NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
*Numer. Linear Algebra Appl.* 2015; **22**:1–25
Published online 1 May 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1935

# Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems

Martin B. van Gijzen[1], Gerard L. G. Sleijpen[2] and Jens-Peter M. Zemke[3,*,†]

[1]*Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands*
[2]*Mathematical Institute, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, The Netherlands*
[3]*Institut für Mathematik, Technische Universität Hamburg-Harburg, Schwarzenbergstr. 95, D-21073 Hamburg, Germany*

SUMMARY

We give two generalizations of the induced dimension reduction (IDR) approach for the solution of linear systems. We derive a flexible and a multi-shift quasi-minimal residual IDR variant. These variants are based on a generalized Hessenberg decomposition. We present a new, more stable way to compute basis vectors in IDR. Numerical examples are presented to show the effectiveness of these new IDR variants and the new basis compared with existing ones and to other Krylov subspace methods. Copyright © 2014 John Wiley & Sons, Ltd.

Received 14 December 2012; Revised 24 February 2014; Accepted 9 March 2014

KEY WORDS: iterative methods; IDR; IDR(s); quasi-minimal residual; Krylov subspace methods; large sparse nonsymmetric linear systems

## 1. INTRODUCTION

IDR($s$) [1, 2] is a family of fast methods for solving linear systems

$$\mathbf{Ax} = \mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$$

with $\mathbf{A} \in \mathbb{C}^{N\times N}$ a large, sparse, and, in general, non-symmetric matrix. IDR(s) has attracted considerable attention: for an overview, see [3–5], for analysis and more recent generalizations, see [6–10]. In this contribution, we present a new, more stable approach to compute basis vectors and extend the family by two new members of type IDR, more specifically, of type quasi-minimal residual IDR (QMRIDR): a *flexible* QMRIDR (FQMRIDR) variant and a *multi-shift* QMRIDR (MSQMRIDR) variant.

### *1.1. Motivation*

The analysis contained in [6, 7, 9, 10] clearly reveals that IDR methods are specially structured Krylov subspace methods. As the field of Krylov subspace methods has been researched for quite a while, see e.g. [11] and the references therein, many ideas successfully applied there should carry over to the context of IDR based methods with little effort. In this note, we sketch two such generalizations, namely the implementation of a *flexible* IDR method and the implementation of a *multi-shift* IDR method. Both these methods are based on a minimal residual (MR) approach like MinRes [12], GMRes [13], or QMR [14]. The prototype IDR($s$) in [1] relies on an orthogonal residual (OR) approach in OrthoRes-style, i.e., the basis vectors are simply the residuals. OR approaches

*Correspondence to: Jens-Peter M. Zemke, Institut für Mathematik, Technische Universität Hamburg-Harburg, Schwarzenbergstr. 95, D-21073 Hamburg, Germany.
†E-mail: zemke@tu-harburg.de

such as CG [15] or FOM [13, 16] break down whenever the underlying (oblique or orthogonal) projection of the operator $\mathbf{A}$ is singular and lead, in general, to badly conditioned updates for the approximate solution vectors. It is well known that other choices of basis vectors lead to a more stable scheme and that methods based on MR approaches still can be carried out in case of intermediate singularities.

For these reasons, we sketch a scheme to come up with a more stable set of basis vectors, exhibit the underlying structure of a generalized Hessenberg decomposition, show how to apply the MR approach, and apply the flexible and the multi-shift paradigms from the context of Krylov subspace methods. The difference in the latter is the one between Hessenberg decompositions and *generalized* Hessenberg decompositions, i.e., the choice of vectors that are multiplied by $\mathbf{A}$: in classical Krylov methods the last basis vector is multiplied by $\mathbf{A}$, in IDR based methods, a linear combination $\mathbf{v}$ of previously obtained basis vectors is multiplied by (a linear polynomial of exact degree 1 in) $\mathbf{A}$. The adoption of flexible and multi-shift paradigms to the recent extension IDRstab [9] of the IDR approach to use higher degree polynomials will be treated elsewhere.

### *1.2. Related* IDR methods

Because BiCGStab [17, 18] is mathematically an IDR method [7], the first implementation of a quasi-minimal residual IDR method, or QMRIDR method for short, is QMRCGStab [19], the QMR implementation of BiCGStab. More recently, a QMR variant of the prototype IDR($s$) [1] has been proposed by Du *et al*. [20, 21]. This method is mathematically different from the one we present in this paper. The quasi-minimization in the variant of Du *et al*. is performed over the space that is spanned by the residuals generated by the prototype IDR($s$) variant [1]. Because the residuals are used as basis vectors, the QMRIDR variant of Du *et al*. shares the problems of the latter, especially the numerical instability arising for larger values of $s$. To circumvent this problem, our QMRIDR approach is based on a more stable basis expansion, in which new basis vectors are orthonormalized against all basis vectors in the same Sonneveld space [9]. As a consequence, our QMRIDR method is mathematically equivalent to GMRes in the first $s$ iterations.

In [22], a multi-shift IDR method is presented. This method is based on the prototype-IDR($s$) variant and exploits the collinearity of the residuals, i.e., it uses the OR approach.

### *1.3. Outline*

In Section 2, we sketch an implementation of an IDR variant intended to compute a basis in a stable manner. The arising coefficients are gathered in a generalized Hessenberg decomposition in Section 2.2. This generalized Hessenberg decomposition forms in Section 3 the basis to derive a QMRIDR implementation along the lines of the QMR approach in [14]. The flexible QMRIDR variant is developed in Section 4, the multi-shift QMRIDR variant in Section 5. For the reader's convenience pseudo-code implementations of the main algorithms are collected in Section 6; illustrating numerical examples are given in Section 7. We conclude with some remarks about possible generalizations in Section 8.

### *1.4. Notation*

We use standard notation. The system matrix is denoted by $\mathbf{A} \in \mathbb{C}^{N\times N}$, the identity matrix of size $n$ by letter $\mathbf{I} = \mathbf{I}_n \in \mathbb{C}^{n\times n}$, its columns by $\mathbf{e}_j \in \mathbb{C}^n$, $1 \leqslant j \leqslant n$, and its elements by $\delta_{ij}$, $1 \leqslant i, j \leqslant n$. There exist extended variants of the identity matrix and its columns: $\underline{\mathbf{I}}_n \in \mathbb{C}^{(n+1)\times n}$ denotes $\mathbf{I}_n$ with an additional zero row appended at the bottom, and $\underline{\mathbf{e}}_j$, $1 \leqslant j \leqslant n$ denotes its columns. The zero matrix of size $k$ is denoted by $\mathbf{O}_k$, a zero vector of length $k$ by $\mathbf{o}_k$. In context of Krylov methods, unreduced Hessenberg matrices $\mathbf{H}_n \in \mathbb{C}^{n\times n}$ and their unreduced extended counterparts $\underline{\mathbf{H}}_n \in \mathbb{C}^{(n+1)\times n}$ naturally arise. To simplify notation, we use $\underline{\mathbf{U}}_n \in \mathbb{C}^{(n+1)\times n}$ to denote the upper triangular matrix $\mathbf{U}_n \in \mathbb{C}^{n\times n}$ appended with an extra zero row at the bottom. The columns of matrices are denoted by the same lowercase letter. The transpose and complex conjugate transpose of matrices and vectors are denoted by appending $^{\mathrm{T}}$ and $^{\mathrm{H}}$, respectively. The Moore–Penrose or pseudoinverse is denoted by appending $^{\dagger}$. Subspaces are denoted by calligraphic letters like $\mathcal{S}$, $\mathcal{K}_n$,



DOI: 10.1002/nla

and $\mathcal{G}_j$. In this paper, the notation $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ denotes *strict* inclusion of sets, which implies that $\mathcal{G}_j \neq \mathcal{G}_{j-1}$. The Euclidean norm for vectors and the corresponding operator norm for matrices is denoted throughout by $\|\cdot\|$. The letter $n$ is reserved to denote the current step, e.g., $n \in \mathbb{N}$; $s \in \mathbb{N}$ is the codimension of the space $\mathcal{S}$ defining IDR($s$); the parameter $j \in \mathbb{N}_0$ denotes the index of the current Sonneveld space. By nature of IDR methods, $j = \lfloor n/(s+1) \rfloor$ depends on $n$ and $s$, where $\lfloor x \rfloor \in \mathbb{Z}$ denotes the largest integer with $\lfloor x \rfloor \leqslant x \in \mathbb{R}$. Similarly, $\lceil x \rceil \in \mathbb{Z}$ denotes the smallest integer with $\mathbb{R} \ni x \leqslant \lceil x \rceil$. We remark that, like in [6], we use a simplified way to denote Krylov subspace methods, e.g., we write GMRes in place of GMRES as is done in the original publication, because the acronym stands for the phrase Generalized Minimal RESidual.

## 2. A GENERALIZED HESSENBERG RELATION FOR GENERATING VECTORS IN SONNEVELD SPACES

In this section, we review the principle of induced dimension reduction (IDR) methods and give a stable algorithm for generating vectors $\mathbf{g}_1, \ldots, \mathbf{g}_{n+1}$ in the nested Sonneveld spaces $\mathcal{G}_0, \ldots, \mathcal{G}_j$, $j = \lfloor (n+1)/(s+1) \rfloor$, to be defined in the succeeding text. As was shown in [6], the matrices

$$\mathbf{G}_n = (\mathbf{g}_1, \ldots, \mathbf{g}_n), \qquad \mathbf{G}_{n+1} = (\mathbf{G}_n, \mathbf{g}_{n+1}) \tag{1}$$

built from such vectors satisfy a so-called generalized Hessenberg decomposition [6, p. 287, Eqn (1.13)]

$$\mathbf{A}\mathbf{G}_n\mathbf{U}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n, \tag{2}$$

where $\mathbf{U}_n \in \mathbb{C}^{n\times n}$ is upper triangular and $\underline{\mathbf{H}}_n \in \mathbb{C}^{(n+1)\times n}$ is unreduced extended Hessenberg; the entries of $\mathbf{U}_n$ and $\underline{\mathbf{H}}_n$ are uniquely determined by the recurrence coefficients. IDR($s$) is a short-term recurrence Krylov subspace method; this is reflected in the structure of the Sonneveld pencil $(\mathbf{H}_n = \underline{\mathbf{I}}_n^{\mathrm{T}}\underline{\mathbf{H}}_n, \mathbf{U}_n)$ [6, Definition 4.4] that has upper bandwidth $s$. In the following, we sketch how to obtain a stable set of vectors $\mathbf{g}_1, \ldots, \mathbf{g}_{n+1}$ together with the corresponding generalized Hessenberg decomposition.

IDR methods generate vectors $\mathbf{g}_i$, $i = 1, \ldots, n+1$, in the IDR spaces, a special case of Sonneveld subspaces [9, Definition 2.2, p. 2690], which are nested subspaces of shrinking dimension.

Let the subspace $\mathcal{G}_0$ be the full Krylov subspace $\mathcal{K}(\mathbf{A}, \mathbf{g}_1) = \mathcal{K}_N(\mathbf{A}, \mathbf{g}_1)$:

$$\mathcal{G}_0 = \mathcal{K}(\mathbf{A}, \mathbf{g}_1) = \mathcal{K}_N(\mathbf{A}, \mathbf{g}_1) = \operatorname{span}\{\mathbf{g}_1, \mathbf{A}\mathbf{g}_1, \ldots, \mathbf{A}^{N-1}\mathbf{g}_1\}.$$

In case of non-derogatory $\mathbf{A} \in \mathbb{C}^{N\times N}$ and a generic starting vector $\mathbf{g}_1 \in \mathbb{C}^N$, $\mathcal{G}_0 = \mathbb{C}^N$. Cases exist where $\mathcal{G}_0 \subset \mathbb{C}^N$ [23, Section 6.2]. Starting from $\mathcal{G}_0$, the Sonneveld spaces $\mathcal{G}_j$ are recursively defined by

$$\mathcal{G}_j = (\mathbf{A} - \mu_j\mathbf{I})(\mathcal{G}_{j-1} \cap \mathcal{S}) \quad j = 1, 2, \ldots.$$

Here, $\mathcal{S}$ is a space of codimension $s$, which is best described as the left null space of a fixed, full rank $N \times s$ matrix $\widetilde{\mathbf{R}}_0$:

$$\widetilde{\mathbf{R}}_0 = \begin{pmatrix} \tilde{\mathbf{r}}_1 & \tilde{\mathbf{r}}_2 & \cdots & \tilde{\mathbf{r}}_s \end{pmatrix}.$$

The matrix $\widetilde{\mathbf{R}}_0$ is sometimes referred to as the matrix of *initial shadow residuals* or *shadow vectors*. The columns of $\widetilde{\mathbf{R}}_0$ are typically chosen to be orthonormalized random vectors, for a reason see the convergence analysis of IDR-based methods in [10]. The parameter $\mu_j$ is a complex number that can, in principle, be chosen freely. We will sketch a method to select a 'good' $\mu_j$ in a later section. By construction the Sonneveld spaces are nested, but we can say more:



DOI: 10.1002/nla

*Theorem 2.1 (*IDR Theorem [1])
Under mild conditions on the matrices $\mathbf{A}$ and $\widetilde{\mathbf{R}}_0$,

(i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ for all $\mathcal{G}_{j-1} \neq \{\mathbf{o}\}$, $j > 0$.
(ii) $\mathcal{G}_j = \{\mathbf{o}\}$ for some $j \leqslant N$.

For the proof, we refer to [1, 7]. As a consequence of the IDR theorem, $\mathbf{g}_n = \mathbf{o} \in \mathcal{G}_j = \{\mathbf{o}\}$ for some $n$, i.e., IDR methods are in principle direct methods. Like Lanczos-based methods, the finite termination property is lost in finite precision, and the methods deviate. This is not surprising, as IDR methods are based on some underlying Lanczos process [6, 9] we can characterize the IDR Sonneveld spaces in terms of the orthogonal complement of left block Krylov subspaces

$$\mathcal{K}_j(\mathbf{A}^{\mathrm{H}}, \widetilde{\mathbf{R}}_0) = \left\{ \sum_{i=0}^{j-1} (\mathbf{A}^{\mathrm{H}})^i \widetilde{\mathbf{R}}_0 \mathbf{c}_i \mid \mathbf{c}_i \in \mathbb{C}^s \right\}$$

as

$$\mathcal{G}_j = \{M_j(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_j(\mathbf{A}^{\mathrm{H}}, \widetilde{\mathbf{R}}_0)\}, \quad \text{where} \quad M_j(z) = \prod_{i=1}^{j} (z - \mu_i),$$

[7, Theorem 11, p. 1104].

The Sonneveld spaces $\mathcal{G}_j$ as defined in the preceding text use linear factors of the form $\mathbf{A} - \mu_j \mathbf{I}$. This is slightly different from the definition of the Sonneveld spaces used in [1] that uses linear factors $\mathbf{I} - \omega_j \mathbf{A}$, $\omega_j \neq 0$, i.e., linear factors that are normalized such that the polynomial $1 - \omega_j z$ takes the value one at $z = 0$. The difference in definition of the linear factors stems from the fact that, in [1], the aim is to generate *residuals* in the Sonneveld spaces, which leads to the prototype IDR($s$) algorithm. The present goal, however, is to generate a stable set of vectors $\mathbf{g}_n$. In contrast to residuals, the vectors $\mathbf{g}_n$ can be scaled. This property makes the choice $\mu_j = 0$ admissible, whereas the corresponding choice $\omega_j = \infty$ clearly is not.

### *2.1. An algorithm for generating vectors in $\mathcal{G}_j$*

The algorithm that we describe next is one of many possible algorithms to generate vectors $\mathbf{g}_n \in \mathcal{G}_j$. As is explained in [1, 2], $s + 1$ vectors in $\mathcal{G}_{j-1}$ are needed to compute a vector in $\mathcal{G}_j$. The first vector in a new subspace $\mathcal{G}_j$ is up to a multiple uniquely defined. The other vectors, however, are not. In each Sonneveld space $\mathcal{G}_j$, $j = 0, 1, \ldots$, the algorithm in the succeeding text computes $s + 1$ orthonormalized vectors $\mathbf{g}_i$, $i = j(s + 1) + 1, \ldots, (j + 1)(s + 1)$. We distinguish three different phases in the algorithm: (1) the generation of $s + 1$ vectors in $\mathcal{G}_0$, (2) the generation of the first vector in $\mathcal{G}_j$, $j > 0$, and (3) the computation of $s$ additional vectors in $\mathcal{G}_j$, $j > 0$. We remark that we always use and store as few vectors as possible to compute the next vector; other schemes are possible, see [24].

*2.1.1. Generating $s+1$ vectors in $\mathcal{G}_0$.* The first $s+1$ vectors should be in $\mathcal{K}(\mathbf{A}, \mathbf{g}_1)$ and hence can be generated with any Krylov subspace method. Because in our specific algorithm we want to generate orthonormalized vectors $\mathbf{g}_i$, we use Arnoldi's method. Let $\mathbf{g}_1$ be a normalized starting vector, then the next $s$ vectors $\mathbf{g}_2, \ldots, \mathbf{g}_{s+1}$ are computed by the recursion

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{A}\mathbf{g}_n - \sum_{i=1}^{n} \mathbf{g}_i \beta_{i,n}, \quad n \leqslant s. \tag{3}$$

The parameters $\beta_{i,n}$, $i = 1, \ldots, n$ are uniquely determined by the orthogonality conditions $\mathbf{g}_{n+1} \perp \mathbf{g}_i$, $i = 1, \ldots, n$,

$$\beta_{i,n} = \mathbf{g}_i^{\mathrm{H}} \mathbf{A}\mathbf{g}_n, \quad i = 1, \ldots, n,$$



DOI: 10.1002/nla

and $\beta_{n+1,n} \geqslant 0$ is selected such that the normalization condition $\|\mathbf{g}_{n+1}\|_2 = 1$ is fulfilled, i.e.,

$$\beta_{n+1,n} = \|\mathbf{A}\mathbf{g}_n - \sum_{i=1}^{n} \mathbf{g}_i \beta_{i,n}\|.$$

Note that (3) can also be written as

$$\mathbf{A}\mathbf{g}_n = \sum_{i=1}^{n+1} \mathbf{g}_i \beta_{i,n}, \quad n \leqslant s \quad \Leftrightarrow \quad \mathbf{A}\mathbf{G}_s = \mathbf{G}_{s+1}\underline{\mathbf{H}}_s. \tag{4}$$

In the terminology of [6], the latter equality in (4) is the *Hessenberg decomposition* that captures the quantities from Arnoldi's method. This Hessenberg decomposition is the leading part of the generalized Hessenberg decomposition that captures our QMRIDR variant, e.g., the leading $s \times s$ part of the upper triangular $\mathbf{U}_n$, for all $n \geqslant s$, is the identity matrix $\mathbf{I}_s$.

*2.1.2. Generating the first vector in $\mathcal{G}_j$, $j > 0$.* Suppose that after $n$ iterations (with $n = j(s+1)$, $j > 0$), we have explicitly available the vectors $\mathbf{g}_{n-s}, \ldots, \mathbf{g}_n \in \mathcal{G}_{j-1}$. A vector $\mathbf{v}_n \in (\mathcal{G}_{j-1} \cap \mathcal{S})$ can then be computed by

$$\mathbf{v}_n = \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}, \tag{5}$$

in which the $\gamma_{i,n}$ are uniquely determined by the condition that $\mathbf{v}_n \in \mathcal{S}$:

$$\widetilde{\mathbf{R}}_0^{\mathrm{H}} \mathbf{v}_n = \mathbf{o}. \tag{6}$$

Combining (5) and (6) yields an $s \times s$ linear system from which the parameters $\gamma_{i,n}$, $i = n-s, \ldots, n-1$, can be determined. After selection of a new parameter $\mu_j$, the first vector $\mathbf{t} \in \mathcal{G}_j$ can be computed by

$$\mathbf{t} = (\mathbf{A} - \mu_j \mathbf{I})\mathbf{v}_n. \tag{7}$$

To select a new $\mu_j$, we aim at minimizing the norm of $\mathbf{t}$. In order to avoid a very large value of $\mu_j$, which leads to a small angle between $\mathbf{t}$ and $\mathbf{v}_n$, we combine the minimization with the strategies explained in [25]. We refer to Section 6 for the precise algorithm to compute $\mu_j$. As a final step, we normalize the vector $\mathbf{t}$ which gives us $\mathbf{g}_{n+1}$:

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{t}, \quad \beta_{n+1,n} = \|\mathbf{t}\|. \tag{8}$$

The Eqns (5), (7), and (8) can be combined to give

$$\mathbf{g}_{n+1}\beta_{n+1,n} = (\mathbf{A} - \mu_j \mathbf{I}) \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right),$$

which can be rewritten as

$$\mathbf{A} \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right) = \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right) \mu_j + \mathbf{g}_{n+1}\beta_{n+1,n}. \tag{9}$$



DOI: 10.1002/nla

*2.1.3. Generating s additional vectors in $\mathcal{G}_j$, $j > 0$.* Suppose that after $n$ iterations (with $(j+1)(s+1) > n > j(s+1)$, $j > 0$), we have explicitly available the vectors $\mathbf{g}_{n-s}, \ldots, \mathbf{g}_n \in \mathcal{G}_{j-1}$. Furthermore, suppose that some of these vectors are already known to be in $\mathcal{G}_j$; namely, suppose that $\mathbf{g}_{j(s+1)+1}, \ldots, \mathbf{g}_n \in \mathcal{G}_j$.

A new vector $\mathbf{t} \in \mathcal{G}_j$ can be computed by repeating steps (5)–(7). Because a linear combination of vectors in the subspace $\mathcal{G}_j$ is also in $\mathcal{G}_j$, the vector

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{t} - \sum_{i=j(s+1)+1}^{n} \mathbf{g}_i \beta_{i,n} \tag{10}$$

is also in $\mathcal{G}_j$. The parameters $\beta_{i,n}$ are again selected such that the vectors $\mathbf{g}_{j(s+1)+1}, \ldots, \mathbf{g}_n \in \mathcal{G}_j$ are orthonormalized. This implies that

$$\beta_{i,n} = \mathbf{g}_i^{\mathrm{H}}\mathbf{t}, \quad i = j(s+1)+1, \ldots, n,$$

and we chose $\beta_{n+1,n} \geqslant 0$ to normalize $\mathbf{g}_{n+1}$, which implies

$$\beta_{n+1,n} = \left\| \mathbf{t} - \sum_{i=j(s+1)+1}^{n} \mathbf{g}_i \beta_{i,n} \right\|.$$

The equations (5), (7), and (10) can be combined, which gives the following relation between the vectors $\mathbf{g}_i$:

$$\mathbf{g}_{n+1}\beta_{n+1,n} = (\mathbf{A} - \mu_j \mathbf{I}) \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right) - \sum_{i=j(s+1)+1}^{n} \mathbf{g}_i \beta_{i,n}. \tag{11}$$

Rearranging this equation by sorting terms involving $\mathbf{A}$ to the left gives the following relation:

$$\mathbf{A} \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right) = \left( \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n} \right) \mu_j + \sum_{i=j(s+1)+1}^{n+1} \mathbf{g}_i \beta_{i,n}. \tag{12}$$

### *2.2. A generalized Hessenberg decomposition for the vectors* $\mathbf{g}_n$

In this section, we give the precise structure of $\mathbf{U}_n$ and $\underline{\mathbf{H}}_n$ in the generalized Hessenberg decomposition (2). To illustrate the structure of these matrices, we give $\mathbf{U}_n$ and $\underline{\mathbf{H}}_n$ for $s = 2$ and $n = 7$,

$$\mathbf{U}_7 = \begin{pmatrix} 1 & -\gamma_{1,3} & & & & & \\ & 1 & -\gamma_{2,3} & -\gamma_{2,4} & & & \\ & & 1 & -\gamma_{3,4} & -\gamma_{3,5} & & \\ & & & 1 & -\gamma_{4,5} & -\gamma_{4,6} & \\ & & & & 1 & -\gamma_{5,6} & -\gamma_{5,7} \\ & & & & & 1 & -\gamma_{6,7} \\ & & & & & & 1 \end{pmatrix}, \tag{13}$$

$$\underline{\mathbf{H}}_7 = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & -\mu_1\gamma_{1,3} & & & & \\ \beta_{2,1} & \beta_{2,2} & -\mu_1\gamma_{2,3} & -\mu_1\gamma_{2,4} & & & \\ & \beta_{3,2} & \mu_1 & -\mu_1\gamma_{3,4} & -\mu_1\gamma_{3,5} & & \\ & & \beta_{4,3} & \beta_{4,4}+\mu_1 & \beta_{4,5}-\mu_1\gamma_{4,5} & -\mu_2\gamma_{4,6} & \\ & & & \beta_{5,4} & \beta_{5,5}+\mu_1 & -\mu_2\gamma_{5,6} & -\mu_2\gamma_{5,7} \\ & & & & \beta_{6,5} & \mu_2 & -\mu_2\gamma_{6,7} \\ & & & & & \beta_{7,6} & \beta_{7,7}+\mu_2 \\ & & & & & & \beta_{8,7} \end{pmatrix}. \tag{14}$$



DOI: 10.1002/nla

The first $s$ columns of the matrices correspond to the initialization phase, where, $s+1$ orthonormal vectors in $\mathcal{G}_0$ are generated (i.e., the initial vector plus $s$ additional vectors). Subsequent blocks of $s+1$ columns corresponds to the same subspace $\mathcal{G}_j$, in this case, column 3–5 correspond to $\mathcal{G}_1$, and column 6 and 7 to $\mathcal{G}_2$.

In general, we define the vectors $\mathbf{u}_i \in \mathbb{C}^n$, $\underline{\mathbf{h}}_i \in \mathbb{C}^{n+1}$, $i = s+1, \ldots, n$, as follows:

$$\mathbf{u}_i = \begin{pmatrix} \mathbf{o}_{i-(s+1)} \\ -\gamma_{i-s,i} \\ \vdots \\ -\gamma_{i-1,i} \\ 1 \\ \mathbf{o}_{n-i} \end{pmatrix}, \qquad \underline{\mathbf{h}}_i = \begin{pmatrix} \mathbf{o}_{i-(s+1)} \\ -\gamma_{i-s,i}\mu_j \\ \vdots \\ -\gamma_{i-1,i}\mu_j \\ \mu_j \\ \mathbf{o}_{n-i+1} \end{pmatrix} + \begin{pmatrix} \mathbf{o}_{j(s+1)} \\ \beta_{j(s+1)+1,i} \\ \vdots \\ \beta_{i+1,i} \\ \mathbf{o}_{n-i} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_i \\ 0 \end{pmatrix} \mu_j + \begin{pmatrix} \mathbf{o}_{j(s+1)} \\ \beta_{j(s+1)+1,i} \\ \vdots \\ \beta_{i+1,i} \\ \mathbf{o}_{n-i} \end{pmatrix}.$$

We remark that the index $j$ in the part defined by the $\beta$s depends on the index $i$. The first $s$ vectors $\mathbf{u}_i \in \mathbb{C}^n$ and $\underline{\mathbf{h}}_i \in \mathbb{C}^{n+1}$, $i = 1, \ldots, s$, are defined to be those that contain in the first $s$ and $s+1$ elements the columns of $\mathbf{U}_s = \mathbf{I}_s$ and $\underline{\mathbf{H}}_s$, respectively, from the Hessenberg decomposition (4) resulting from Arnoldi's method. By defining the matrices

$$\mathbf{U}_n = \begin{pmatrix} \mathbf{u}_1, \cdots, \mathbf{u}_n \end{pmatrix}, \qquad \underline{\mathbf{H}}_n = \begin{pmatrix} \underline{\mathbf{h}}_1 \cdots \underline{\mathbf{h}}_n \end{pmatrix}, \tag{15}$$

Eqns (4), (9), and (12) can be compactly written as a generalized Hessenberg decomposition (2). The matrix $\mathbf{U}_n$ is an $n \times n$ upper triangular matrix with upper bandwidth $s$. $\underline{\mathbf{H}}_n$ is an $(n+1) \times n$ extended Hessenberg matrix, also with upper bandwidth $s$. The generalized Hessenberg decomposition (2) will be at the basis of the solution algorithms for linear systems that we will present in the next sections.

*Remark 1 (Arnoldi's method & computation of $\mu_j$)*
In the generalized Hessenberg decomposition that we have outlined in the preceding text, all vectors $\mathbf{g}_i \in \mathcal{G}_{j-1} \setminus \mathcal{G}_j$, $(j-1)(s+1) < i \leqslant j(s+1)$ are orthonormalized for $1 \leqslant i \leqslant n+1$, $1 \leqslant j \leqslant \lfloor (n+1)/(s+1) \rfloor + 1$. The resulting algorithm is therefore in spirit close to Arnoldi's algorithm. The two algorithms coincide for $n \leqslant s$.

After every $s+1$ steps of the algorithm, a new value for $\mu_j$ has to be selected. In the spirit of Arnoldi's algorithm, it is a natural idea to select a new $\mu_j$ to make the first $\mathbf{g}$-vector in $\mathcal{G}_j$ orthogonal to the last $\mathbf{g}$-vector in $\mathcal{G}_{j-1}$. However, from many experiments (not reported in this paper), we concluded that this choice for $\mu_j$ may lead to very slow convergence or even stagnation of the solution algorithms based on the generalized Hessenberg decomposition (2) that will be presented in the next sections. In this paper, we limit us to giving the strategy for selecting $\mu_j$, which gave us the best results. The detailed algorithm for computing this $\mu$ will be presented as Algorithm 2 in Section 6.

## 3. A SOLUTION ALGORITHM BASED ON THE GENERALIZED HESSENBERG DECOMPOSITION

In this section, we will outline a quasi-minimal residual algorithm for solving the system $\mathbf{A}\mathbf{x} = \mathbf{r}_0$. The derivation of the algorithm is almost completely analogous to that of MinRes [12], GMRes [13], and QMR [14]. The main difference is that our algorithm is based on a *generalized* Hessenberg decomposition.

The goal is to find approximate solution vectors $\underline{\mathbf{x}}_n \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ such that the norm of the corresponding residual $\mathbf{r}_0 - \mathbf{A}\underline{\mathbf{x}}_n$ is minimized:

$$\|\mathbf{r}_0 - \mathbf{A}\underline{\mathbf{x}}_n\| = \min_{\underline{\mathbf{x}} \in \mathcal{K}_n} \|\mathbf{r}_0 - \mathbf{A}\underline{\mathbf{x}}\|. \tag{16}$$



DOI: 10.1002/nla

We assume that in the $n$th iteration, we have available matrices $\mathbf{G}_n$, $\mathbf{U}_n$, $\mathbf{G}_{n+1}$, and $\underline{\mathbf{H}}_n$ that satisfy Eqn (2). Moreover, we start the process with $\mathbf{g}_1\|\mathbf{r}_0\| = \mathbf{r}_0$, so that $\mathbf{G}_{n+1}\underline{\mathbf{e}}_1\|\mathbf{r}_0\| = \mathbf{r}_0$, with $\underline{\mathbf{e}}_1$ the first canonical basis vector of length $n+1$. We construct $\underline{\mathbf{x}}_n$ as a linear combination of the columns of $\mathbf{G}_n$ by putting

$$\underline{\mathbf{x}}_n = \mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}_n = \mathbf{V}_n\underline{\mathbf{z}}_n \tag{17}$$

with unknown coefficient vector $\underline{\mathbf{z}}_n \in \mathbb{C}^n$. Here, we did define $\mathbf{V}_n = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$, with $\mathbf{v}_i = \mathbf{g}_i$, $1 \leqslant i \leqslant s$, and $\mathbf{v}_i$, $s < i \leqslant n$ defined by Eqn (5). The second equality in (17) is based on the definitions of $\mathbf{G}_n$ and $\mathbf{U}_n$ in Eqns (1) and (15), respectively.

Substituting this expression in (16) yields a minimization problem for the vector $\underline{\mathbf{z}}_n$:

$$\|\mathbf{r}_0 - \mathbf{A}\mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}_n\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \|\mathbf{r}_0 - \mathbf{A}\mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}\|.$$

Using Eqn (2) gives

$$\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n)\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}})\|. \tag{18}$$

Unfortunately, the matrix $\mathbf{G}_{n+1}$ does not have orthonormal columns, else $\underline{\mathbf{z}}_n$ would be the solution to the uniquely solvable least-squares problem

$$\left\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n\right\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \left\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}\right\|. \tag{19}$$

Because

$$\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n)\| \leqslant \|\mathbf{G}_{n+1}\| \cdot \left\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n\right\|, \tag{20}$$

we 'quasi-minimize' (18) by minimizing this upper bound. We remark that we know a priori that

$$\|\mathbf{G}_{n+1}\| \leqslant \sqrt{\lceil (n+1)/(s+1)\rceil}, \tag{21}$$

because every consecutive block of $(s+1)$ columns consists of orthonormal vectors [24, Lemma 4, p. 1058].

Clearly, the upper bound in Eqn (20) is minimized by the solution of Eqn (19). The solution of this system can be determined with the aid of the tall QR decomposition of $\underline{\mathbf{H}}_n$:

$$\underline{\mathbf{H}}_n = \underline{\mathbf{Q}}_n\mathbf{R}_n, \qquad \underline{\mathbf{Q}}_n \in \mathbb{C}^{(n+1)\times n}, \quad \mathbf{R}_n \in \mathbb{C}^{n\times n}, \tag{22}$$

in which $\underline{\mathbf{Q}}_n$ is a matrix with orthonormal columns, and $\mathbf{R}_n$ is upper triangular. Because $\underline{\mathbf{H}}_n$ only has one nonzero subdiagonal, the QR decomposition of $\underline{\mathbf{H}}_n$ is typically computed using Givens rotations. The solution to (19) is then given by

$$\underline{\mathbf{z}}_n = \mathbf{R}_n^{-1}\underline{\mathbf{Q}}_n^{\mathrm{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\| = \underline{\mathbf{H}}_n^{\dagger}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|,$$

which must be combined with (17) to give the approximate solution vector $\underline{\mathbf{x}}_n$.

Next, we will explain how the approximate solution vector

$$\underline{\mathbf{x}}_n = \mathbf{G}_n\mathbf{U}_n\mathbf{R}_n^{-1}\underline{\mathbf{Q}}_n^{\mathrm{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\| = \mathbf{V}_n\mathbf{R}_n^{-1}\underline{\mathbf{Q}}_n^{\mathrm{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\| \tag{23}$$

can be computed using short recurrences, in a way such that the storage requirements and work per iteration are constant, exactly like it is done in MinRes [12] and QMR [14]. As was remarked



DOI: 10.1002/nla

before, the matrix $\underline{\mathbf{H}}_n$ is an extended upper Hessenberg matrix with upper bandwidth $s$. From this nonzero pattern immediately follows that $\underline{\mathbf{h}}_i \perp \underline{\mathbf{h}}_j, |i-j| > s+1$. Therefore, the upper triangular matrix $\mathbf{R}_n$ has upper bandwidth $s+1$. We introduce the auxiliary matrix

$$\mathbf{W}_n = \mathbf{G}_n \mathbf{U}_n \mathbf{R}_n^{-1} = \mathbf{V}_n \mathbf{R}_n^{-1}.$$

In order to compute $\mathbf{w}_n = \mathbf{W}_n \mathbf{e}_n$, we write

$$\mathbf{W}_n \mathbf{R}_n \mathbf{e}_n = \mathbf{G}_n \mathbf{U}_n \mathbf{e}_n = \mathbf{V}_n \mathbf{e}_n = \mathbf{v}_n, \tag{24}$$

see Eqn (5). Eqn (24) can therefore be rewritten as

$$\sum_{i=n-s-1}^{n} \mathbf{w}_i \mathbf{R}_n(i,n) = \mathbf{v}_n, \tag{25}$$

in which $\mathbf{R}_n(i,n)$ is entry $(i,n)$ of the matrix $\mathbf{R}_n$. From (25), it follows that the update formula for $\mathbf{w}_n$ is

$$\mathbf{w}_n = \left( \mathbf{v}_n - \sum_{i=n-s-1}^{n-1} \mathbf{w}_i \mathbf{R}_n(i,n) \right) \cdot \frac{1}{\mathbf{R}_n(n,n)}. \tag{26}$$

Note that this is a short recurrence formula: only the $s+1$ most recent vectors $\mathbf{w}_i$, $n-(s+1) \leqslant i \leqslant n-1$ are needed to compute $\mathbf{w}_n$. Let the vector $\boldsymbol{\phi}_n$ be defined by

$$\boldsymbol{\phi}_n = \underline{\mathbf{Q}}_n^{\mathrm{H}} \underline{\mathbf{e}}_1 \|\mathbf{r}_0\|.$$

This vector grows by one entry in every iteration when a new Givens rotation is applied. The approximate solution vector is then given by

$$\underline{\mathbf{x}}_n = \mathbf{W}_n \boldsymbol{\phi}_n.$$

Because $\underline{\mathbf{x}}_{n-1} = \mathbf{W}_{n-1} \boldsymbol{\phi}_{n-1}$, the short recurrence update for $\underline{\mathbf{x}}_n$ becomes

$$\underline{\mathbf{x}}_n = \underline{\mathbf{x}}_{n-1} + \mathbf{w}_n \boldsymbol{\phi}_n(n), \tag{27}$$

in which $\boldsymbol{\phi}_n(n)$ is the $n$th coefficient of the vector $\boldsymbol{\phi}_n$.

All the elements of the solution algorithm have now been derived. In Section 6, we will put all the elements in place in the form of relatively easily implementable algorithms. However, before we present the algorithms, we will make two straightforward generalizations.

## 4. FLEXIBLE QMRIDR(s)

The first generalization is to include a variable preconditioner in the solution algorithm. The idea to change the preconditioner in a Krylov subspace method in every step dates back to 1993: flexible GMRes [26]. More recent variants include GMResR [27], flexible CG [28], flexible QMR [29], and flexible BiCG and flexible BiCGStab [30]. The latter is a flexible IDR variant, as BiCGStab is from the IDR family. In contrast to flexible BiCGStab, where the preconditioning matrix remains constant for every $2 = s+1$ steps[‡], we allow for a different preconditioning matrix in every step: let $\mathbf{P}_n$

[‡]We count the number of matrix-vector multiplications as steps to obtain a fair comparison between different iterative methods.



DOI: 10.1002/nla

be the preconditioning matrix that may be different in every iteration $n$. A generalized Hessenberg relation is derived by replacing Eqn (7) by

$$\mathbf{t} = \left(\mathbf{A}\mathbf{P}_n^{-1} - \mu_j \mathbf{I}\right) \mathbf{v}_n. \tag{28}$$

If we put $\widehat{\mathbf{v}}_i = \mathbf{P}_i^{-1}\mathbf{v}_i$, $1 \leqslant i \leqslant n$, then it is easy to see by following the steps explained in Section 2 that this leads to the Hessenberg relation (cf. [26])

$$\mathbf{A}\widehat{\mathbf{V}}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n,$$

in which the matrix $\widehat{\mathbf{V}}_n$ has the vectors $\widehat{\mathbf{v}}_i$, $i = 1, \ldots, n$, as its columns. In general, this relation no longer can be written in the form of a generalized Hessenberg *decomposition*, which is why we term it a Hessenberg *relation*. Now, we look for an approximate solution vector of the form

$$\underline{\mathbf{x}}_n = \widehat{\mathbf{V}}_n \underline{\mathbf{z}}_n. \tag{29}$$

Using $\mathbf{r}_0 = \mathbf{G}_{n+1}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|$ and $\mathbf{A}\widehat{\mathbf{V}}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n$ gives

$$\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n)\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}})\|.$$

This is exactly the same minimization problem as (18), which again we approximate by quasi-minimization. Proceeding in the same way as in (23), we compute the tall QR decomposition $\underline{\mathbf{H}}_n = \underline{\mathbf{Q}}_n\mathbf{R}_n$, and we introduce the auxiliary matrix

$$\mathbf{W}_n = \widehat{\mathbf{V}}_n\mathbf{R}_n^{-1}.$$

In order to compute $\mathbf{w}_n$, the $n$th column of $\mathbf{W}_n$, we write

$$\mathbf{W}_n\mathbf{R}_n\mathbf{e}_n = \widehat{\mathbf{V}}_n\mathbf{e}_n,$$

in which $\mathbf{e}_n$ is the $n$th canonical basis vector of dimension $n$. We notice that

$$\widehat{\mathbf{V}}_n\mathbf{e}_n = \widehat{\mathbf{v}}_n = \mathbf{P}_n^{-1}\mathbf{v}_n.$$

The update formula for $\mathbf{w}_n$ therefore becomes

$$\mathbf{w}_n = \left(\mathbf{P}_n^{-1}\mathbf{v}_n - \sum_{i=n-s-1}^{n-1} \mathbf{w}_i\mathbf{R}_n(i,n)\right) \cdot \frac{1}{\mathbf{R}_n(n,n)}.$$

Finally, the solution vector $\underline{\mathbf{x}}_n$ is computed using Eqn (27).

From this outline, it follows that the only modification needed with respect to the algorithm without preconditioning is in the computation and storage of the extra vector

$$\widehat{\mathbf{v}}_n = \mathbf{P}_n^{-1}\mathbf{v}_n.$$

An interesting observation is that if $n \leqslant s$, the columns of $\mathbf{G}_{n+1}$ form an orthonormal set, and as a result, a true minimization is performed: the method outlined in the preceding text is in that case mathematically and algorithmically equivalent with FGMRes [26]. On the other hand, when using a variable preconditioner, the vectors that are generated by performing the IDR recurrences do not satisfy the IDR theorem any more: we cannot expect the vectors $\mathbf{g}_i$, $1 \leqslant i \leqslant n+1$, to stem from a sequence of nested subspaces. We can retain part of the IDR properties by choosing a new preconditioner only every $s+1$ steps like in flexible BiCGStab, but despite this restriction, the resulting method will no longer be a Krylov subspace method in general.



DOI: 10.1002/nla

## 5. MULTI-SHIFT QMRIDR(s)

Multi-shift methods have been considered in [31, p. 230–231], see also [32]. Many multi-shift Krylov subspace methods exist. We mention for example multi-shift QMR and multi-shift TFQMR [33], multi-shift CG/BiCG and BiCGStab [34], multi-shift GMRes($k$) [35], multi-shift FOM($k$) [36], multi-shift BiCGStab($\ell$) [37], and multi-shift CGLS [38]. For more on multi-shift algorithms, we refer to [11].

The QMRIDR($s$) algorithm can be easily adapted to solve shifted systems of the form

$$(\mathbf{A} - \sigma\mathbf{I})\mathbf{x}^\sigma = \mathbf{r}_0 = \mathbf{b}.$$

We assume that, in the $n$th iteration, we have available matrices $\mathbf{G}_n$, $\mathbf{U}_n$, $\mathbf{G}_{n+1}$, and $\underline{\mathbf{H}}_n$ that satisfy (2), with $\mathbf{G}_n$ such that $\mathbf{g}_1\|\mathbf{r}_0\| = \mathbf{r}_0$. We use as the initial approximation $\mathbf{x}_0 = \mathbf{o}$. Initial approximations such that the direction of the first residuals is independent of $\sigma$ are mandatory: the condition $\mathbf{g}_1\|\mathbf{r}_0^\sigma\| = \mathbf{r}_0^\sigma$ must hold simultaneously for *all* shifted systems. Analogous to the solution algorithm for the unshifted system, we construct approximate solution vectors $\underline{\mathbf{x}}_n^\sigma$ as a linear combination of the columns of $\mathbf{G}_n$ by putting

$$\underline{\mathbf{x}}_n^\sigma = \mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}_n^\sigma = \mathbf{V}_n\underline{\mathbf{z}}_n^\sigma. \tag{30}$$

Note that we can also write this as

$$\underline{\mathbf{x}}_n^\sigma = \mathbf{G}_{n+1}\underline{\mathbf{U}}_n\underline{\mathbf{z}}_n^\sigma, \tag{31}$$

in which $\underline{\mathbf{U}}_n \in \mathbb{C}^{(n+1)\times n}$ is the matrix $\mathbf{U}_n$ with an extra zero row appended at the bottom. Using this notation, we can formulate the minimization problems for the shifted systems as

$$\|\mathbf{r}_0 - (\mathbf{A}\mathbf{G}_n\mathbf{U}_n - \sigma\mathbf{G}_{n+1}\underline{\mathbf{U}}_n)\underline{\mathbf{z}}_n^\sigma\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \|\mathbf{r}_0 - (\mathbf{A}\mathbf{G}_n\mathbf{U}_n - \sigma\mathbf{G}_{n+1}\underline{\mathbf{U}}_n)\underline{\mathbf{z}}\|.$$

Using $\mathbf{r}_0 = \mathbf{G}_{n+1}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|$ and $\mathbf{A}\mathbf{G}_n\mathbf{U}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n$ gives

$$\left\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}}_n^\sigma)\right\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \left\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}})\right\|. \tag{32}$$

In order to 'quasi-minimize' (32) we solve the least-squares problems

$$\left\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}}_n^\sigma\right\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \left\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}}\right\|.$$

The solution of these systems can be determined by computing the tall QR decompositions of all shifted Hessenberg matrices $\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n$:

$$\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n = \underline{\mathbf{Q}}_n^\sigma\mathbf{R}_n^\sigma, \qquad \underline{\mathbf{Q}}_n^\sigma \in \mathbb{C}^{(n+1)\times n}, \quad \mathbf{R}_n^\sigma \in \mathbb{C}^{n\times n}. \tag{33}$$

A short recurrence update formula for the approximate solution vectors $\underline{\mathbf{x}}_n^\sigma$ is determined in the same way as for the unshifted case. With the vectors $\boldsymbol{\phi}_n^\sigma$ defined by

$$\boldsymbol{\phi}_n^\sigma = (\underline{\mathbf{Q}}_n^\sigma)^{\mathrm{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|, \tag{34}$$

and the update vectors $\mathbf{w}_n^\sigma$ for the shifted systems given by

$$\mathbf{w}_n^\sigma = \left(\mathbf{v}_n - \sum_{i=n-s-1}^{n-1} \mathbf{w}_i^\sigma\mathbf{R}_n^\sigma(i,n)\right) \cdot \frac{1}{\mathbf{R}_n^\sigma(n,n)}, \tag{35}$$



DOI: 10.1002/nla

the approximate solution vectors $\underline{\mathbf{x}}_n^\sigma$ for the shifted systems are recursively computed by

$$\underline{\mathbf{x}}_n^\sigma = \underline{\mathbf{x}}_{n-1}^\sigma + \mathbf{w}_n^\sigma \boldsymbol{\phi}_n^\sigma(n). \tag{36}$$

We remark that the last four steps, namely Eqns. (33)–(36), are the only steps actually depending on $\sigma$. For different values of $\sigma$, each of these steps can be carried out in parallel.

## 6. ALGORITHMS

In this section, we give pseudo-code implementations of the main algorithms. The first algorithm, i.e., Algorithm 1, is a pseudo-code implementation of the flexible variant of QMRIDR($s$) using double classical Gram-Schmidt orthonormalization as orthonormalization scheme[§] and Givens rotations for the updated solution of the nested extended Hessenberg least-squares problems.

The naming of the data structures used in the algorithms corresponds to the naming of the matrices and vectors as used in the previous sections. Only the data corresponding to the previous $s$ iterations are stored. Columns of matrices are denoted by small characters, e.g., the variable $\underline{\mathbf{h}}$ as used in the algorithm contains the possibly nonzero elements of the last column of $\underline{\mathbf{H}}_n$. All data are stored in consecutive order. For example, the variable $\mathbf{G}$ as used in the algorithm contains in iteration $n$ the vectors $\mathbf{g}_{n-s+1}, \ldots, \mathbf{g}_n$. The oldest vector is stored in the first column of $\mathbf{G}$ and the latest vector in the last column. To maintain this ordering, in line 17 columns 2 to $s$ are copied to positions 1 to $s-1$. The new vector $\mathbf{g}$ is then stored in column $s$. This results in additional overhead that should be avoided by using indirect addressing. For clarity, however, we present the algorithm with consecutive ordering of the data.

Algorithm 1 requires, apart from storage for the matrices $\mathbf{A}$ and $\mathbf{P}_n$ and some scalars, storage for $3s+6$ $N$-vectors. This includes storage for $\mathbf{b}$ and the solution vector $\mathbf{x}$.

The algorithm for computing $\mu$ using the strategy explained in [25] is given as Algorithm 2. Note that with respect to the computation of a stable basis, we could take $\mu = 0$ if $|\omega| <$ `eps`, with `eps` the relative machine precision. The occurrence of $\omega = 0$ leads to breakdown in IDR($s$), a breakdown that does not take place in the basis expansion. OR methods break down because the Hessenberg matrices are singular, in which case MR methods like QMRIDR($s$) stagnate, compared with [39]. In the context of IDR methods, a stagnation due to $\mu = 0$ is *incurable*, which is clearly visible in the structure of the matrix $\underline{\mathbf{H}}_n$ once one of the $\mu$s is zero, as the first columns are all orthogonal to the later columns. We depict as example the matrix $\underline{\mathbf{H}}_7$ in Eqn (14) with $\mu_1 = 0$, explicitly denoting the new zeros thus introduced:

$$\underline{\mathbf{H}}_7 = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & 0 & & & & \\ \beta_{2,1} & \beta_{2,2} & 0 & 0 & & & \\ & \beta_{3,2} & 0 & 0 & 0 & & \\ & & \beta_{4,3} & \beta_{4,4}+\mu_1 & \beta_{4,5} & -\mu_2\gamma_{4,6} & \\ & & & \beta_{5,4} & \beta_{5,5}+\mu_1 & -\mu_2\gamma_{5,6} & -\mu_2\gamma_{5,7} \\ & & & & \beta_{6,5} & \mu_2 & -\mu_2\gamma_{6,7} \\ & & & & & \beta_{7,6} & \beta_{7,7}+\mu_2 \\ & & & & & & \beta_{8,7} \end{pmatrix}. \tag{37}$$

The MR solutions, or coefficient vectors, $\underline{\mathbf{z}}_k = \underline{\mathbf{H}}_k^\dagger \underline{\mathbf{e}}_1 \|\mathbf{r}_0\|$, and the MR approximations $\underline{\mathbf{x}}_k = \mathbf{G}_k \underline{\mathbf{z}}_k$, $k \geqslant 2$ satisfy the trivial recurrences

$$\underline{\mathbf{z}}_{k+1} = \begin{pmatrix} \underline{\mathbf{z}}_k \\ 0 \end{pmatrix}, \quad \underline{\mathbf{x}}_{k+1} = \underline{\mathbf{x}}_k, \quad k \geqslant 2. \tag{38}$$

A simple way around this incurable stagnation is to give $\mu$ some nonzero value, e.g., $\mu = 1$.

[§]There are other options, in our experience this one proved to be sufficiently stable and is the one we used in our examples.



DOI: 10.1002/nla

**Algorithm 1** FQMRIDR($s$)

INPUT: $\mathbf{A} \in \mathbb{C}^{N\times N}$; $\mathbf{x}_0, \mathbf{b} \in \mathbb{C}^N$; $s > 0$; $\widetilde{\mathbf{R}}_0 \in \mathbb{C}^{N\times s}$; $TOL \in (0,1)$;
OUTPUT: Approximate solution $\underline{\mathbf{x}}$ such that $\|\mathbf{b} - \mathbf{A}\underline{\mathbf{x}}\| \leqslant TOL \cdot \|\mathbf{b}\|$.

$\mathbf{g} = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\underline{\mathbf{x}} = \mathbf{x}_0$; $\mu = 0$; $\mathbf{M} = \mathbf{O} \in \mathbb{C}^{s\times s}$; $\mathbf{G} = \mathbf{O} \in \mathbb{C}^{N\times s}$; // *Initialization*
$\mathbf{W} = \mathbf{O} \in \mathbb{C}^{N\times(s+1)}$; $\mathbf{w} = \mathbf{o} \in \mathbb{C}^N$; $\mathbf{cs} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\mathbf{sn} = \mathbf{o} \in \mathbb{C}^{s+2}$;
$\rho = \|\mathbf{g}\|$; $\rho_0 = \rho$; $\mathbf{g} = \frac{1}{\rho}\mathbf{g}$; $\phi = 0$; $\hat{\phi} = \rho$; $n = 0$; $j = 0$;
**while** $\rho/\rho_0 > TOL$ **do**
  **for** $k = 1, s+1$ **do**
    $n = n + 1$
    $\underline{\mathbf{u}} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\underline{\mathbf{u}}_{(s+1)} = 1$; // *Initialize new column of* $\underline{\mathbf{U}}$
    $\mathbf{m} = \widetilde{\mathbf{R}}_0^{\mathrm{H}}\mathbf{g}$; // *Construct* $\mathbf{v} \perp \widetilde{\mathbf{R}}_0$
    **if** $n > s$ **then**
      Solve $\gamma$ from $\mathbf{M}\gamma = \mathbf{m}$;
      $\mathbf{v} = \mathbf{g} - \mathbf{G}\gamma$;
      $\underline{\mathbf{u}}_{(1:s)} = -\gamma$;
    **else**
      $\mathbf{v} = \mathbf{g}$;
    **end if**
    $\mathbf{M}_{(:,1:s-1)} = \mathbf{M}_{(:,2:s)}$; $\mathbf{M}_{(:,s)} = \mathbf{m}$;
    $\mathbf{G}_{(:,1:s-1)} = \mathbf{G}_{(:,2:s)}$; $\mathbf{G}_{(:,s)} = \mathbf{g}$;
    Solve $\widehat{\mathbf{v}}$ from $\mathbf{P}_n\widehat{\mathbf{v}} = \mathbf{v}$; // *Variable preconditioning*
    $\mathbf{g} = \mathbf{A}\widehat{\mathbf{v}}$;
    **if** $k = s + 1$ **then**
      $j = j + 1$; $\mu = \mathrm{COMPMU}(\mathbf{g}, \mathbf{v})$; // *New Sonneveld space*
    **end if**
    $\mathbf{g} = \mathbf{g} - \mu\mathbf{v}$;
    $\underline{\mathbf{h}} = \mu\underline{\mathbf{u}}$; // *Initialize new column of* $\underline{\mathbf{H}}$
    **if** $k < s + 1$ **then**
      $\beta = \mathbf{G}^{\mathrm{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\beta$;
      $\hat{\beta} = \mathbf{G}^{\mathrm{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\hat{\beta}$;
      $\beta = \beta + \hat{\beta}$;
      $\underline{\mathbf{h}}_{(s+1-k+1:s+1)} = \underline{\mathbf{h}}_{(s+1-k+1:s+1)} + \beta$;
    **end if**
    $\underline{\mathbf{h}}_{(s+2)} = \|\mathbf{g}\|$; $\mathbf{g} = \frac{1}{\underline{\mathbf{h}}_{(s+2)}}\mathbf{g}$;
    $\mathbf{r} = \mathbf{o} \in \mathbb{C}^{s+3}$; $\mathbf{r}_{(2:s+3)} = \underline{\mathbf{h}}$; // *Initialize new column of* $\mathbf{R}$
    $l_b = \max(1, s + 3 - n)$;
    **for** $l = l_b, s + 1$ **do**
      $t = \mathbf{r}_{(l)}$;
      $\mathbf{r}_{(l)} = \mathbf{cs}_{(l)}t + \mathbf{sn}_{(l)}\mathbf{r}_{(l+1)}$;
      $\mathbf{r}_{(l+1)} = -\overline{\mathbf{sn}}_{(l)}t + \mathbf{cs}_{(l)}\mathbf{r}_{(l+1)}$;
    **end for**
    $[\mathbf{cs}_{(s+2)}, \mathbf{sn}_{(s+2)}, \mathbf{r}_{(s+2)}] = \mathrm{ROTG}(\mathbf{r}_{(s+2)}, \mathbf{r}_{(s+3)})$;
    $\phi = \mathbf{cs}_{(s+2)}\hat{\phi}$; $\hat{\phi} = -\overline{\mathbf{sn}}_{(s+2)}\hat{\phi}$;
    $\mathbf{cs}_{(:,1:s+1)} = \mathbf{cs}_{(:,2:s+2)}$; $\mathbf{sn}_{(:,1:s+1)} = \mathbf{sn}_{(:,2:s+2)}$;
    $\mathbf{w} = (\widehat{\mathbf{v}} - \mathbf{W}\mathbf{r}_{(1:s+1)})/\mathbf{r}_{(s+2)}$;
    $\mathbf{W}_{(:,1:s)} = \mathbf{W}_{(:,2:s+1)}$; $\mathbf{W}_{(:,s+1)} = \mathbf{w}$;
    $\underline{\mathbf{x}} = \underline{\mathbf{x}} + \phi\mathbf{w}$;
    $\rho = |\hat{\phi}|\sqrt{j+1}$; // *Compute upper bound for residual norm*
  **end for**
**end while**



DOI: 10.1002/nla

**Algorithm 2** COMPMU

INPUT: $\mathbf{t}, \mathbf{v} \in \mathbb{C}^N$;
OUTPUT: $\mu$;
1: $\kappa = 0.7$; // *Value* $\kappa = 0.7$ *is recommended in* [25]
2: $\omega = (\mathbf{t}^{\mathrm{H}}\mathbf{v})/(\mathbf{t}^{\mathrm{H}}\mathbf{t})$;
3: $\rho = (\mathbf{t}^{\mathrm{H}}\mathbf{v})/(\|\mathbf{t}\|\|\mathbf{v}\|)$;
4: **if** $|\rho| < \kappa$ **then**
5: $\quad \omega = \omega\kappa/|\rho|$;
6: **end if**
7: **if** $|\omega| >$ `eps` **then**
8: $\quad \mu = 1/\omega$;
9: **else**
10: $\quad \mu = 1$;
11: **end if**

The Givens rotations are computed with the BLAS algorithm ROTG, which is given as Algorithm 3. This algorithm computes $cs$, $sn$, and $r$ such that

$$\begin{pmatrix} cs & sn \\ -\overline{sn} & cs \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}.$$

**Algorithm 3** ROTG

INPUT: $a, b \in \mathbb{C}$;
OUTPUT: $cs, sn, r$;
1: **if** $|a| <$ `eps` **then**
2: $\quad cs = 0$; $sn = 1$; $r = b$;
3: **else**
4: $\quad t = |a| + |b|$; $\rho = t\sqrt{|a/t|^2 + |b/t|^2}$;
5: $\quad \alpha = a/|a|$;
6: $\quad cs = |a|/\rho$; $sn = \alpha\overline{b}/\rho$; $r = \alpha\rho$;
7: **end if**

The multi-shift algorithm MSQMRIDR($s$) is presented as Algorithm 4. The algorithm is very similar to FQMRIDR($s$), with as most important differences that no preconditioner is used in MSQMRIDR($s$), and the computation of the update vectors for every shifted system is carried out in a loop (line 34 to line 45) over the number of shifts. The algorithm as presented continues executing these loops until the upper bound on the residual norms of all the shifted systems is below a given tolerance. In practice, these loops only have to be executed for the shifted systems of which the residual norms are above that tolerance, thus saving some unnecessary operations. For ease of presentation, we did not include it in algorithm MSQMRIDR($s$), but we did implement this in the algorithm with which we did our numerical experiments.

Algorithm 4 requires, apart from storage for the matrix $\mathbf{A}$ and for scalars, storage for $2s + 4 + n_\sigma(s + 2)$ $N$-vectors, with $n_\sigma$ the number of shifts. This includes storage for $\mathbf{b}$ and the solutions $\mathbf{x}^{\sigma_i}$ of the $n_\sigma$ shifted systems. For each extra shift, there are therefore $s + 2$ extra $N$-vectors of storage required.

For comparison, we tabulate in Table I the additional $N$-vectors per shift of a number of standard multi-shift algorithms. The information in Table I is taken from [34]. The additional storage requirements for MSQMRIDR(1) are the same as for the shifted version of QMR. For large $s$, however, the additional storage requirements may become prohibitive if many shifted systems have to be solved. In this case, one may be forced to choose a small value for $s$.



DOI: 10.1002/nla

**Algorithm 4** MSQMRIDR($s$)

INPUT: $\mathbf{A} \in \mathbb{C}^{N\times N}$; $\mathbf{b} \in \mathbb{C}^N$; $\sigma_i \in \mathbb{C}$, $i = 1, \ldots, n_\sigma$; $s > 0$; $\widetilde{\mathbf{R}}_0 \in \mathbb{C}^{N\times s}$; $TOL \in (0, 1)$;
OUTPUT: Approximate solutions $\underline{\mathbf{x}}^{\sigma_i}$ such that $\|\mathbf{b} - (\mathbf{A} - \sigma_i\mathbf{I})\underline{\mathbf{x}}^{\sigma_i}\| \leqslant TOL \cdot \|\mathbf{b}\|$, $i = 1, \ldots, n_\sigma$.

$\mathbf{g} = \mathbf{b}$; $\rho = \|\mathbf{g}\|$, $\rho_0 = \rho$; $\mathbf{g} = \frac{1}{\rho}\mathbf{g}$; $\underline{\mathbf{x}}^{\sigma_i} = \mathbf{o}$, $i = 1, \ldots, n_\sigma$; *// Initialization*
$\mu = 0$; $\mathbf{M} = \mathbf{O} \in \mathbb{C}^{s\times s}$; $\mathbf{G} = \mathbf{O} \in \mathbb{C}^{N\times s}$; $\mathbf{w} = \mathbf{o} \in \mathbb{C}^N$; $\mathbf{v} = \mathbf{o} \in \mathbb{C}^N$;
**for** $i = 1, n_\sigma$ **do**
  $\mathbf{W}^{\sigma_i} = \mathbf{O} \in \mathbb{C}^{N\times(s+1)}$;
  $\mathbf{cs}^{\sigma_i} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\mathbf{sn}^{\sigma_i} = \mathbf{o} \in \mathbb{C}^{s+2}$;
  $\phi^{\sigma_i} = 0$; $\hat{\phi}^{\sigma_i} = \rho$;
**end for**
$n = 0$; $j = 0$;
**while** $\rho/\rho_0 > TOL$ **do**
  **for** $k = 1, s + 1$ **do**
    $n = n + 1$
    $\underline{\mathbf{u}} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\underline{\mathbf{u}}_{(s+1)} = 1$; *// Initialize new column of* $\underline{\mathbf{U}}$
    $\mathbf{m} = \widetilde{\mathbf{R}}_0^{\mathrm{H}}\mathbf{g}$; *// Construct* $\mathbf{v} \perp \widetilde{\mathbf{R}}_0$
    **if** $n > s$ **then**
      Solve $\gamma$ from $\mathbf{M}\gamma = \mathbf{m}$;
      $\mathbf{v} = \mathbf{g} - \mathbf{G}\gamma$;
      $\underline{\mathbf{u}}_{(1:s)} = -\gamma$;
    **else**
      $\mathbf{v} = \mathbf{g}$;
    **end if**
    $\mathbf{M}_{(:,1:s-1)} = \mathbf{M}_{(:,2:s)}$; $\mathbf{M}_{(:,s)} = \mathbf{m}$; $\mathbf{G}_{(:,1:s-1)} = \mathbf{G}_{(:,2:s)}$; $\mathbf{G}_{(:,s)} = \mathbf{g}$; $\mathbf{g} = \mathbf{A}\mathbf{v}$;
    **if** $k = s + 1$ **then**
      $j = j + 1$; $\mu = \mathrm{COMPMU}(\mathbf{g}, \mathbf{v})$; *// New Sonneveld space*
    **end if**
    $\mathbf{g} = \mathbf{g} - \mu\mathbf{v}$;
    $\underline{\mathbf{h}} = \mu\underline{\mathbf{u}}$; *// Initialize new column of* $\underline{\mathbf{H}}$
    **if** $k < s + 1$ **then**
      $\beta = \mathbf{G}^{\mathrm{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\beta$;
      $\hat{\beta} = \mathbf{G}^{\mathrm{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\hat{\beta}$;
      $\beta = \beta + \hat{\beta}$; $\underline{\mathbf{h}}_{(s+1-k+1:s+1)} = \underline{\mathbf{h}}_{(s+1-k+1:s+1)} + \beta$;
    **end if**
    $\underline{\mathbf{h}}_{(s+2)} = \|\mathbf{g}\|$; $\mathbf{g} = \frac{1}{\underline{\mathbf{h}}_{(s+2)}}\mathbf{g}$;
    $\rho = 0$;
    **for** $i = 1, n_\sigma$ **do**
      $\mathbf{r} = \mathbf{o} \in \mathbb{C}^{s+3}$; $\mathbf{r}_{(2:s+3)} = \underline{\mathbf{h}} - \sigma_i\underline{\mathbf{u}}$; *// Initialize new column of* $\mathbf{R}^{\sigma_i}$
      $l_b = \max(1, s + 3 - n)$;
      **for** $l = l_b, s + 1$ **do**
        $t = \mathbf{r}_{(l)}$; $\mathbf{r}_{(l)} = \mathbf{cs}^{\sigma_i}_{(l)}t + \mathbf{sn}^{\sigma_i}_{(l)}\mathbf{r}_{(l+1)}$; $\mathbf{r}_{(l+1)} = -\overline{\mathbf{sn}}^{\sigma_i}_{(l)}t + \mathbf{cs}^{\sigma_i}_{(l)}\mathbf{r}_{(l+1)}$;
      **end for**
      $[\mathbf{cs}^{\sigma_i}_{(s+2)}, \mathbf{sn}^{\sigma_i}_{(s+2)}, \mathbf{r}_{(s+2)}] = \mathrm{ROTG}(\mathbf{r}_{(s+2)}, \mathbf{r}_{(s+3)})$;
      $\phi^{\sigma_i} = \mathbf{cs}^{\sigma_i}_{(s+2)}\hat{\phi}^{\sigma_i}$; $\hat{\phi}^{\sigma_i} = -\overline{\mathbf{sn}}^{\sigma_i}_{(s+2)}\hat{\phi}^{\sigma_i}$; $\mathbf{cs}^{\sigma_i}_{(:,1:s+1)} = \mathbf{cs}^{\sigma_i}_{(:,2:s+2)}$; $\mathbf{sn}^{\sigma_i}_{(:,1:s+1)} = \mathbf{sn}^{\sigma_i}_{(:,2:s+2)}$;
      $\mathbf{w} = (\mathbf{v} - \mathbf{W}^{\sigma_i}\mathbf{r}_{(1:s+1)})/\mathbf{r}_{(s+2)}$; $\mathbf{W}^{\sigma_i}_{(:,1:s)} = \mathbf{W}^{\sigma_i}_{(:,2:s+1)}$; $\mathbf{W}^{\sigma_i}_{(:,s+1)} = \mathbf{w}$;
      $\underline{\mathbf{x}}^{\sigma_i} = \underline{\mathbf{x}}^{\sigma_i} + \phi^{\sigma_i}\mathbf{w}$;
      $\rho = \max(\rho, |\hat{\phi}^{\sigma_i}|\sqrt{j + 1})$; *// Compute upper bound for maximum residual norm*
    **end for**
  **end for**
**end while**



DOI: 10.1002/nla

Table I. Additional $N$-vectors per shift.

| Method | Extra $N$-vectors per shift |
|---|---|
| QMR | 3 |
| TFQMR | 5 |
| BiCG | 2 |
| BiCGStab | 2 |
| MSQMRIDR($s$) | $s+2$ |

## 7. NUMERICAL EXPERIMENTS

The experiments that are presented in this section have been performed on a standard desktop computer running under Linux with kernel 3.12.9, with four Intel® Xeon® X5667 CPUs and 12GB of RAM using Matlab 8.1.

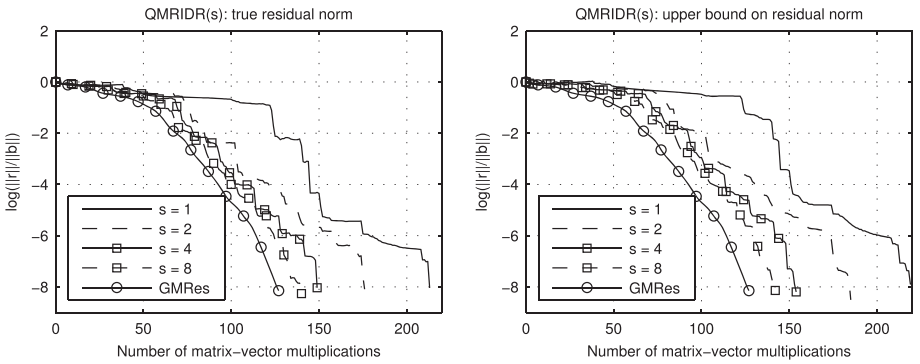
In all our experiments, we take for $\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_s$ the orthonormalization of $s$ normally distributed random vectors, with mean 0 and standard deviation 1, i.e., stated in form of a Matlab command: $\widetilde{\mathbf{R}}_0 =$ `orth(randn(N,s))`.

### *7.1. Example 1: SHERMAN 4*

In the first experiment, we investigate how sharp the upper bound (21) on the residual norm is, and compare the convergence of QMRIDR($s$) with the convergence of IDR($s$), and with the optimal convergence of full GMRes. To this end, we have chosen the matrix SHERMAN 4 with corresponding right-hand side from the MATRIX MARKET [40]. $N = 1104$ for this problem. This classic test problem is reasonably well conditioned, and as a result, it suffices to use a small value for $s$. In the computations, we have used $s = 1$, 2, 4, and 8. The required tolerance is $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| < 10^{-8}$.

For every iteration, Figure 1(a) gives the true residual norms for the four variants of QMRIDR($s$) and Figure 1(b) the upper bound on the residual norm. Also given in these figures is the convergence curve for full GMRes, which shows how close to optimal the QMRIDR convergence curves are (increasingly closer for larger $s$), even for small values of $s$. The upper bound on the residual norms is quite useful for this example: a termination criterion based on the cheap upper bound requires only a small number of additional iterations compared with a termination criterion based on the true residual norm. We have observed this for many other test problems as well. We mention that, as suggested in [14], a good strategy is to use the upper bound in most of the iterations and the true residual norm only in the final iterations.

Figure 2(a) shows for comparison the convergence of IDR($s$) (and of full GMRes) for the same test problem. Clearly, the convergence of QMRIDR($s$) is much smoother. We remark, however, that

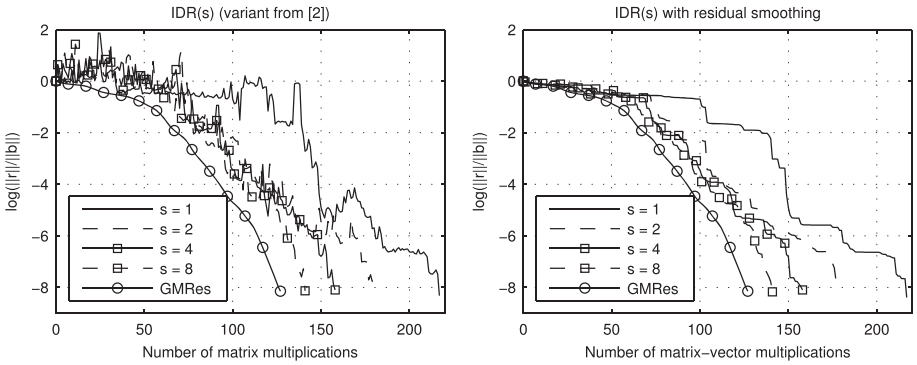



(a) Convergence of QMRIDR($s$), true residual norm. (b) Convergence of QMRIDR($s$), upper bound.

Figure 1. Example 1: Convergence of QMRIDR(s): true residual norms (left) and upper bound (right).



DOI: 10.1002/nla




(a) Convergence of IDR($s$). (b) Convergence of IDR($s$) with smoothing.

Figure 2. Example 1: Convergence of IDR(s) without (left) and with (right) residual smoothing.

Table II. Example 1: Number of iterations for the different (QMR)IDR variants.

| $s$ | QMRIDR($s$) True residual norm | QMRIDR($s$) Upper bound | IDR($s$) No smoothing | IDR($s$) Smoothing |
|---|---|---|---|---|
| 1 | 213 | 223 | 217 | 217 |
| 2 | 176 | 185 | 181 | 181 |
| 4 | 149 | 154 | 158 | 158 |
| 8 | 140 | 142 | 141 | 141 |

the rate of convergence of the two methods is essentially the same. We consider the smoother convergence of QMRIDR($s$) a nice, but not very important feature of the method. Smooth convergence of the residual norms can easily be achieved in IDR($s$) as well by using a residual smoothing technique. The Matlab code that is described in [2] incorporates as an option residual smoothing using the technique developed by Hestenes and Stiefel [15, §7, p. 418–419], see also Schönauer and Weiß [41, 42]. Figure 2(b) shows the resulting monotonic convergence. Also here we remark that the rate of convergence of IDR($s$) with and without smoothing is essentially the same. For completeness, we give the numbers of iterations for the different methods in Table II. The IDR($s$) variant in [2] is both simpler and cheaper than QMRIDR($s$). We therefore consider IDR($s$) the better suited method for problems that can be solved with a moderate value of $s$.

### *7.2. Example 2: SHERMAN 2*

Next, we consider the matrix SHERMAN 2 with corresponding right-hand side, here $N = 1180$. This problem is very ill conditioned, and as a result, IDR($s$) requires a high choice for $s$ to converge [2]. The purpose of this example is to illustrate that for high values of $s$, the more stable computation of the basis vectors in QMRIDR($s$) may result in considerably faster convergence than for IDR($s$).

We solve this problem with QMRIDR($s$) and with IDR($s$), with values of $s$ ranging from 20 to 140. The required tolerance for this example is $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| < 10^{-4}$. The upper bound on the residual norm is used in the termination criterion. Table III gives the number of iterations to reach the required accuracy for QMRIDR($s$) and IDR($s$) for the different values of $s$. As can be seen from the table, IDR($s$) requires considerably more iterations than QMRIDR($s$) for all choices of $s$, except for $s = 140$. This can be explained by the fact that QMRIDR($s$) always keeps the last $(s+1)$-block of $\mathbf{g}$-vectors orthonormal, which for high values of $s$ yields a quasi-minimization that is close to the real minimization of the residual norm. If the number of iterations is less than $s$, the quasi-minimization becomes a true minimization, since then all the $\mathbf{g}$-vectors are orthonormal. In that case, QMRIDR($s$) and GMRes are mathematically equivalent. This is illustrated in Figure 3, which shows the convergence for QMRIDR($s$) and IDR($s$) for $s = 140$, and of full GMRes. Note



DOI: 10.1002/nla

Table III. Example 2: Number of iterations for increasing $s$.

| Method | No. of iterations | Method | No. of iterations |
|---|---|---|---|
| QMRIDR(20) | 1099 | IDR(20) | 1908 |
| QMRIDR(40) | 445 | IDR(40) | 938 |
| QMRIDR(60) | 277 | IDR(60) | 851 |
| QMRIDR(80) | 144 | IDR(80) | 828 |
| QMRIDR(100) | 131 | IDR(100) | 789 |
| QMRIDR(120) | 119 | IDR(120) | 455 |
| QMRIDR(140) | 119 | IDR(140) | 163 |

that the required number of iterations for GMRes is 119, which is smaller than $s$. Because also the upper bound on the QMRIDR($s$) residual norm is exact as long as the number of iterations is smaller than $s$, the convergence curves for GMRes and QMRIDR($s$) coincide. The SHERMAN 2 example is in our experience exceptional in the sense that the convergence of IDR($s$) for small values of $s$ is far worse than the convergence of GMRes. For such problems that require large values for $s$, QMRIDR($s$) gives a real computational advantage over IDR($s$).

### *7.3. A convection–diffusion–reaction problem*

The third example is a finite difference discretization of a convection–diffusion–reaction problem. We will use this example to illustrate the numerical behavior of QMRIDR($s$), first as a flexible method, and then as a multi-shift method. We start with the definition of the test problem; Section 7.3.2 gives the experiment where QMRIDR($s$) is combined with a varying preconditioner, and Section 7.3.3 describes the experiment where QMRIDR($s$) is used to simultaneously solve several shifted systems.

*7.3.1. Description of the test problem.* The test problem is an academic example taken from [2]. The system that we use in the experiments is the finite difference discretization of the following convection–diffusion–reaction equation with homogeneous Dirichlet boundary conditions on the unit cube:

$$-\epsilon \Delta u + \vec{\beta} \cdot \nabla u - ru = F.$$

The right-hand side vector $F$ is defined by the solution $u(x, y, z) = x(1-x)y(1-y)z(1-z)$ of the unshifted system ($r = 0$). The problem is discretized using central differences with grid size $h = 0.025$, which gives $N \approx 60,000$. We take the following values for the parameters: $\epsilon = 1$ (diffusion), $\vec{\beta} = (0/\sqrt{5} \quad 250/\sqrt{5} \quad 500/\sqrt{5})^{\mathrm{T}}$ (convection). The value for $r$ (reaction) depends on

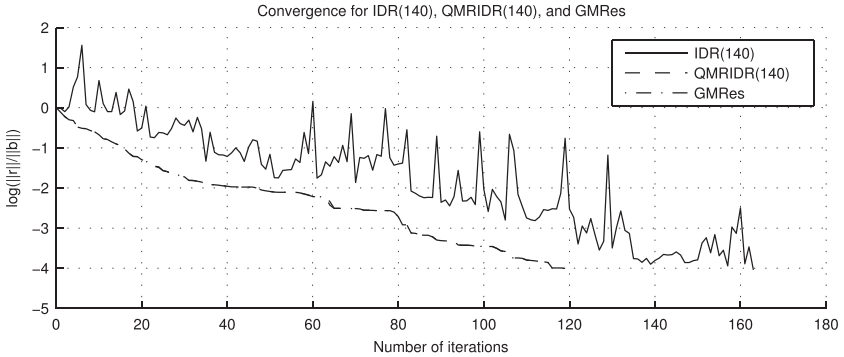



Figure 3. Example 2: Convergence of QMRIDR(140), IDR(140) and of full GMRes.



DOI: 10.1002/nla

the experiment. The resulting matrices are highly nonsymmetric and for larger $r$ indefinite, properties that make the resulting systems difficult to solve with an iterative solver. In all experiments, we use the upper bound on the residual norm. After the iterative process has ended, the norm of the true residual is computed to verify that the required accuracy is achieved. The required tolerance is $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| < 10^{-8}$.

*7.3.2. Flexible QMRIDR* ($s$). In the first experiment, we take $r = 0$. At every iteration, we use 20 steps of full GMRes as preconditioner. For every iteration, Figure 4 gives the upper bound on the residual norms of FQMRIDR($s$) for four values of $s$. Also included in this figure is the convergence curve for (full) FGMRes. For this experiment, the computing times are almost completely determined by the GMRes preconditioning iterations. As a result, the required number of iterations is a good measure for the relative performance of the different methods.

The first observation is that all four FQMRIDR variants achieve the required accuracy: the algorithm is in this respect robust for variations in the preconditioner. Second, the rates of convergence for $s = 1$, $s = 2$, and $s = 4$ are almost the same, with a trend towards a *lower* rate of convergence for higher $s$. This tendency is stronger if fewer inner GMRes iterations are performed. The explanation is that when a variable preconditioner is used, the $\mathbf{g}$-vectors are no longer in a sequence of nested subspaces, because the IDR theorem no longer holds. If fewer inner GMRes iterations are performed, the preconditioner becomes more variable. Third, FQMRIDR(8) performs considerably better than the other FQMRIDR variants. This is caused by the fact that in the first $s$ iterations, the convergence curve of FQMRIDR(8) coincides with the (optimal) convergence curve of FGMRes. The convergence of FQMRIDR(16) (not shown) completely coincides with the convergence of FGMRes, because the required number of iterations of FGMRes and of FQMRIDR(16) is 12, which is smaller than $s = 16$.

In case of strongly variable preconditioners, it is in our experience best to choose $s$ small, i.e., $s = 1$. Alternatively, because FQMRIDR and FGMRes are mathematically equivalent in the first $s$ iterations, one can take $s$ sufficiently large to (almost) obtain the convergence of full FGMRes. But this strategy is only of theoretical interest: in that case, it may be better to simply use FGMRes.

*7.3.3. Multi-shift* QMRIDR ($s$). In the following experiments, we take six different shifts: $r = 0$, $r = 200$, $r = 400$, $r = 600$, $r = 800$, and $r = 1000$ and study the simultaneous solution of the six shifted systems. For larger shifts even GMRes exhibits a long phase where no convergence takes place, similarly for QMRIDR($s$). Figure 5 shows the convergence curves for MSQMRIDR($s$) for each of the shifted systems in four different subplots. Each subplot gives the results for a specific value of $s$.

Note that in MSQMRIDR($s$) the (upper bound) on the residual norms is available for every shifted system at no extra cost.

Although MSQMRIDR($s$) requires the computation of only one set of $\mathbf{g}$-vectors, which implies a big saving in matrix-vector multiplications, the saving in vector and scalar operations is less because

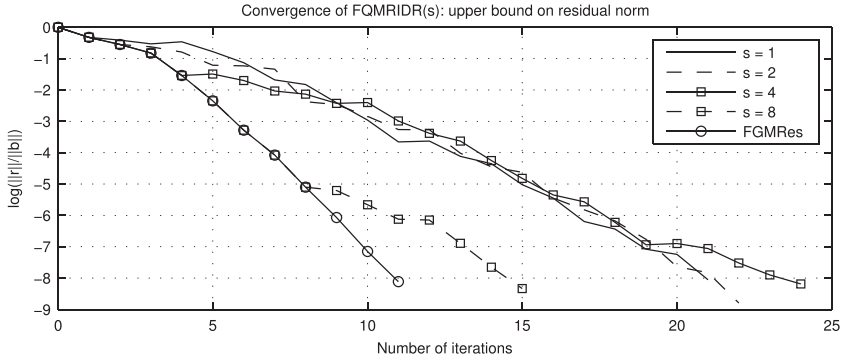
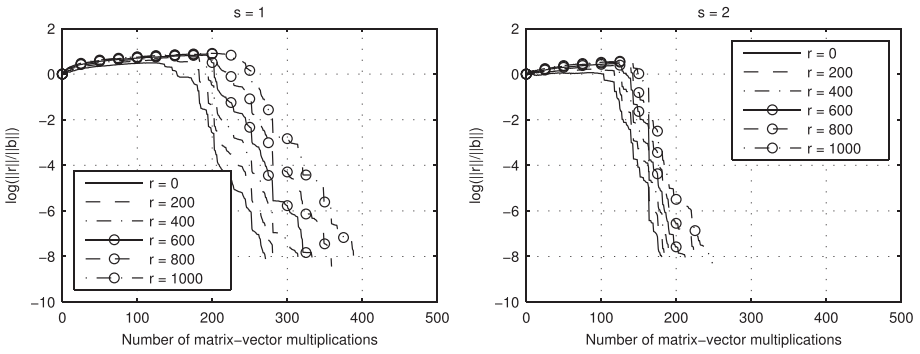


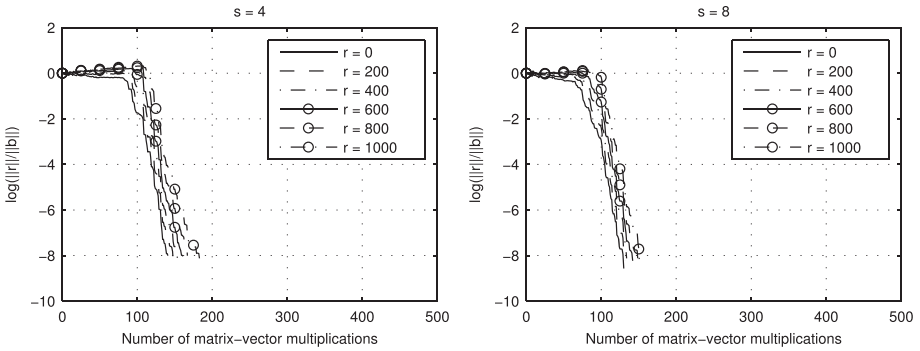

Figure 4. Example 3: Convergence of FQMRIDR(s) with a variable preconditioner.



DOI: 10.1002/nla




(a) MSQMRIDR(1). (b) MSQMRIDR(2).




(c) MSQMRIDR(4). (d) MSQMRIDR(8).

Figure 5. Example 3: Convergence for the simultaneous solution of six shifted systems.

Table IV. Example 3: Number of iterations, CPU time, and storage for the solution of the six shifted systems.

| | Simultaneous solution | | | One system at a time | | |
|---|---|---|---|---|---|---|
| $s$ | Iterations | CPU time | $N$-vectors | Iterations | CPU time | $N$-vectors |
| 1 | 389 | 5.05s | 24 | 1948 | 12.6s | 9 |
| 2 | 248 | 4.52s | 32 | 1241 | 9.6s | 12 |
| 4 | 183 | 5.13s | 48 | 954 | 11.3s | 18 |
| 8 | 151 | 6.27s | 80 | 838 | 15.9s | 30 |
| MSGMRes | 130 | 6.54s | 138 | 720 | 31.4s | 132 |

many of these operations are related to solving the (projected) shifted system. So the question arises how much more efficient the multi-shift algorithm is compared with the solution of the shifted systems one after the other. The answer to this question is very problem and implementation dependent, but to give at least the answer for the given problem and implementation, we have tabulated in Table IV the number of iterations and CPU time for the multi-shift algorithm, and the accumulated numbers of iterations and CPU time if the systems are solved subsequently. For comparison, we also give the results for multi-shift GMRes (MSGMRes). The results in Table IV show that the saving in CPU time of MSQMRIDR($s$) over the subsequent solution of the shifted systems with QMRIDR($s$) for a single shift is significant. QMRIDR($s$) is slightly faster than MSGMRes for all values of $s$ tested. QMRIDR requires much less memory space for $N$-vectors than full MSGMRes. We remark that the large memory consumption of full MSGMRes can be overcome by using the restarted MSGMRes method proposed in [35]. The drawback of restarting is that the convergence is slower than for the unrestarted method.



DOI: 10.1002/nla

### *7.4. A Helmholtz problem*

The final example is the finite element discretization of a Helmholtz equation. The test problem models wave propagation in the earth crust. We will illustrate the performance of FQMRIDR($s$) in combination with a variable multigrid preconditioner. Then, we apply MSQMRIDR($s$) to simultaneously solve the Helmholtz equation at different frequencies.

*7.4.1. Description of the test problem.* The test problem that we consider mimics three layers with a simple heterogeneity. The problem is modeled by the following equations:

$$-\Delta p - \left(\frac{2\pi f}{c(\mathbf{x})}\right)^2 p = s \qquad \text{in} \quad \Omega = (0,600)\times(0,1000), \tag{39}$$

$$s = \delta(x_1 - 300, x_2) \qquad \text{for} \quad x_1 = (0,600),\ x_2 = (0,1000), \tag{40}$$

$$\frac{\partial p}{\partial n} = 0 \qquad \text{on} \quad \partial\Omega. \tag{41}$$

The sound source $s$ is located at the surface and transmits a sound wave with frequency $f$. Homogeneous Neumann conditions are imposed at the boundaries. The local sound velocity is given as in Figure 6.

The problem is discretized with linear finite elements which leads to a system of the form

$$(\mathbf{K} - z_1\mathbf{M})\mathbf{p} = \mathbf{b}, \quad z_1 = (2\pi f)^2,$$

in which $\mathbf{K}$ is the discretized Laplacian, and the mass matrix $\mathbf{M}$ is the discretization of $\frac{1}{c(\mathbf{x})}$. We use a lumped mass matrix, which means that $\mathbf{M}$ is diagonal. The grid size is $h = 12.5m$, which yields $N \approx 3700$. The system is symmetric and indefinite. In all experiments, we use the upper bound on the residual norm. After the iterative process has ended, the norm of the true residual is

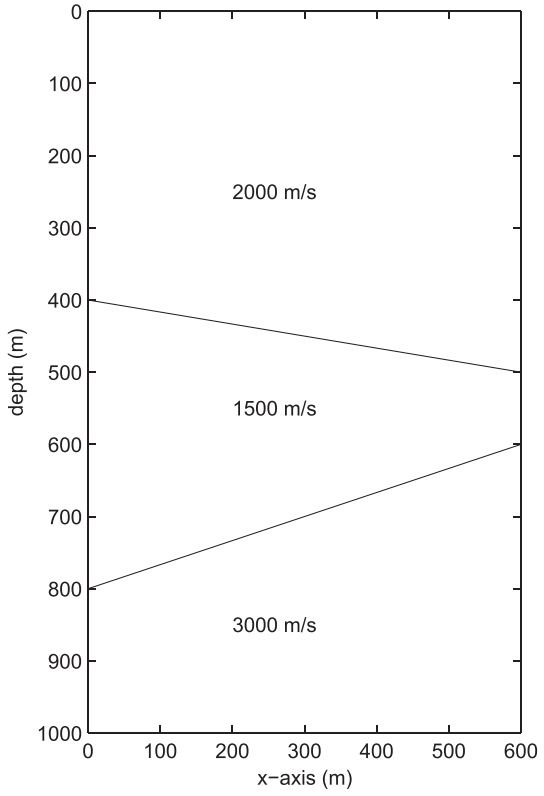



Figure 6. Example 4: Problem geometry with sound velocity profile.



DOI: 10.1002/nla

computed to verify that the required accuracy is achieved. The tolerance used in the experiments is $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| < 10^{-8}$.

*7.4.2. Flexible QMRIDR* ($s$). Shifted Laplace preconditioners for the discrete Helmholtz equation are of the form

$$\mathbf{P} = \mathbf{K} - z_2\mathbf{M}.$$

The shift $z_2$ is chosen on the one hand to ensure a fast rate of convergence and on the other hand such that the systems with $\mathbf{P}$ are easily solvable. In practice, $z_2$ is chosen such that the action of $\mathbf{P}^{-1}$ is well approximated by one multigrid cycle. In the experiments, we use $z_2 = -iz_1$ [43, 44]. The action of $\mathbf{P}^{-1}$ is approximated by one cycle of AGMG, the AGregation based algebraic MultiGrid method proposed by Notay [45]. AGMG uses a Krylov subspace method as smoother at every level. As a result, this multigrid operator is variable (changes in every iteration), and a flexible method must be used if AGMG is used as a preconditioner.

In this experiment, we take $f = 8$. The convergence of FQMRIDR($s$) with one cycle of AGMG as preconditioner is displayed in Figure 7. For every iteration, this figure gives the upper bound on the residual norms of FQMRIDR($s$) for eight values of $s$. Also included in this figure is the convergence curve for (full) FGMRes.

For this problem, it pays off to choose $s$ larger. This is in contrast to the previous example, which gave only a faster rate of convergence for $s$ in the order of the number of FGMRes iterations to solve the system. This difference stems from the fact that the AGMG preconditioner is less variable than the GMRes inner iterations that were used as preconditioner in Section 7.3.2. For $s = 128$, the convergence curves of FGMRes and FQMRIDR($s$) coincide almost completely, because the two methods are mathematically equivalent in the first $s$ iterations. The computing time for FQMRIDR(16) is about the same as for full FGMRes, both take about 2s, but FQMRIDR(16) uses of course much less memory than FGMRes.

*7.4.3. Multi-shift QMRIDR* ($s$). In the next experiments, we use MSQMRIDR($s$) to simultaneously solve four shifted systems $(\mathbf{M}^{-1}\mathbf{K} - z_1\mathbf{I})\mathbf{p} = \mathbf{M}^{-1}\mathbf{b}$, for the frequencies $f = 1$, $f = 2$, $f = 4$, and $f = 8$. Figure 8 shows the convergence curves for MSQMRIDR(1), MSQMRIDR(2), MSQMRIDR(4), and MSQMRIDR(8) for every shifted system in four different subplots. In contrast to Example 3, this example requires a larger value of $s$ to solve the systems: only for $s = 8$, all systems are solved to the required accuracy within 2000 iterations. The convergence curves for the different frequencies are more spread out than for the previous example, but the spread in the shifts for this example is also much bigger than for the previous example. Multi-shift GMRes takes 1113 iterations for this example, which is slightly less than the 1295 iterations that QMRIDR(8) takes to converge.

These high numbers of iterations show that this problem should really be solved with a *preconditioned* iterative method. But in order to combine a multi-shift method with preconditioning, it

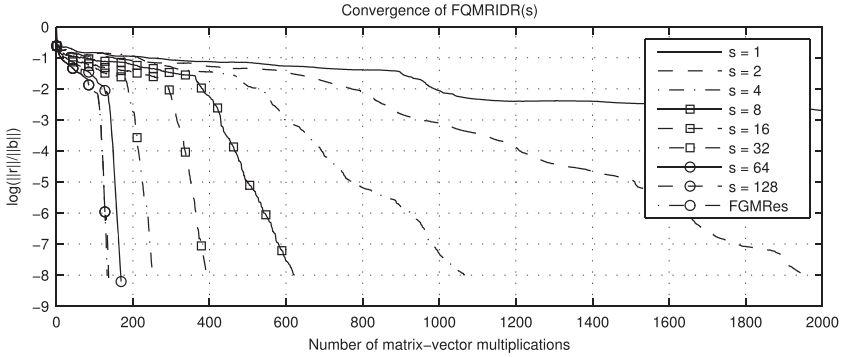
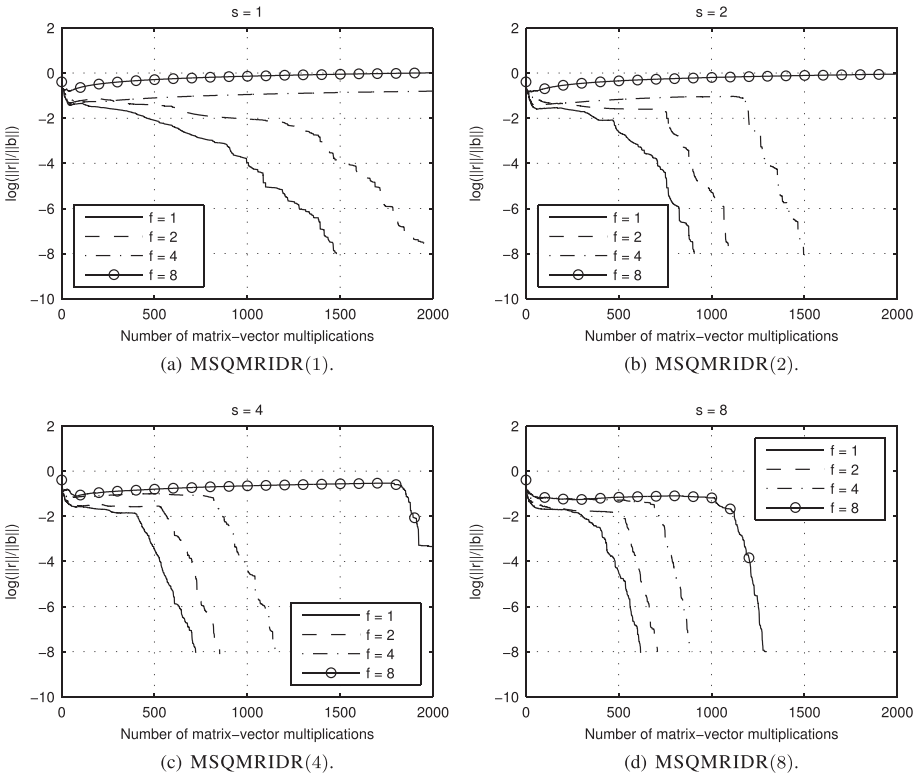



Figure 7. Example 4: Convergence of FQMRIDR($s$) with a variable preconditioner.



DOI: 10.1002/nla




(a) MSQMRIDR(1). (b) MSQMRIDR(2).

(c) MSQMRIDR(4). (d) MSQMRIDR(8).

Figure 8. Example 4: Convergence for the simultaneous solution of four shifted systems.

should be possible to formulate the preconditioned problem as a multi-shift problem. Because of this requirement, multi-shift methods cannot be combined with e.g., ILU-type or AMG-type preconditioners. This makes multi-shift problems less suitable for ill-conditioned problems as the one discussed previously. How to efficiently precondition the multi-shift problem is to a large extent an open question, see e.g. the discussion in [34, Section 4]. With a good preconditioner, one could solve one system at the time.

## 8. CONCLUDING REMARKS

We have presented two new members from the family of IDR methods highlighting the role of the underlying generalized Hessenberg decomposition. The derivation of the flexible and the multi-shift QMRIDR algorithms should allow others to easily incorporate ideas from the context of Krylov subspace methods to the slightly more complex case of IDR methods. The numerical experiments clearly reveal that IDR is a scheme that allows to narrow the gap between optimal long-term recurrences like GMRes and Lanczos-based methods without the need for complicated truncation or restart schemes. We repeat here the remark[¶], given in the abstract of [46]:

> '...can be viewed as a bridge connecting the Arnoldi-based FOM/GMRes methods and the Lanczos-based BiCGStab methods.'

With QMRIDR, we have the IDR method that gives the perfect bridge to GMRes and FGMRes: for $s$ large enough, mathematical equivalence is reached.

[¶]This remark is about ML($k$)BiCGStab, a method that is mathematically closely related to IDR($s$).



DOI: 10.1002/nla

ACKNOWLEDGEMENTS

The authors would like to thank Olaf Rendel for pointing out the proof for the upper bound in Eqn (21) and the occurrence of incurable breakdowns in case of $\mu = 0$. We acknowledge the referee for the many detailed and insightful comments that helped us to improve our manuscript.

REFERENCES

1. Sonneveld P, van Gijzen MB. IDR($s$): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing* 2008/09; **31**(2):1035–1062.
2. Van Gijzen MB, Sonneveld P. Algorithm 913: an elegant IDR($s$) variant that efficiently exploits biorthogonality properties. *ACM Transactions on Mathematical Software* December 2011; **38**(1):5:1–5:19.
3. Gutknecht MH. IDR explained. *Electronic Transactions on Numerical Analysis* 2009/10; **36**:126–148.
4. Onoue Y, Nakashima N, Fujino S. A difference between easy and profound preconditionings of IDR($s$) method. *Transactions of the Japan Society for Computational Engineering and Science* 2008; **20080023**. (in Japanese).
5. Onoue Y, Nakashima N, Fujino S. An overview of a family of new iterative methods based on IDR theorem and its estimation. *Proceedings of the international multi-conference of engineers and computer scientists 2009*, Vol. II, IMECS 2009, Hong Kong, 2009 March 18; 129–2134.
6. Gutknecht MH, Zemke JPM. Eigenvalue computations based on IDR. *SIAM Journal on Matrix Analysis and Applications* 2013; **34**(2):283–311.
7. Sleijpen GL, Sonneveld P, van Gijzen MB. Bi-CGSTAB as an induced dimension reduction method. *Applied Numerical Mathematics* 2010; **60**(11):1100–1114.
8. Simoncini V, Szyld DB. Interpreting IDR as a Petrov-Galerkin method. *SIAM Journal on Scientific Computing* 2010; **32**(4):1898–1912.
9. Sleijpen GL, van Gijzen MB. Exploiting BiCGstab($\ell$) strategies to induce dimension reduction. *SIAM Journal on Scientific Computing* 2010; **32**(5):2687–2709.
10. Sonneveld P. On the convergence behavior of IDR($s$) and related methods. *SIAM Journal on Scientific Computing* 2012; **34**(5):A2576–A2598.
11. Simoncini V, Szyld DB. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications* 2007; **14**(1):1–59.
12. Paige CC, Saunders MA. Solutions of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 1975; **12**(4):617–629.
13. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**(3):856–869.
14. Freund RW, Nachtigal NM. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik* 1991; **60**(3):315–339.
15. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**:409–436 (1953).
16. Saad Y. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation* 1981; **37**(155):105–126.
17. van der Vorst HA, Sonneveld P. CGSTAB, a more smoothly converging variant of CG-S. *Technical Report Report 90-50*, Department of Mathematics and Informatics, Delft University of Technology, 1990.
18. van der Vorst HA. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1992; **13**(2):631–644.
19. Chan TF, Gallopoulos E, Simoncini V, Szeto T, Tong CH. A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems. *SIAM Journal on Scientific Computing* 1994; **15**(2):338–347. Iterative methods in numerical linear algebra (Copper Mountain Resort, CO, 1992).
20. Du L, Sogabe T, Zhang SL. Quasi-minimal residual smoothing technique for the IDR($s$) method. *JSIAM Letters* 2011; **3**:13–16.
21. Du L, Sogabe T, Zhang SL. A variant of the IDR($s$) method with the quasi-minimal residual strategy. *Journal of Computational and Applied Mathematics* 2011; **236**:621–630.
22. Kirchner S. IDR-Verfahren zur Lösung von Familien geshifteter linearer Gleichungssysteme. *Diplomarbeit*, Bergische Universität Wuppertal, Fachbereich Mathematik, Mai 2011.
23. Saad Y. *Iterative Methods for Sparse Linear Systems* (Second edition). Society for Industrial and Applied Mathematics: Philadelphia, PA, 2003.
24. Rendel O, Rizvanolli A, Zemke JPM. IDR: A new generation of Krylov subspace methods? *Linear Algebra and its Applications* 2013; **439**(4):1040–1061.
25. Sleijpen GL, van der Vorst HA. Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. *Numerical Algorithms* 1995; **10**(3-4):203–223.
26. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing* 1993; **14**(2):461–469.
27. van der Vorst HA, Vuik C. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications* 1994; **1**(4):369–386.
28. Notay Y. Flexible conjugate gradients. *SIAM Journal on Scientific Computing* 2000; **22**(4):1444–1460 (electronic).



DOI: 10.1002/nla

29. Szyld DB, Vogel JA. FQMR: a flexible quasi-minimal residual method with inexact preconditioning. *SIAM Journal on Scientific Computing* 2001; **23**(2):363–380. Copper Mountain Conference (2000).
30. Vogel JA. Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems. *Applied Mathematics and Computation* 2007; **188**(1):226–233.
31. Datta BN, Saad Y. Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment. *Linear Algebra and its Applications* 1991; **154/156**:225–244.
32. Gear CW, Saad Y. Iterative solution of linear equations in ODE codes. *SIAM Journal on Scientific and Statistical Computing* 1983; **4**(4):583–601.
33. Freund RW. Solution of shifted linear systems by quasi-minimal residual iterations. In *Numerical Linear Algebra (Kent, OH, 1992)*. de Gruyter: Berlin, 1993; 101–121.
34. Jegerlehner B. Krylov space solvers for shifted linear systems. *Technical Report IUHET-353*, Indiana University, Department of Physics, 1996. (Available from: arXiv.org: http://arXiv.org/abs/hep-lat/9612014, see also http://cdsweb.cern.ch/record/316892/files/9612014.pdf).
35. Frommer A, Glässner U. Restarted GMRES for shifted linear systems. *SIAM Journal on Scientific Computing* 1998; **19**(1):15–26 (electronic). Special issue on iterative methods (Copper Mountain, CO, 1996).
36. Simoncini V. Restarted full orthogonalization method for shifted linear systems. *BIT* 2003; **43**(2):459–466.
37. Frommer A. BiCGStab($\ell$) for families of shifted linear systems. *Computing* 2003; **70**(2):87–109.
38. van den Eshof J, Sleijpen GL. Accurate conjugate gradient methods for families of shifted systems. *Applied Numerical Mathematics* 2004; **49**(1):17–37.
39. Cullum J, Greenbaum A. Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**(2):223–247.
40. Boisvert RF, Pozo R, Remington K, Barrett RF, Dongarra JJ. Matrix market: a Web resource for test matrix collections. In *Quality of Numerical Software: Assessment and Enhancement*, Boisvert RF (ed.)., IFIP Advances in Information and Communication Technology. Chapman and Hall: London, 1997; 125–137.
41. Schönauer W. *Scientific Computing on Vector Computers*. Elsevier: Amsterdam, 1987.
42. Weiß R. Convergence behavior of generalized conjugate gradient methods. *Dissertation*, Universität Karlsruhe, 1990.
43. Erlangga YA, Oosterlee CW, Vuik C. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing* 2006; **27**(4):1471–1492 (electronic).
44. van Gijzen MB, Erlangga YA, Vuik C. Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian. *SIAM Journal on Scientific Computing* 2007; **29**(5):1942–1958 (electronic).
45. Notay Y. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis* 2010; **37**:123–146.
46. Yeung MC, Chan TF. ML($k$)BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors. *SIAM Journal on Scientific Computing* 1999/00; **21**(4):1263–1290 (electronic).



DOI: 10.1002/nla