householder transformations can be difficult to interpret. Indeed, in [32], the authors interpret the block GMRES iteration for the case $p > 1$ in terms of a block vector iteration over a one-sided vector space with scalars from the *-algebra of $\mathbb{C}^{p\times p}$, and they use this interpretation to meaningfully extend the results from [47] to the block Gmres setting. When considering a block Krylov iteration in this way, it is more natural to to formally consider[2] the triangularization of $\underline{\boldsymbol{H}_m}$ via a block generalization of Givens rotations.

For a block version of Gcro-Dr, the computation and updating of the recycled subspace $\mathcal{U}$ is a direct generalization of the non-block case. We note that if no space $\mathcal{U}$ is given at execution, we follow [5] and run a cycle of block Gmres, computing harmonic Ritz vectors with respect to the block Krylov subspace at the end of the cycle. We discuss this computation in more detail and also ponder other recycling strategies.

4.1.2 Harmonic Ritz vector computation

This is the strategy implemented in [14] following the harmonic Ritz vector deflation strategy of Morgan in [36]. At the end of the cycle, we have generated an orthonormal basis for the subspace $\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0)$ with the block Arnoldi relation $\boldsymbol{A}\boldsymbol{W}_m = \boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m}$. Following [48], the block harmonic Ritz problem for $\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0)$ is to find all pairs

$$(\boldsymbol{y}, \mu) \in \mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0) \times \mathbb{C} \text{ such that } \boldsymbol{A}^{-1}\boldsymbol{y} - \mu\boldsymbol{y} \perp \boldsymbol{A}\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0). \tag{17}$$

As with the scalar case, (17) can be equivalently solved as a generalized eigenvalue problem whose solution pairs $(\boldsymbol{t}, \tilde{\theta}) \in \mathbb{C}^{mp} \times \mathbb{C}$ can be used to reconstruct the pairs $(\boldsymbol{y}, \mu) \in \mathbb{C}^n \times \mathbb{C}$ as described in the following.

Proposition 4.2. *Given the block Krylov subspace $\mathbb{K}_m(\boldsymbol{A}, \boldsymbol{R}_0)$, solving the harmonic Ritz problem (17) is equivalent to solving the $mp \times mp$ eigenvalue problem*

$$(\boldsymbol{H}_m + (\boldsymbol{H}_m^*)^{-1}\widehat{\boldsymbol{E}}(\boldsymbol{H}_{m+1,m}^*\boldsymbol{H}_{m+1,m})\widehat{\boldsymbol{E}}^*)\boldsymbol{t} = \tilde{\theta}\boldsymbol{t}$$

and then for a solution pair $(\boldsymbol{t}, \tilde{\theta})$ assigning $\boldsymbol{y} = \boldsymbol{W}_m\boldsymbol{t}$, where the columns of $\widehat{\boldsymbol{E}} \in \{0,1\}^{mp\times p}$ are columns $(mp-p+1), \ldots, mp$ of the identity matrix of order mp, and $\mu = 1/\tilde{\theta}$.

[2]We say only formally because it is not practical to implement them.

It should be noted that, as a practical matter, the expression

$$\boldsymbol{H}_m + (\boldsymbol{H}_m^*)^{-1}\widehat{\boldsymbol{E}}(\boldsymbol{H}_{m+1,m}^*\boldsymbol{H}_{m+1,m})\widehat{\boldsymbol{E}}^*$$

simply means that the last p columns of $\boldsymbol{H}_m$ are modified by the $mp \times p$ matrix

$$(\boldsymbol{H}_m^*)^{-1}\widehat{\boldsymbol{E}}(\boldsymbol{H}_{m+1,m}^*\boldsymbol{H}_{m+1,m}).$$

Proof of Proposition 4.2. This is a generalization of the harmonic Ritz computation in the case of a single-vector Krylov subspace; see e.g., [49]. We can prove this through algebraic manipulation using the block Arnoldi relation. Condition (17) is equivalent to

$$\begin{aligned}
(\boldsymbol{A}\boldsymbol{W}_m)^*(\boldsymbol{A}^{-1}\boldsymbol{y} - \mu\boldsymbol{y}) &= 0\\
(\boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m})^*(\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{W}_m\boldsymbol{t} - \mu\boldsymbol{A}\boldsymbol{W}_m\boldsymbol{t}) &= 0\\
\underline{\boldsymbol{H}_m}^*\boldsymbol{W}_{m+1}^*(\boldsymbol{W}_m\boldsymbol{t} - \mu\boldsymbol{W}_{m+1}\underline{\boldsymbol{H}_m}\boldsymbol{t}) &= 0\\
\underline{\boldsymbol{H}_m}^*\boldsymbol{W}_{m+1}^*\boldsymbol{W}_m\boldsymbol{t} &= \mu\underline{\boldsymbol{H}_m}^*\underline{\boldsymbol{H}_m}\boldsymbol{t}\\
\boldsymbol{H}_m^*\boldsymbol{t} &= \mu(\boldsymbol{H}_m^*\boldsymbol{H}_m + \widehat{\boldsymbol{E}}(\boldsymbol{H}_{m+1,m}^*\boldsymbol{H}_{m+1,m})\widehat{\boldsymbol{E}}^*)\boldsymbol{t}\\
\tilde{\theta}\boldsymbol{t} &= (\boldsymbol{H}_m + (\boldsymbol{H}_m^*)^{-1}\widehat{\boldsymbol{E}}(\boldsymbol{H}_{m+1,m}^*\boldsymbol{H}_{m+1,m})\widehat{\boldsymbol{E}}^*)\boldsymbol{t}
\end{aligned}$$

□

The computation in the case of the augmented subspace $\mathcal{U} + \mathcal{K}_m((\boldsymbol{I}-\boldsymbol{\Phi})\,\boldsymbol{A}, (\boldsymbol{I}-\boldsymbol{\Phi})\,\boldsymbol{R}_0)$ is similar to the computation employed in [5], as described in the following.

Proposition 4.3. *In a cycle of block recycled* GMRES*, if we have generated an augmented space* $\mathcal{U} + \mathcal{K}_m((\boldsymbol{I}-\boldsymbol{\Phi})\,\boldsymbol{A}, (\boldsymbol{I}-\boldsymbol{\Phi})\,\boldsymbol{R}_0)$ *then solving the associated harmonic Ritz problem is equivalent to solving the generalized eigenvalue problem*

$$\underline{\boldsymbol{G}_m}^*\underline{\boldsymbol{G}_m}\boldsymbol{t} = \tilde{\theta}\underline{\boldsymbol{G}_m}^*\widetilde{\boldsymbol{W}}_{m+1}^*\widehat{\boldsymbol{W}}_m\boldsymbol{t} \tag{18}$$

and assigning $\boldsymbol{y} = \widehat{\boldsymbol{W}}_m\boldsymbol{t}$ *for each solution pair* $(\tilde{\theta}, \boldsymbol{t})$ *where* $\underline{\boldsymbol{G}_m}$, $\widetilde{\boldsymbol{W}}_{m+1}^*$, *and* $\widehat{\boldsymbol{W}}_m$ *are defined as in* (15).

Proof. The proof is nearly identical to the one developed in [5, Equation 2.16]. □

4.1.3 Other recycled space selection techniques

Block Krylov subspaces have been originally proposed for the computation of eigenvalues/eigenvectors with the justification that they generate richer subspaces, see, e.g., [50, 51]. Thus, recycling approximate eigenvectors, as in [5, 25, 36, 50, 52] offers the possibility of rapidly acquiring high quality eigenvector approximations with which to deflate. This makes the use of block GCRO-DR or some block/non-block hybrid strategy more attractive for the case $p = 1,\ L > 1$. For the first few cycles, one can inflate the block size in order to more quickly obtain a high quality recycled subspace $\mathcal{U}$ and then switch at some restart to non-block Gcro-Dr (i.e., $L = 1$) thereafter.

However, our motivation arises mainly from considerations in the high-performance computing setting. For dense linear algebra computations, it has been shown that level-3 Blas (i.e., matrix-times-matrix) operations exhibit superior data movement efficiency properties, as measured amount of data moved per operation and efficiency of data reuse in cache [53]. The assumption in designing this algorithm is that sparse matrix-times-matrix operations would also exhibit similar superior properties and that level-1 and level-2 Blas operations generalize to level-3 Blas. This has been previously discussed [20, 21]. Careful experimentation will be necessary to demonstrate this, not only to understand this behavior for the application of a large, sparse operator $\boldsymbol{A}$ but also for the application of the projected operator $(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}$.

In the current version of our codes [14] (as well as in the current version of the publicly available Gcro-Dr codes [54]) harmonic Ritz vectors are computed to generate a subspace to recycle.

Indeed, there are other recycling strategies discussed in the literature. Morgan suggests that in an eigenvector deflation algorithm based upon Fom, called Fom-Dr [36], deflation using Ritz vectors is more effective. It is suggested in [5] that perhaps a mix of Ritz and harmonic Ritz vectors may be appropriate in some cases. In his paper on optimal truncation methods [37], de Sturler demonstrates that one can calculate which subspace of dimension k of the current Krylov subspace of dimension m most important to maintain orthogonality against, for the purpose of reducing the residual. This subspace is then recycled under the assumption that it is most important to continue to maintain orthogonality with respect to this subspace. Ahuja et al. [55], observed that the preconditioned systems with which they dealt had eigenvalue clusters well separated from the origin, rendering the use of harmonic Ritz vectors less effective. Instead, they chose to recycle Krylov vectors which had dominant components in the right-hand side, and this gave improved convergence results.

Gaul and Schlömmer [56] suggest that in the context of recycled Minres being used to solve a Schrödinger-type equation, Ritz vectors are good candidates with which to recycle. In [57], the authors propose a method of recycling using a proper orthogonal decomposition approach coming from model order reduction. In the context of ill-posed image recovery problems, it has been demonstrated that one can also augment the Krylov subspace with vectors which encode knowledge of characteristics of the true solution, e.g., edge characteristics of the image [58]. In that work, flexible Gmres [59] is used to augment the subspace. This follows from the work in [60–62] in which Gmres for ill-posed problem is augmented with vectors encoding features of the reconstructed image which are difficult for a Krylov method to reconstruct (such as discontinuities and hard edges). Using the augmented method framework discussed in [18], the author of [63] re-interpreted the work of [62] in order to propose an alternative implementation.

5 Convergence discussion

It is shown, e.g., in [42], that the convergence of GMRES accelerates, entering a superlinear phase, once the Krylov method has adequately captured a subspace spanned by eigenvectors associated to eigenvalues which often cause slow convergence, i.e,. those near the origin. For these eigenvalues, low-degree residual polynomial interpolation can be difficult; see also, [43, 44]. This explains some of the convergence difficulties exhibited by restarted methods, in which we discard the entire basis and start over. Furthermore, once this eigenspace is well-represented by the Krylov subspace, cf. [64], the convergence behavior mimics that of an operator from which the eigenspace has been removed. This is one motivation for the subspace augmentation and recycling technique, e.g., [5, 36]. By recycling a selected subspace and iterating orthogonally to it, we hope to enter the superlinear convergence phase of GMRES more quickly. By building a block Krylov subspace, one can capture these invariant subspaces in fewer iterations.

For rGMRES, this makes sense in terms of the commonly discussed intuition regarding GMRES convergence. Eigenspaces which cause the most trouble for GMRES are those associated to eigenvalues near zero. Implicitly, GMRES constructs a residual polynomial $p(x)$ which we would like to be small near the eigenvalues of $\boldsymbol{A}$. However, this polynomial must also satisfy $p(0) = 1$. When $\boldsymbol{A}$ has eigenvalues near the origin, construction of such a polynomial becomes more difficult and is one cause of the slow convergence phase of GMRES. Harmonic Ritz values and vectors have been show to yield better approximations to eigenvalues near the origin and their associated eigenspaces. Therefore, removing the influence of this approximate eigenspace may yield better convergence. Numerical experience has shown that this strategy frequently gives good results.

However, it is now more well understood that this is not a robust explanation as to what leads to slower GMRES convergence. Indeed, it is well understood that pathologically, the eigenvalues themselves can have no connection to the residual convergence pattern of GMRES [47], a result that has been extended to the block GMRES setting [32]. More practically, examples are presented in [65] that illuminate the complicated nature of the mechanics of GMRES convergence speed.

A block rGMRES iteration is equivalent to a block GMRES iteration applied to a projected problem. Thus, the convergence results for block GMRES can be extended to the recycled block GMRES case. We focus without loss of generality on the true block method case of $p > 1$. Simoncini and Gallopoulos discussed the convergence properties of block GMRES [33], including a result by Vital [8], which follows directly from the containment of the single-vector Krylov subspace in the block Krylov subspace,

$$\max_{i=1,\ldots,p} \min_{\boldsymbol{t}\in\mathbb{K}_m(\boldsymbol{A},\boldsymbol{R}_0)} \left\| \boldsymbol{b}^{(i)} - \boldsymbol{A}\left(\boldsymbol{x}_0^{(i)} + \boldsymbol{t}\right) \right\| \leq \max_{i=1,\ldots,p} \min_{\boldsymbol{t}\in\mathcal{K}_m(\boldsymbol{A},\boldsymbol{r}_0^{(i)})} \left\| \boldsymbol{b}^{(i)} - \boldsymbol{A}(\boldsymbol{x}_0^{(i)} + \boldsymbol{t}) \right\| .$$

The same subspace containment can be used to show.that

$$\min_{\boldsymbol{t}\in\mathbb{K}_m(\boldsymbol{A},\boldsymbol{R}_0)} \left\| \boldsymbol{b}^{(i)} - \boldsymbol{A}(\boldsymbol{x}_0^{(i)} + \boldsymbol{t}) \right\| \leq \min_{\boldsymbol{t}\in\mathcal{K}_m(\boldsymbol{A},\boldsymbol{r}_0^{(i)})} \left\| \boldsymbol{b}^{(i)} - \boldsymbol{A}(\boldsymbol{x}_0^{(i)} + \boldsymbol{t}) \right\| ,$$

for all $i = 1, 2, \cdots, p$.

The subspaces underlying rGmres and block rGmres satisfy the same containment relationships. Thus we have

$$\begin{aligned}\max_{i=1,2,\ldots p} \min_{\substack{s\in\mathcal{U}\\ t\in\mathbb{K}_m\left((I-\Phi)A,\widehat{R}\right)}} &\left\|b^{(i)} - A(\widehat{x}_0^{(i)} + t + s)\right\| \\ &\leq \max_{j=1,2,\ldots,p} \min_{\substack{s\in\mathcal{U}\\ t\in\mathcal{K}_m\left((I-\Phi)A,\widehat{r}^{(i)}\right)}} \left\|b^{(i)} - A(\widehat{x}_0^{(i)} + t + s)\right\|\end{aligned}$$

Furthermore, we have that

$$\min_{\substack{s\in\mathcal{U}\\ t\in\mathbb{K}_m\left((I-\Phi)A,\widehat{R}\right)}} \left\|b^{(i)} - A(\widehat{x}_0 + t + s)\right\| \leq \min_{\substack{s\in\mathcal{U}\\ t\in\mathcal{K}_m\left((I-\Phi)A,\widehat{r}^{(i)}\right)}} \left\|b^{(i)} - A(\widehat{x}_0 + t + s)\right\|.$$

In addition, it should be noted that the polynomial approximation interpretation of Gmres has been extended to the block case, whereby it has been observed that this can be generalized to *matrix-valued polynomials* in the block case [33].

It should be noted that any per iteration gains realized by using a block method need to be weighed against the additional cost. Each iteration of a block method requires more FLOPS than the non-block variant, but this comes with the possibility of accelerated convergence. Previous researchers have demonstrated that the addition expense of moving to a block method (as measured in data movement metrics) is only marginally greater than that of its single-vector counterpart. We explore this advantage in Section 6.

6 Numerical results

We have described the original implementation of block Gcro-Dr, with versions of in Matlab [15] and a fully deployed implementation in the Belos package of Sandia's Trilinos Project [14].

One point which must be discussed is how to compare the performance of a block Gcro-Dr to algorithms that execute only a matrix–vector product per iteration. Iteration-for-iteration, block methods have a different dominant core operation in the iteration, the block p matvec. However, the block p matvec does not cost p times as much as a single standard matvec. Thus we present two sets of experiments. One set, shown in Section 6.1, are all performed in Matlab to demonstrate characteristics of algorithm performance for small-scale problems. The second set of experiments, shown in Section 6.2 are performed in Trilinos, and demonstrate performance characteristics of the core operations of block Gcro-Dr for very large, sparse matrices. After each cycle, harmonic Ritz vectors are used to build the recycled subspace.

6.1 Small-scale convergence experiments

The experiments in this section were performed on a Macbook Pro with a 3.1 GHz Dual-Core Intel Core i5 processor and 8 GB of 2133 MHz DDR3 main memory. We demonstrate timing comparisons for performing single and block p matvecs for computing the action of a sparse matrix A on equal numbers of vectors.

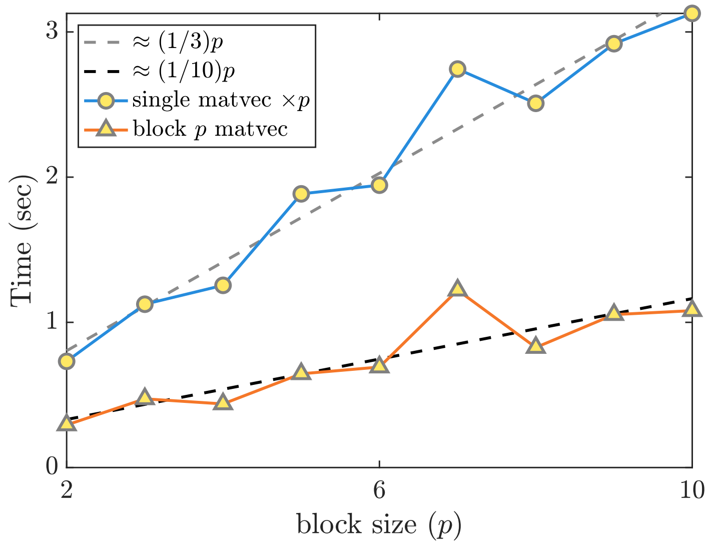

Fig. 1 Comparison of the time taken to perform p single matrix-vector products for the sparse `sherman5` of order 3312 from [66] matrix for the single matrix-vector product versus the block size p block p matvec for different values of p, the block size. The **dashed lines** show the linear least-squares fits to the data from each experiment. Each experiment was performed 10^5 times.

In Figure 1, we see that the block p matvec is able to outperform the single matrix-vector product, in MATLAB. However, we are more interested in an iteration for iteration performance comparison of a block Krylov subspace method versus a single-vector Krylov subspace method. We pose the question, do the benefits of convergence in fewer iterations outweigh the increased number of floating-point operations of the block block p matvec? In Figure 2, we compare the time taken to perform matrix-vector products with the time taken to compute block p matvecs. We see that, though block p matvecs are more expensive to compute than the single-vector variety, they are not p times as expensive.

In Figure 3, we test the code's convergence properties as we increase the number of right-hand sides. As is predicted by the underlying theory for this problem, the increased number of right-hand sides generates a richer space from which to select our approximation updates and from which to recycle, though the marginal benefit decreases for each additional right-hand side.

We extracted matrices from seven consecutive iterations of a TRAMONTO Newton iteration from the `POLY_CMS_1D` test problem [67]. For each iteration, we precondition using `ILU(0)`. We compare the performance of GMRES, block GMRES with 3 right-hand sides, RGMRES, and our block RGMRES algorithm on all 7 systems. In the case of the block methods, one right-hand side generated by the TRAMONTO package, and the two additional right-hand sides were random, generated using MATLAB's `rand()`. In the case of these Newton iterations for these relatively small systems (dimension ≈ 14000),

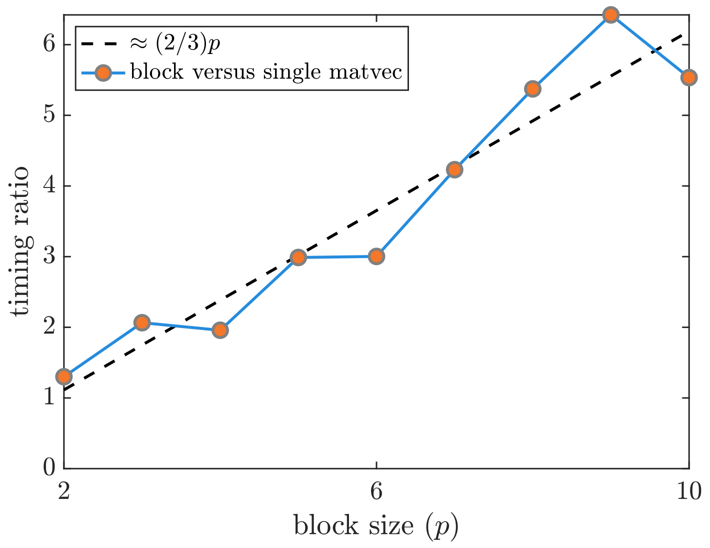

Fig. 2 This figure shows the ratio of time taken for the block matrix-matrix products over the time taken for matrix-vector products for the `sherman5` matrix from [66] for different block sizes p. As predicted, the block p matvec is not p times as expensive. The **dashed line** shows the linear least-squares fit to the experiment data, showing that a block p matvec is roughly $\frac{2}{3}p$ times as expensive as a single matrix-vector product, for this matrix. Each experiment was performed 10^5 times.

convergence for the preconditioned system is quick enough that we are able to recycle the entire Krylov subspace when running rGmres and block rGmres for the first few systems in the sequence before our total subspace dimension exceeds the chosen dimension k and we must down-select by computing harmonic Ritz vectors. In Figure 4, we plot the number of block p matvecs needed to solve each system. Observe that for both algorithms, recycling greatly reduces the number of block p matvecs needed to solve later systems. Furthermore, we get a per-system reduction when moving from rGmres to block rGmres, particularly for systems appearing early in the sequence. In Figure 4, we see that for later systems, Gcro-Dr is able to catch up to the block method in terms of number of iterations. This suggests that for some problems for which we use block methods, the additional expense of recycling may bring the most benefit for the earlier systems. This can yield a high-quality recycled subspace, and we may then be able to apply single-vector rGmres for the rest of the systems. We also see that, for large enough recycled subspace, we achieve a 30% reduction in overall matvecs when moving from single right-hand side rGmres to block rGmres with two random right-hand sides.

6.2 Data movement experiments

In this section, we run a variety of performance tests on matrices of various sparsity patterns and levels coming both from real applications [68] and from test sets we

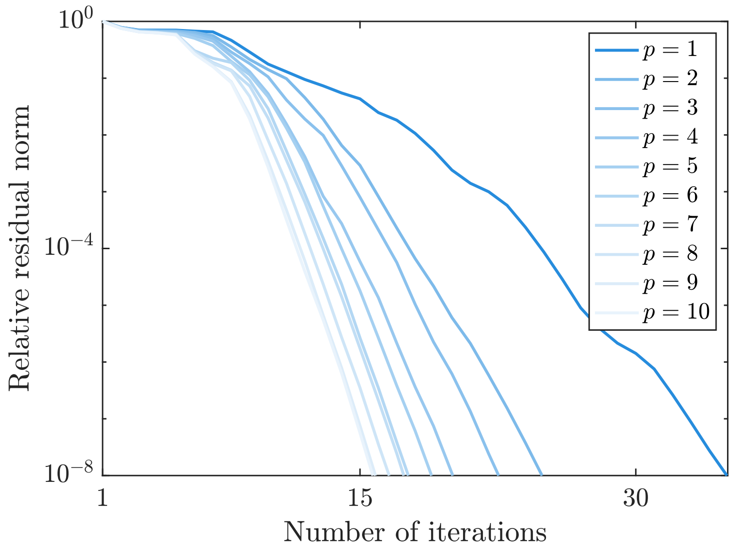

Fig. 3 Performance of block recycled GMRES ($m = 100$; $k = 50$) for the `sherman5` matrix from [66] preconditioned with `ILU(0)` as we increase the number of right-hand sides. The first right-hand side is packaged with the matrix; the others are generated using `rand()`. Convergence is measured by computing the residual norm associated to the first right-hand side.

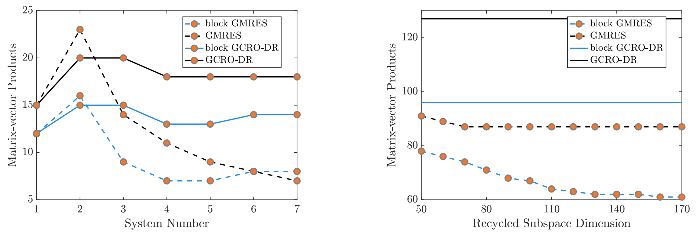

Fig. 4 **Left:** block p matvec count for block recycled GMRES solving linear systems involving the Jacobian for 7 sequential Newton Steps of fluid DFT problem from TRAMONTO software package. ILU(0) preconditioning was used. **Right:** total block p matvec count for various recycled space dimensions Effectively, as we move to the right in this figure, we reach a point at which no recycled subspace information is ever discarded in the experiment.

artificially constructed. The tests were performed on a shared memory machine with 8 Intel Xeon E7-4870 2.4Ghz processors, each with 10 cores (i.e., a total of 80 CPU cores) and a total of 1 terabyte main memory. Each processor has 30 megabytes of L3 cache (3 megabytes per core). Experiments were run in serial. Using compiled TRILINOS codes [23], we compare performance of large sparse operators being applied to blocks of vectors of varying block sizes. For each matrix $\boldsymbol{A}$, we compare multiplying the matrix times the entire block versus multiplying the matrix times each vector

Table 1 Fifteen different matrices downloaded from the University of Florida Sparse Matrix Library [68] with various sparsity patterns. The matrix `qcdRealPart` is the real part of the matrix `conf5-0-4x4-10` from [68]

Name	Dimension	# Non-zeros	Sparsity
`Freescale1`	3428755	17052626	0.0001%
`CoupCons3D`	416800	17277420	0.0099%
`rajat31`	4690002	20316253	0.0001%
`FullChip`	2987012	26621983	0.0003%
`cage14`	1505785	27130349	0.0012%
`RM07R`	381689	37464962	0.0257%
`epb3`	84617	463625	0.0065%
`qcdRealPart`	49152	1916928	0.0793%
`crashbasis`	160000	1750416	0.0068%
`Hamrle3`	1447360	5514242	0.0003%
`HV15R`	2017169	283073458	0.0070%
`lung2`	160000	1750416	0.0068%
`ML_Geer`	1504002	110686677	0.0049%
`pre2`	659033	5834044	0.0013%
`twotone`	120750	1206265	0.0083%

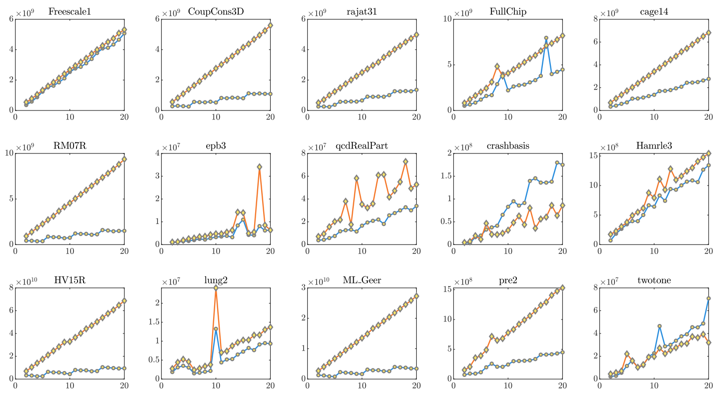

Fig. 5 For the fifteen different matrices from Table 1, a comparison of cache misses for the matrix applied to different sizes of vector blocks both all-at-once (**blue lines with yellow circles**) and one-at-a-time (**red lines with yellow diamonds**). The horizontal axes show block size and the vertical axes show number of cache misses per 100 executions of the experiment.

individually. For each matrix, the experiment is repeated for the projected operator $(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}$ for subspaces $\mathcal{U}$ of different dimensions. Before each test was performed, a block p matvec was executed so that any prefetching of data into the cache would occur before the start of the test and thus would not interfere with our measurements.

Performance is measured in multiple ways. First, each experiment is performed 100 times and the average time in seconds for those experiments is taken. Second, we compiled the Intel Performance Counter Monitor (PCM) libraries [69] and inserted

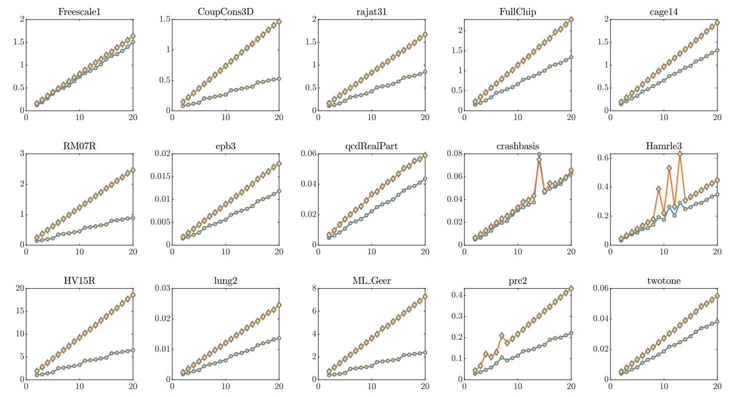

Fig. 6 For the fifteen different matrices from Table 1, a comparison average timings (in seconds) for the projected matrix applied to different sizes of vector blocks both all-at-once (**blue lines with yellow circles**) and one-at-a-time (**red lines with yellow diamonds**). The horizontal axes show block size and the vertical axes show the average time in seconds required for each experiment. Note that in each plot, the vertical axis is in a different scale.

appropriate function calls into our test code, and these were used to take measurements directly from the processor for each experiment. Namely, the PCM allows one to measure bytes read by the processor, the percentage of cache hits, and the number of cache misses occurring during the experiment. A *cache hit* refers to the instance that a wanted piece of data is already stored in cache on the processor, increasing the speed of access. A *cache miss* is when a wanted piece of data is not on the cache and thus must be accessed from main memory, causing a delay due to data movement needs. In our experiments, we demonstrate that often, the sparse block p matvec has superior performance when measured in these cache- and data- related metrics over the sparse matrix-vector product.

In our first set of experiments, we take measurements for fifteen sample matrices arising in a variety of applications, downloaded from [68]. We begin by taking measurements for just the application of the matrix to various sizes of block vectors. In Table 1, names and relevant characteristics of the matrices are shown. In Figure 5, comparisons of cache misses for single- and block-matvecs are shown for block sizes between 2 and 20. In Figure 6, average timings are shown for the same experiments. In Figure 7, we compare the ratio of the time take to multiply the matrix times a block of vectors to the time taken to multiply times just one vector. This demonstrates that it is often the case that multiplying times L vectors is not L-times as expensive as multiplying times a single vector. In these experiments, we see the greatest computational benefit for larger matrices, which is when a matvec becomes an I/O-bound operation, i.e., the rate at which data is used is faster than the rate at which it is retrieved. If one compares the matrix sizes from Table 1 against the data in Figures 5–7, one sees that the greatest performance difference in the two experiments is for the matrices with the

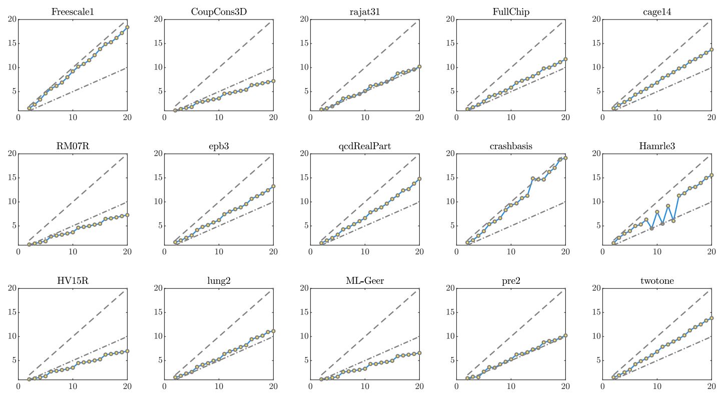

Fig. 7 For the fifteen different matrices from [68], the ratio of the average time taken to apply the matrix to different sizes of vector blocks versus to a single vector. The horizontal axes show block size and the vertical axes show the ratio for each block. For reference, we include references lines for ratio p (gray dashed line) and $\frac{1}{2}p$ (gray dot-dashed line).

largest number of nonzeros (`ML_Geer`, `CoupCons3D`, etc.). For small matrices (e.g., `epb3`) one observes hardly any difference. This confirms that using block operations would likely only provides benefit for large matrices. We note that we can transfer these results to the parallel setting, where we instead consider the situation that the part of the matrix stored on a specific node is large with respect to the L3 cache size. Note that Figure 11 illustrates this relationship; the figures further down and to the right show increasingly larger differences in cache misses between the two experiments.

We then took the same measurements but for the projected operator $(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}$. We show below experiments for the case that $\dim \mathcal{C} = 10$. We also performed the same experiments for $\dim \mathcal{C} = 50$ and $\dim \mathcal{C} = 100$, but the results were not substantially different from those shown. We see in all three experiments that the cache efficiencies observed for applying the matrix to blocks of vectors is diminished when a projector is composed with the operator. In Figures 8 and 9, we see that for many matrices, there is similar performance, in terms of timings and numbers of cache misses, whether the matrix is applied to the full block or to each vector in the block individually. In some cases, one-at-a-time application is actually superior. We also again show the ratio between time taken to multiply the projected matrix times a block of vectors versus multiplying times just one vector. In this experiment, we investigate the difference between using modified Gram-Schmidt or multiple passes of classical Gram-Schmidt (DGKS) [70].

This experience is important when considering the performance of a block recycled GMRES method as compared to standard block GMRES. The cache-efficiency benefits of block methods does not always extend to the orthogonalization routines tested in this paper. Thus it may be more appropriate to compare the performance of block GMRES and recycled block GMRES for total search space dimension being approximately the

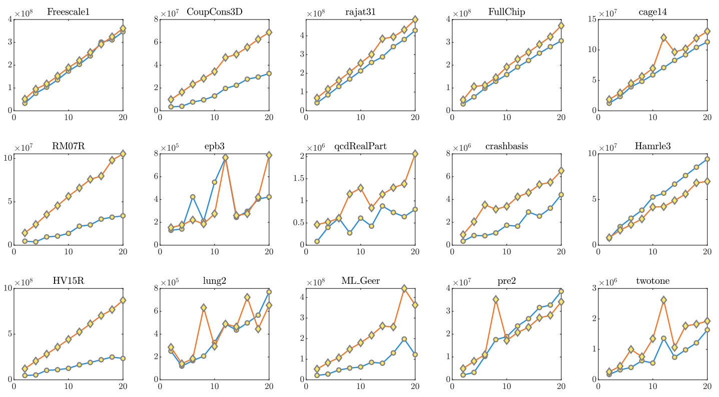

Fig. 8 For the fifteen different matrices from Table 1, a comparison of cache misses for the projected matrix $(\boldsymbol{I} - \boldsymbol{\Phi})\boldsymbol{A}$ applied to different sizes of vector blocks both all-at-once (**blue lines with yellow circles**) and one-at-a-time (**red lines with yellow diamonds**). The projector was applied using a modified Gram-Schmidt algorithm, and $\boldsymbol{C} \in \mathbb{R}^{n\times 100}$. The horizontal axes show block size and the vertical axes show number of cache misses per 100 executions of the experiment.

same, so that the number of orthogonalization is equivalent for both methods. There is much work exploring other methods for efficient, stable orthogonalization in the HPC setting; see e.g., [27, 28, 71–74]. Additionally, using all manner of sketched bases to reduce the computational issues concerning orthogonalization is an active area of exploration [75], and some authors also explore schemes that do not follow the GCRO approach and instead choose an unprojected Krylov subspace (i.e., generated by $\boldsymbol{A}$ rather than $(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}$); see [75, 76].

In our second set of experiments, we repeat the same tests but for some large, sparse, banded matrices. These were constructed in MATLAB using `rand()` and `spdiags()` and saved to disk and then loaded by our compiled code for tests in TRILINOS. The purpose of running tests on such matrices is to give a clear picture of the cache performance for sparse block p matvecs for matrices whose structure is easily understood but similar to what often arises in the discretization of differential operators. These experiments supplement the first set, in which we use matrices arising from real applications but whose structure is not as simple. We constructed matrices with dimensions of 10^4, 10^5, 10^6, and 10^7. Matrices with banded sizes of 4, 10, 20, 30, 40, and 50 were constructed. Block sizes tested were the even integers in the interval $[2, 20]$. As the previous experiments demonstrated that there is a great loss of cache efficiency when applying the projected operator, we restrict these experiments to the case of the unprojected operator. In Figure 11, we compare the number of cache misses encountered when applying the matrices to a block of vector at once versus to the same block one column at-a-time. In Figure 12 we compare average timings for the same two cases. In Figure 13, we calculate the ratio between the average time taking to apply each operator to a block of vectors versus applying it to a single vector. This

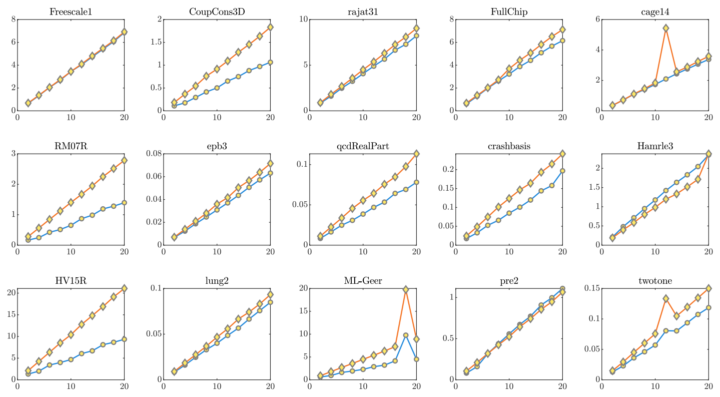

Fig. 9 For the fifteen different matrices from [68], a comparison average timings (in seconds) for the projected matrix $(\boldsymbol{I}-\boldsymbol{\Phi})\boldsymbol{A}$ applied to different sizes of vector blocks both all-at-once (**blue lines with yellow circles**) and one-at-a-time (**red lines with yellow diamonds**). The projector was applied using a modified Gram-Schmidt algorithm, and $\boldsymbol{C} \in \mathbb{R}^{n\times 100}$. The horizontal axes show block size and the vertical axes show the average time in seconds required for each experiment.

to a large extent indicates how much more expensive an iteration (dominated by the cost of the sparse matrix application) will be for a block method versus the non-block version of that method. We see again for these artificially constructed sparse matrices that we benefit from data movement efficiency for block methods. Furthermore, because the structures of these matrices is precisely known, it is easier to compare results for different matrices in this group.

7 Discussion and conclusions

We chose to restrict the experiments in Section 6.2 to the matrix application, as it is the main computational kernel of a Krylov subspace-based iteration. We focus on the performance of the application of $\boldsymbol{A}$ and of $(\boldsymbol{I}-\boldsymbol{\Phi})$ as they are two of the most dominant costs in Algorithm 4.1. Many of the other operations are dense matrix-matrix operations with already confirmed cache efficiency characteristics. In particular, we use the Householder transformation block storage and application strategy of Gutknecht and Schmelzer [46] which means application of the previous Householder transformations is also a matrix-matrix operation. Thus the experiments presented in Section 6.2 are analogous to per iteration costs of our method, and for the unprojected operator any block method; see, e.g., [22]. We have seen that there is a drop in cache efficiency when applying the projected operator. This is perhaps not surprising. When the operator $\boldsymbol{A}$ is being applied, perfect cache efficiency arises from being able to store the entirety of $\boldsymbol{A}$ (a sparse matrix) in cache. For the projected operator, we would in addition need to be able to fit $\boldsymbol{C}$ (a dense block of vectors) into cache, as well. This becomes less likely as the number of non-zeros of $\boldsymbol{C}$ approaches the number of non-zeros of $\boldsymbol{A}$.

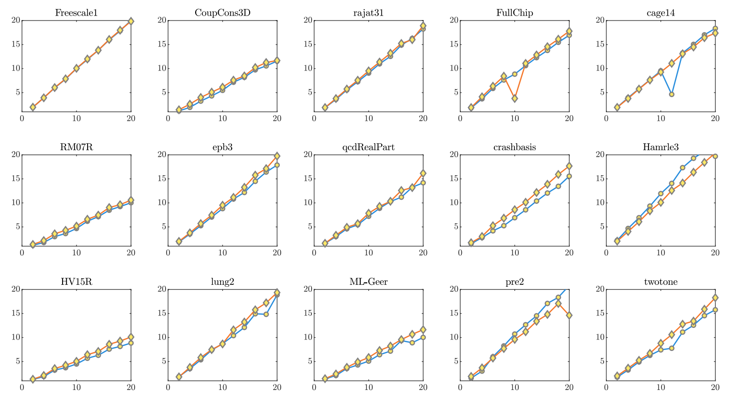

Fig. 10 For the fifteen different matrices from [68], the ratio of the average time taken to apply the matrix to different sizes of vector blocks versus to a single vector for the projected operator $(\boldsymbol{I} - \boldsymbol{\Phi})\,\boldsymbol{A}$. The projector is applied using either a modified Gram-Schmidt algorithm (**blue lines with yellow circles**) or multiple passes of classical Gram-Schmidt (**red lines with yellow diamonds**), and $\boldsymbol{C} \in \mathbb{R}^{n\times 100}$. The horizontal axes show block size and the vertical axes show the ratio for each block size.

The more vectors we recycle, the less likely that is to happen. We refer the reader to [27, 28] wherein the authors present implementations remedying issues such as this.

A C++ implementation of Algorithm 4.1 is available as a part of the Belos package of Trilinos [14], and a Matlab implementation is available at [15]. Furthermore, as we have only shown a small subset of the data coming from our performance results, we also provide tables and csv files containing the full, raw performance results as a supplement to this paper.

Acknowledgments

The authors would like to thank Florian Tischler of the Johann Radon Institute for Computational and Applied Mathematics for technical assistance and for allowing the second author to have access to sensitive processor-level functions to take necessary measurements in Section 6.2. The initial work for this project was undertaken while the second author completed a graduate research internship at Sandia Laboratories in Albuquerque as a guest of the first author. The first and second authors would like to thank Mark Hoemmen for engaging in many helpful conversations and the first author also would like to thank Mike Heroux for the same.

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of

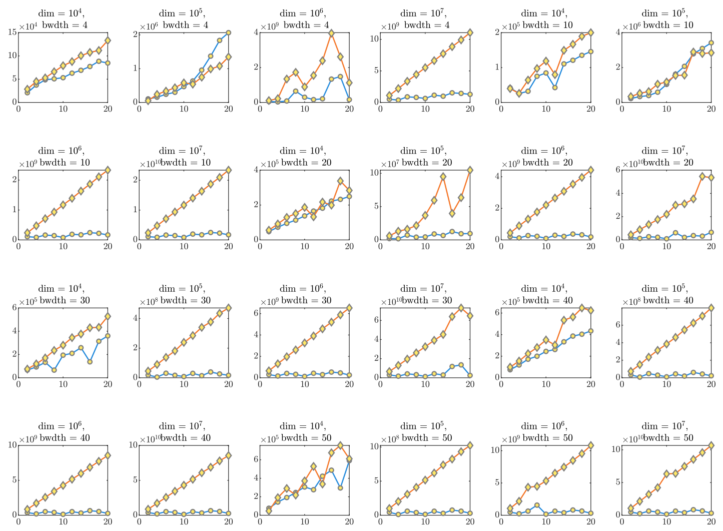
Fig. 11 For banded matrices of different dimensions, a comparison of cache misses for the matrix applied to different sizes of vector blocks both all-at-once (**blue lines with yellow circles**) and one-at-a-time (**red lines with yellow diamonds**). The horizontal axes show block size and the vertical axes show number of cache misses per 100 executions of the experiment.

federally sponsored research in accordance with the DOE Public Access Plan (https://www.energy.gov/doe-public-access-plan).

References

[1] Jin, C., Cai, X.-C.: A preconditioned recycling GMRES solver for stochastic helmholtz problems. Communications in Computational Physics **6**(2), 342 (2009)

[2] Jin, C., Cai, X.-C., Li, C.: Parallel domain decomposition methods for stochastic elliptic equations. SIAM Journal on Scientific Computing **29**(5), 2096–2114 (2007)

[3] Heroux, M.A., Salinger, A.G., Frink, L.J.D.: Parallel segregated Schur complement methods for fluid density functional theories. SIAM Journal on Scientific Computing **29**(5), 2059–2077 (2007) https://doi.org/10.1137/060661594

[4] Wang, S., Sturler, E., Paulino, G.H.: Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. International Journal for Numerical Methods in Engineering **69**(12), 2441–2468 (2007) https://doi.org/10.1002/nme.1798

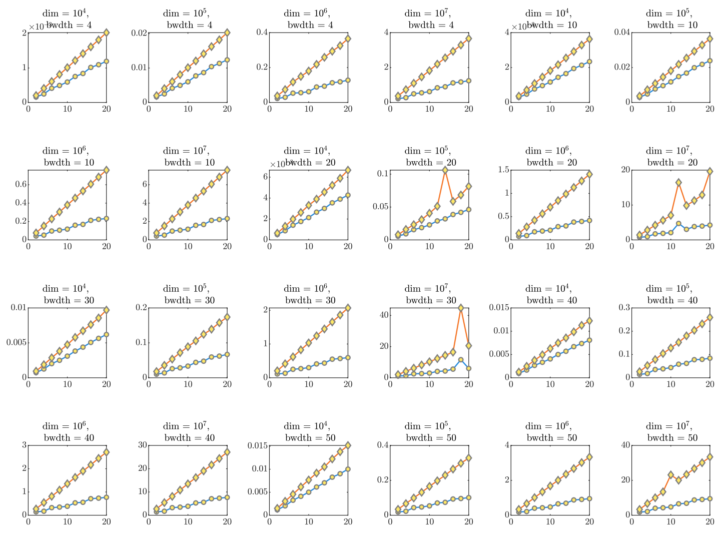

Fig. 12 For narrow bandwidth matrices of different dimensions and bandwidths, a comparison average timings (in seconds) for the projected matrix applied to different sizes of vector blocks both all-at-once **(blue lines with yellow circles)** and one-at-a-time **(gray lines)**. The horizontal axes show block size and the vertical axes show the average time in seconds required for each experiment.

[5] Parks, M.L., Sturler, E., Mackey, G., Johnson, D.D., Maiti, S.: Recycling Krylov subspaces for sequences of linear systems. SIAM Journal on Scientific Computing **28**(5), 1651–1674 (2006) https://doi.org/10.1137/040607277

[6] Kilmer, M.E., Sturler, E.: Recycling subspace information for diffuse optical tomography. SIAM Journal on Scientific Computing **27**(6), 2140–2166 (2006) https://doi.org/10.1137/040610271

[7] O'Leary, D.P.: The block conjugate gradient algorithm and related methods. Linear Algebra and its Applications **29**, 293–322 (1980)

[8] Vital, B.: Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur. PhD thesis, Informatique, Université de Rennes (1990)

[9] Chronopoulos, A.T., Kucherov, A.B.: Block-s-step Krylov iterative methods. Numerical Linear Algebra with Applications **17**(1), 3–15 (2010) https://doi.org/10.1002/nla.643

[10] O'Leary, D.P.: Parallel implementation of the block conjugate gradient algorithm.

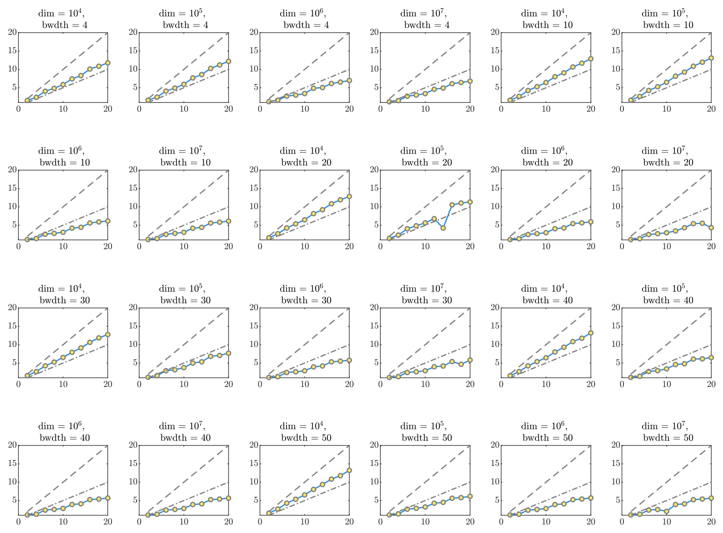

Fig. 13 For narrow bandwidth matrices of different dimensions and bandwidths, the ratio of the average time taken to apply the matrix to different sizes of vector blocks versus to a single vector **(blue lines with yellow circles)**. The horizontal axes show block size and the vertical axes show the ratio for each block. For reference, we include references lines for ratio p **(gray dashed line)** and $\frac{1}{2}p$ **(gray dot-dashed line)**.

Parallel Computing **5**(1–2), 127–139 https://doi.org/10.1016/0167-8191(87)90013-5

[11] Soodhalter, K.M.: A block MINRES algorithm based on the banded Lanczos method. Numerical Algorithms **69**(3), 473–494 (2015) https://doi.org/10.1007/s11075-014-9907-z

[12] Parks, M.L., Soodhalter, K.M., Szyld, D.B.: A block recycled gmres method with investigations into aspects of solver performance. Technical report (2016). arXiv e-print 1604.01713v1. https://arxiv.org/abs/1604.01713v1

[13] Soodhalter, K.M.: Krylov subspace methods with fixed memory requirements: Nearly hermitian linear systems and subspace recycling. PhD thesis, Temple University, Department of Mathematics (2012)

[14] Parks, M.L., Soodhalter, K.M.: Block GCRO-DR. in Belos package of the Trilinos C++ Library (2011). https://trilinos.org/docs/dev/packages/belos/doc/html/classBelos_1_1BlockGCRODRSolMgr.html

[15] Parks, M.L., Soodhalter, K.M., Szyld, D.B.: Block GCRO-DR: A version of the recycled GMRES method using block Krylov subspaces and harmonic Ritz vectors. Available at http://dx.doi.org/10.5281/zenodo.48836 (2016). https://doi.org/10.5281/zenodo.48836

[16] Gutknecht, M.H.: Block Krylov space methods for linear systems with multiple right-hand sides: an introduction. In: Siddiqi, A.H., Duff, I.S., Christensen, O. (eds.) Modern Mathematical Models, Methods and Algorithms for Real World Systems, pp. 420–447. Anamaya Publishers, New Delhi (2007)

[17] Littlewood, D.: Roadmap for software implementation. In: Bobaru, F., Foster, J.T., Geubelle, P.H., Silling, S.A. (eds.) Handbook of Peridynamic Modeling. Advanced in Applied Mathematics, pp. 109–140. CRC Press, Boca Raton, FL (2016). Chap. 5

[18] Soodhalter, K.M., Sturler, E., Kilmer, M.E.: A survey of subspace recycling iterative methods. GAMM-Mitteilungen **43**(4) https://doi.org/10.1002/gamm.202000016

[19] Shalf, J., Dosanjh, S., Morrison, J.: Exascale computing technology challenges. In: Palma, J.M.L.M., Daydé, M., Marques, O., Correia Lopes, J.a. (eds.) High Performance Computing for Computational Science – VECPAR 2010. Lectures Notes in Computer Science, vol. 6449, pp. 1–25. Springer, Berlin, Heidelberg (2011)

[20] Hoemmen, M.: Communication-avoiding krylov subspace methods. PhD thesis, Department of Computer Science, University of California Berkeley (2010)

[21] Mohiyuddin, M., Hoemmen, M., Demmel, J., Yelick, K.: Minimizing communication in sparse matrix solvers. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. SC '09, pp. 36–13612. ACM, New York, NY, USA (2009). https://doi.org/10.1145/1654059.1654096 . http://doi.acm.org/10.1145/1654059.1654096

[22] Baker, A.H., Dennis, J.M., Jessup, E.R.: On improving linear solver performance: a block variant of GMRES. SIAM Journal on Scientific Computing **27**(5), 1608–1626 (2006) https://doi.org/10.1137/040608088

[23] The Trilinos Project Website. Accessed December 10, 2014 at http://trilinos.org/. (2014)

[24] Giraud, L., Jing, Y.-F., Xiang, Y.: A block minimum residual norm subspace solver with partial convergence management for sequences of linear systems. SIAM Journal on Matrix Analysis and Applications **43**(2), 710–739 https://doi.org/10.1137/21m1401127

[25] Darnell, D., Morgan, R.B., Wilcox, W.: Deflated GMRES for systems with multiple shifts and multiple right-hand sides. Linear Algebra and its Applications **429**(10), 2415–2434 (2008) https://doi.org/10.1016/j.laa.2008.04.019

[26] Meng, J., Zhu, P.-Y., Li, H.-B.: A block GCROT(m,k) method for linear systems with multiple right-hand sides. Journal of Computational and Applied Mathematics **255**, 544–554 (2014) https://doi.org/10.1016/j.cam.2013.06.014

[27] Audibert, L., Girardon, H., Haddar, H., Jolivet, P.: Inversion of eddy-current signals using a level-set method and block krylov solvers. SIAM Journal on Scientific Computing **45**(3), 366–389 https://doi.org/10.1137/20m1382064

[28] Jolivet, P., Tournier, P.-H.: Block iterative methods and recycling for improved scalability of linear solvers. In: SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 190–203. IEEE. https://doi.org/10.1109/sc.2016.16 . http://dx.doi.org/10.1109/SC.2016.16

[29] Saad, Y.: Iterative Methods for Sparse Linear Systems, Second edn. SIAM, Philadelphia (2003)

[30] Simoncini, V., Szyld, D.B.: Recent computational developments in Krylov subspace methods for linear systems. Numerical Linear Algebra with Applications **14**(1), 1–59 (2007) https://doi.org/10.1002/nla.499

[31] Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing **7**, 856–869 (1986)

[32] Kubínová, M., Soodhalter, K.M.: Admissible and attainable convergence behavior of block arnoldi and gmres. SIAM Journal on Matrix Analysis and Applications **41**(2), 464–486 https://doi.org/10.1137/19m1272469

[33] Simoncini, V., Gallopoulos, E.: Convergence properties of block GMRES and matrix polynomials. Linear Algebra and its Applications **247**, 97–119 (1996) https://doi.org/10.1016/0024-3795(95)00093-3

[34] de Sturler, E.: Nested Krylov methods based on GCR. Journal of Computational and Applied Mathematics **67**(1), 15–41 (1996) https://doi.org/10.1016/0377-0427(94)00123-5

[35] Lehoucq, R.B., Sorensen, D.C.: Deflation techniques for an implicitly restarted Arnoldi iteration. SIAM Journal on Matrix Analysis and Applications **17**(4), 789–821 (1996) https://doi.org/10.1137/S0895479895281484

[36] Morgan, R.B.: GMRES with deflated restarting. SIAM Journal on Scientific Computing **24**(1), 20–37 (2002) https://doi.org/10.1137/S1064827599364659

[37] de Sturler, E.: Truncation strategies for optimal Krylov subspace methods. SIAM Journal on Numerical Analysis **36**(3), 864–889 (1999) https://doi.org/10.1137/S0036142997315950

[38] Baker, A.H., Jessup, E.R., Manteuffel, T.: A technique for accelerating the convergence of restarted GMRES. SIAM Journal on Matrix Analysis and Applications **26**(4), 962–984 (2005) https://doi.org/10.1137/S0895479803422014

[39] Eiermann, M., Ernst, O.G., Schneider, O.: Analysis of acceleration strategies for restarted minimal residual methods. Journal of Computational and Applied Mathematics **123**(1-2), 261–292 (2000) https://doi.org/10.1016/S0377-0427(00)00398-8

[40] Saad, Y.: Analysis of augmented Krylov subspace methods. SIAM Journal on Matrix Analysis and Applications **18**(2), 435–449 (1997) https://doi.org/10.1137/S0895479895294289

[41] Gaul, A.: Recycling Krylov subspace methods for sequences of linear systems: Analysis and applications. PhD thesis, Fakultät II – Mathematik und Naturwissenschaften, Technischen Universität Berlin (2014)

[42] Simoncini, V., Szyld, D.B.: On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. SIAM Review **47**(2), 247–272 (2005) https://doi.org/10.1137/S0036144503424439

[43] Campbell, S.L., Ipsen, I.C.F., Kelley, C.T., Meyer, C.D.: GMRES and the minimal polynomial. BIT Numerical Mathematics **36**(4), 664–675 (1996) https://doi.org/10.1007/BF01733786

[44] Vorst, H.A., Vuik, K.: The superlinear convergence behaviour of GMRES. Journal of Computational and Applied Mathematics **48**(3), 327–341 (1993) https://doi.org/10.1016/0377-0427(93)90028-A

[45] de Sturler, E.: Convergence Bounds for Approximate Invariant Subspace Recycling for Sequences of Linear Systems. In: Program of the Householder Symposium XVIII on Numerical Linear Algebra, pp. 51–52 (2011)

[46] Gutknecht, M.H., Schmelzer, T.: Updating the QR decomposition of block tridiagonal and block Hessenberg matrices. Applied Numerical Mathematics **58**(6), 871–883 (2008) https://doi.org/10.1016/j.apnum.2007.04.010

[47] Greenbaum, A., Pták, V., Strakoš, Z.: Any nonincreasing convergence curve is possible for GMRES. SIAM Journal on Matrix Analysis and Applications **17**, 465–469 (1996)

[48] Morgan, R.B.: A restarted GMRES method augmented with eigenvectors. SIAM Journal on Matrix Analysis and Applications **16**(4), 1154–1171 (1995) https://

doi.org/10.1137/S0895479893253975

[49] Morgan, R.B.: Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. SIAM Journal on Matrix Analysis and Applications **21**(4), 1112–1135 (2000) https://doi.org/10.1137/S0895479897321362

[50] Abdel-Rehim, A.M., Morgan, R.B., Nicely, D.A., Wilcox, W.: Deflated and restarted symmetric Lanczos methods for eigenvalues and linear equations with multiple right-hand sides. SIAM Journal on Scientific Computing **32**(1), 129–149 (2010) https://doi.org/10.1137/080727361

[51] Ruhe, A.: Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. Mathematics of Computation **33**(146), 680–687 (1979) https://doi.org/10.2307/2006302

[52] Agullo, E., Giraud, L., Jing, Y.-F.: Block GMRES method with inexact breakdowns and deflated restarting. SIAM Journal on Matrix Analysis and Applications **35**(4), 1625–1651 (2014) https://doi.org/10.1137/140961912

[53] Gallivan, K.A., Heath, M.T., Ng, E., Ortega, J.M., Peyton, B.W., Plemmons, R.J., Romine, C.H., Sameh, A.H., Voigt, R.G.: Parallel Algorithms for Matrix Computations. Society for Industrial and Applied Mathematics, Philadelphia, USA (1990)

[54] Parks, M.L.: Original GCRO-DR Implementation. Zenodo. https://doi.org/10.5281/ZENODO.15163388 . https://zenodo.org/doi/10.5281/zenodo.15163388

[55] Ahuja, K., Parks, M.L., Phipps, E.T., Salinger, A.G., Sturler, E.: Krylov recycling for climate modeling and uncertainty quantification. Technical Report SAND2010-8783P, Sandia National Laboratories Computer Science Research Institute (2010)

[56] Gaul, A., Schlömer, N.: Preconditioned recycling Krylov subspace methods for self-adjoint problems. Electronic Transactions on Numerical Analysis **44**(1208.0264), 522–547 (2015)

[57] Carlberg, K., Forstall, V., Tuminaro, R.: Krylov-subspace recycling via the pod-augmented conjugate-gradient method. SIAM Journal on Matrix Analysis and Applications **37**(3), 1304–1336 (2016) https://doi.org/10.1137/16M1057693

[58] Morikuni, K., Reichel, L., Hayami, K.: FGMRES for linear discrete ill-posed problems. Applied Numerical Mathematics **75**, 175–187 (2014) https://doi.org/10.1016/j.apnum.2013.08.004

[59] Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. SIAM Journal on Scientific Computing **14**(2), 461–469 (1993) https://doi.org/10.1137/0914028

[60] Baglama, J., Reichel, L.: Augmented GMRES-type methods. Numerical Linear Algebra with Applications **14**(4), 337–350 (2007) https://doi.org/10.1002/nla.518

[61] Baglama, J., Reichel, L.: Decomposition methods for large linear discrete ill-posed problems. Journal of Computational and Applied Mathematics **198**(2), 332–343 (2007) https://doi.org/10.1016/j.cam.2005.09.025

[62] Dong, Y., Garde, H., Hansen, P.C.: R^{3}GMRES: including prior information in GMRES-type methods for discrete inverse problems. Electronic Transactions on Numerical Analysis **42**, 136–146 (2014)

[63] Soodhalter, K.M.: A note on augmented unprojected krylov subspace methods. ETNA - Electronic Transactions on Numerical Analysis **55**, 532–546 https://doi.org/10.1553/etna_vol55s532

[64] Simoncini, V., Szyld, D.B.: The effect of non-optimal bases on the convergence of Krylov subspace methods. Numerische Mathematik **100**(4), 711–733 (2005)

[65] Carson, E., Liesen, J., Strakoš, Z.: Towards Understanding CG and GMRES Through Examples. https://arxiv.org/abs/2211.00953

[66] Matrix Market Website. http://math.nist.gov/MatrixMarket/. (2011)

[67] Description of the POLY1-CMS-1D problems. Accessed February 24, 2012 at https://software.sandia.gov/tramonto/src_docs/POLY1__CMS__1D_2README.html (2012)

[68] Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Transactions in Mathematical Software **38**(1), 1–1125 (2011) https://doi.org/10.1145/2049662.2049663

[69] Intel Performance Counter Monitor. Accessed on 28 February, 2016 at http:\www.intel.com/software/pcm (2016)

[70] Daniel, J., Gragg, W.B., Kaufman, L., Stewart, G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. Mathematics of Computation **30**(136), 772–795 (1976)

[71] Carson, E., Lund, K., Rozložník, M.: The stability of block variants of classical gram–schmidt. SIAM Journal on Matrix Analysis and Applications **42**(3), 1365–1380 https://doi.org/10.1137/21m1394424

[72] Carson, E., Ma, Y.: On the backward stability of s-step gmres. Technical report (2024). arXiv e-print 2409.03079. https://arxiv.org/abs/2411.07077

[73] Carson, E., Lund, K., Ma, Y., Oktay, E.: On the loss of orthogonality in low-synchronization variants of reorthogonalized block classical gram-schmidt.

Technical report (2024). arXiv e-print 2408.10109. https://arxiv.org/abs/2210.08839

[74] Carson, E., Lund, K., Ma, Y., Oktay, E.: Reorthogonalized pythagorean variants of block classical gram–schmidt. SIAM Journal on Matrix Analysis and Applications **46**(1), 310–340 https://doi.org/10.1137/24m1658723

[75] Burke, L., Güttel, S., Soodhalter, K.M.: GMRES with Randomized Sketching and Deflated Restarting. Accepted in SIMAX

[76] Burke, L., Frommer, A., Ramirez-Hidalgo, G., Soodhalter, K.M.: Krylov Subspace Recycling For Matrix Functions. arXiv preprint: https://arxiv.org/abs/2209.14163, Submitted for publication