Report 11-06, DIAM, TU Delft &
Bericht 156, INS, TU Hamburg-Harburg

FLEXIBLE AND MULTI-SHIFT INDUCED DIMENSION REDUCTION ALGORITHMS FOR SOLVING LARGE SPARSE LINEAR SYSTEMS*

MARTIN B. VAN GIJZEN† GERARD L.G. SLEIJPEN‡ AND JENS-PETER M. ZEMKE§

Abstract. We give two important 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 (QMRIDR) variant. Numerical examples are presented to show the effectiveness of these new IDR variants compared to existing ones and to other Krylov subspace methods.

Key words. Iterative methods; IDR; IDR(s); Krylov subspace methods; large sparse nonsymmetric linear systems.

AMS subject classifications. 65F10 (primary); 65F50

1. Introduction.

IDR(s) [27, 33] is a family of fast and remarkably robust methods for solving linear systems

$$\mathbf{A}\mathbf{x} = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$

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 [11, 36, 37], for analysis and more recent generalizations, see [12, 23, 25, 26]. In this contribution we 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 [12, 23, 25, 26] 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, 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 [17], GMRes [20], or QMR [7]. The prototype IDR(s) in [27] relies on an Orthogonal Residual (OR) approach in OrthoRes-style, i.e., the basis vectors are simply the residuals. OR approaches like CG [13] or FOM [18, 20] 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 in the pre-image of the operator $\mathbf{A}$: 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 [25] of the IDR approach to use higher degree polynomials will be treated elsewhere.

1.2. 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 δ_{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 $\underline{\mathbf{e}}_j$,

*Version of August 11, 2011, 20:37.

†Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands, (e-mail: M.B.vanGijzen@tudelft.nl),

‡Mathematical Institute, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, The Netherlands, (e-mail: sleijpen@math.uu.nl),

§Institut für Numerische Simulation, Technische Universität Hamburg-Harburg, Schwarzenbergstr. 95, D-21073 Hamburg, Germany (e-mail: zemke@tu-harburg.de).

$1 \leqslant j \leqslant n$, 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 letter, e.g., the columns of $\underline{\mathbf{H}}_n \in \mathbb{C}^{(n+1)\times n}$ are denoted by $\underline{\mathbf{h}}_j \in \mathbb{C}^{n+1}$, $1 \leqslant j \leqslant n$, and the columns of $\mathbf{U}_n \in \mathbb{C}^{n\times n}$ are denoted by $\mathbf{u}_j \in \mathbb{C}^n$, $1 \leqslant j \leqslant n$. The transpose and complex conjugate transpose of matrices and vectors is denoted by appending $^{\mathsf{T}}$ and $^{\mathsf{H}}$, respectively. The Moore-Penrose- or pseudo-inverse is denoted by appending †. Subspaces are denoted by calligraphic letters like $\mathcal{S}$, $\mathcal{K}_n$, and $\mathcal{G}_j$. In this report, the notation $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ denotes *strict* inclusion of sets, i.e., implies that $\mathcal{G}_j \neq \mathcal{G}_{j-1}$. The Euklidian 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 [12] 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, since the acronym stands for the phrase Generalized Minimal RESidual.

1.3. Outline. In §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 §2.3. This generalized Hessenberg decomposition forms in §3 the basis to derive a QMRIDR implementation along the lines of the QMR approach in [7]. The flexible QMRIDR variant is developed in §4, the multi-shift QMRIDR variant in §5. For the reader's convenience pseudo-code implementations of the main algorithms are collected in §6; illustrating numerical examples are given in §7. We conclude with some remarks about possible generalizations in §8.

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 below. As was shown in [12], the matrices

$$\mathbf{G}_n = \big(\mathbf{g}_1, \ldots, \mathbf{g}_n\big), \qquad \mathbf{G}_{n+1} = \big(\mathbf{G}_n, \mathbf{g}_{n+1}\big) \tag{2.1}$$

built from such vectors satisfy a so-called generalized Hessenberg decomposition [12, p. 8, Eqn. (2.1)]

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

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 Krylov subspace method; this is reflected in the structure of the Sonneveld pencil $(\mathbf{H}_n = \underline{\mathbf{I}}_n^{\mathsf{T}}\underline{\mathbf{H}}_n, \mathbf{U}_n)$ [12, 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.

2.1. The Sonneveld subspaces $\mathcal{G}_j$. IDR methods generate vectors $\mathbf{g}_i$, $i = 1, \ldots, n+1$, in the IDR spaces, a special case of Sonneveld subspaces [25, Definition 2.2, p. 2690], which are nested subspaces of shrinking dimension.

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

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

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 $\mathcal{G}_0 \subset \mathbb{C}^N$. 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 [26]. The parameter μ_j is a complex number that can, in principle, be chosen freely. We will sketch a method to select a 'good' μ_j in a later section. By construction the Sonneveld spaces are nested, but we can say more:

THEOREM 2.1 (IDR Theorem [27]). *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 [27, 23]. 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, see [25, 12]: we can characterize the IDR Sonneveld spaces in terms of the orthogonal complement of left block Krylov subspaces

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

as

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

see [23, Theorem 11, p. 1104]. Numerical experiments indicate that the "local closeness" of this Lanczos process to an unperturbed one is the driving force behind IDR based methods.

The Sonneveld spaces $\mathcal{G}_j$ as defined above 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 [27] 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 [27] 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.

2.2. 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 [27, 33], $s + 1$ 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. The algorithm below computes in each Sonneveld space $\mathcal{G}_j$ $s + 1$ orthonormalized vectors $\mathbf{g}_i$, $i = j(s + 1) + 1, \ldots, (j + 1)(s + 1)$, $j = 0, 1, \ldots$. 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 little vectors as possible to compute the next vector; other schemes are possible.

2.2.1. 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. Since 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

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{A}\mathbf{g}_n - \sum_{i=1}^{n} \mathbf{g}_i \beta_{i,n}, \quad n \leqslant s. \tag{2.2}$$

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

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

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 (2.2) can also be written as

$$\mathbf{A}\mathbf{g}_n = \sum_{i=1}^{n+1} \mathbf{g}_i \beta_{i,n}, \quad n \leqslant s \quad \Leftrightarrow \quad \mathbf{A}\mathbf{G}_s = \mathbf{G}_{s+1}\underline{\mathbf{H}}_s. \tag{2.3}$$

In the terminology of [12] the latter equality in (2.3) 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 $\mathbf{I}_s$.

2.2.2. 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

$$\mathbf{v}_n = \mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}, \tag{2.4}$$

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

$$\widetilde{\mathbf{R}}_0^{\mathsf{H}} \mathbf{v}_n = \mathbf{o}. \tag{2.5}$$

Combining (2.4) and (2.5) 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 μ_j, the first vector $\mathbf{t} \in \mathcal{G}_j$ can be computed by

$$\mathbf{t} = (\mathbf{A} - \mu_j \mathbf{I})\mathbf{v}_n. \tag{2.6}$$

To select a new μ_j, we aim at minimizing the norm of $\mathbf{t}$. An order to avoid a very large value of μ_j, which leads to a small angle between $\mathbf{t}$ and $\mathbf{v}_n$, we combine the minimization with the strategies explained in [24]. We refer to Section 6 for the precise algorithm to compute μ_j. As a final step we normalize the vector $\mathbf{t}$ which gives us $\mathbf{g}_{n+1}$:

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{t}, \quad \beta_{n+1,n} = \|\mathbf{t}\|. \tag{2.7}$$

The equations (2.4), (2.6), and (2.7) can be combined to give

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

which can be rewritten to

$$\mathbf{A}\Big(\mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}\Big) = \Big(\mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}\Big)\mu_j + \mathbf{g}_{n+1}\beta_{n+1,n}. \tag{2.8}$$

2.2.3. 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 (2.4)–(2.6). Since a linear combinations of vectors in the subspace $\mathcal{G}_j$ is also in $\mathcal{G}_j$, the vector

$$\mathbf{g}_{n+1}\beta_{n+1,n} = \mathbf{t} - \sum_{i=j(s+1)+1}^{n} \mathbf{g}_i \beta_{i,n} \tag{2.9}$$

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

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

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

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

The equations (2.4), (2.6), and (2.9) can be combined, which gives the following relation between the vectors $\mathbf{g}_i$:

$$\mathbf{g}_{n+1}\beta_{n+1,n} = (\mathbf{A} - \mu_j \mathbf{I})\Big(\mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}\Big) - \sum_{i=j(s+1)+1}^{n} \mathbf{g}_i \beta_{i,n}. \tag{2.10}$$

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

$$\mathbf{A}\Big(\mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}\Big) = \Big(\mathbf{g}_n - \sum_{i=n-s}^{n-1} \mathbf{g}_i \gamma_{i,n}\Big)\mu_j + \sum_{i=j(s+1)+1}^{n+1} \mathbf{g}_i \beta_{i,n}. \tag{2.11}$$

2.3. A generalized Hessenberg decomposition for the vectors $\mathbf{g}_n$. 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 β'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 (2.3) resulting from Arnoldi's method. By defining the matrices

$$\mathbf{U}_n = \begin{pmatrix} \mathbf{u}_1, \cdots, \mathbf{u}_n \end{pmatrix}, \qquad \underline{\mathbf{H}}_n = \begin{pmatrix} \underline{\mathbf{h}}_1 & \cdots & \underline{\mathbf{h}}_n \end{pmatrix}, \tag{2.12}$$

equation (2.3), (2.8), and (2.11) can be compactly written as a generalized Hessenberg decomposition

$$\mathbf{A}\mathbf{G}_n\mathbf{U}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n. \tag{2.13}$$

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 (2.13) will be at the basis of the solution algorithms for linear systems that we will present in the next sections.

EXAMPLE 1. *To illustrate the structure of $\mathbf{U}_n$ and $\underline{\mathbf{H}}_n$, we give these matrices for $s = 2$ and $n = 7$. 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* $\mathcal{G}_2$*.*

$$\mathbf{U}_7 = \begin{pmatrix} 1 & & -\gamma_{1,3} & & & & \\ & 1 & -\gamma_{2,3} & -\gamma_{2,4} & & & \\ & & 1 & -\gamma_{3,4} & -\gamma_{3,5} & & \\ & & & 1 & -\gamma_{4,5} & -\gamma_{4,6} & \\ & & & & 1 & -\gamma_{5,6} & -\gamma_{5,7} \\ & & & & & 1 & -\gamma_{6,7} \\ & & & & & & 1 \end{pmatrix}, \tag{2.14}$$

$$\underline{\mathbf{H}}_7 = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & -\mu_1\gamma_{1,3} & & & & \\ \beta_{2,1} & \beta_{2,2} & -\mu_1\gamma_{2,3} & -\mu_1\gamma_{2,4} & & & \\ & \beta_{3,2} & \mu_1 & -\mu_1\gamma_{3,4} & -\mu_1\gamma_{3,5} & & \\ & & \beta_{4,3} & \beta_{4,4}+\mu_1 & \beta_{4,5}-\mu_1\gamma 4,5 & -\mu_2\gamma_{4,6} & \\ & & & \beta_{5,4} & \beta_{5,5}+\mu_1 & -\mu_2\gamma_{5,6} & -\mu_2\gamma_{5,7} \\ & & & & \beta_{6,5} & \mu_2 & -\mu_2\gamma_{6,7} \\ & & & & & \beta_{7,6} & \beta_{7,7}+\mu_2 \\ & & & & & & \beta_{8,7} \end{pmatrix}. \tag{2.15}$$

2.4. A remark on the computation of μ_j. In the generalized Hessenberg decomposition that we have outlined above, 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 μ_j has to be selected. In the spirit of Arnoldi's algorithm it is a natural idea to select a new μ_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 μ_j may lead to very slow convergence or even stagnation of the solution algorithms based on the generalized Hessenberg decomposition (2.13) that will be presented in the next sections. In this paper we limit us to giving the strategy for selecting μ_j which gave us the best results. The detailed algorithm for computing this μ 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 systems $\mathbf{A}\mathbf{x} = \mathbf{r}_0$. The derivation of the algorithm is analogous to that of MinRes [17], GMRes [20], and QMR [7]. We remark that since BiCGStab [31, 30] is an IDR method the first implementation of an QMRIDR method is QMRCGStab [1], the QMR implementation of BiCGStab. More recently, a QMR variant of the prototype IDR(s) [27] has been published, cf. [4]. Our QMRIDR variant is a genuine implementation based on a stable basis expansion; the variant based on the prototype IDR(s) [27] shares the problems of the latter, especially the numerical instability arising for larger values of s.

For simplicity we assume that the starting vector is $\mathbf{x}_0 = 0$. 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:

$$\|\mathbf{r}_0 - \mathbf{A}\underline{\mathbf{x}}_n\| = \min_{\underline{\mathbf{x}} \in \mathcal{K}_n} \|\mathbf{r}_0 - \mathbf{A}\underline{\mathbf{x}}\|. \tag{3.1}$$

We assume that in the nth iteration we have available matrices $\mathbf{G}_n$, $\mathbf{U}_n$, $\mathbf{G}_{n+1}$ and $\underline{\mathbf{H}}_n$ that satisfy Eqn. (2.13). 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

$$\underline{\mathbf{x}}_n = \mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}_n = \mathbf{V}_n\underline{\mathbf{z}}_n \tag{3.2}$$

with unknown coefficient vector $\underline{\mathbf{z}}_n \in \mathbb{C}^n$. Here, we did define $\mathbf{V} = \big(\mathbf{v}_1, \ldots, \mathbf{v}_n\big)$, with $\mathbf{v}_i = \mathbf{g}_i$, $1 \leqslant i \leqslant s$, and $\mathbf{v}_i$, $s < i \leqslant n$ defined by Eqn. (2.4). The second equality in (3.2) is based on the definitions of $\mathbf{G}_n$ and $\mathbf{U}_n$ in Eqn. (2.1) and Eqn. (2.12), respectively.

Substituting this expression in (3.1) 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 $\mathbf{r}_0 = \mathbf{G}_{n+1}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|$ and $\mathbf{A}\mathbf{G}_n\mathbf{U}_n = \mathbf{G}_{n+1}\underline{\mathbf{H}}_n$ gives

$$\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n)\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}})\|. \tag{3.3}$$

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

$$\big\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n\big\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n} \big\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}\big\|. \tag{3.4}$$

Since

$$\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n)\| \leqslant \|\mathbf{G}_{n+1}\| \cdot \big\|\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - \underline{\mathbf{H}}_n\underline{\mathbf{z}}_n\big\|, \tag{3.5}$$

we 'quasi-minimize' (3.3) by minimizing this upper bound. We remark that we know a priori that

$$\|\mathbf{G}_{n+1}\| \leqslant \sqrt{\lceil (n+1)/(s+1)\rceil}, \tag{3.6}$$

because every consecutive block of $(s+1)$ columns consists of orthonormal vectors.

The relation (3.6) is proven as follows: let $j = \lceil (n+1)/(s+1)\rceil$ and define for $1 \leqslant i \leqslant j$ the block matrices $\mathbf{G}_{n+1}^{(i)} = \mathbf{G}_{n+1}(:, (i-1)(s+1)+1 : \min(i(s+1), n+1)$ with orthonormal columns, and denote the corresponding grouping of elements for some generic $\mathbf{w} \in \mathbb{C}^{n+1}$ by $\mathbf{w}^{(i)} = \mathbf{w}((i-1)(s+1)+1 : \min(i(s+1), n+1)$. Then

$$\begin{aligned}\|\mathbf{G}_{n+1}\|^2 &= \max_{\|\mathbf{w}\|=1} \|\mathbf{G}_{n+1}\mathbf{w}\|^2 = \max_{\|\mathbf{w}\|=1} \Big\|\sum_{i=1}^{j} \mathbf{G}_{n+1}^{(i)}\mathbf{w}^{(i)}\Big\|^2 \\ &\leqslant \sum_{i=1}^{j} \max_{\|\mathbf{w}^{(i)}\|=1} \|\mathbf{G}_{n+1}^{(i)}\mathbf{w}^{(i)}\|^2 = \sum_{i=1}^{j} \max_{\|\mathbf{w}^{(i)}\|=1} \|\mathbf{w}^{(i)}\|^2 = \sum_{i=1}^{j} 1 = j.\end{aligned} \tag{3.7}$$

Clearly, the upper bound in Eqn. (3.5) is minimized by the solution of Eqn. (3.4). The solution of this system can be determined with the aid of the tall QR decomposition of $\underline{\mathbf{H}}_n$:

$$\underline{\mathbf{H}}_n = \underline{\mathbf{Q}}_n\mathbf{R}_n, \qquad \underline{\mathbf{Q}}_n \in \mathbb{C}^{(n+1)\times n}, \quad \mathbf{R}_n \in \mathbb{C}^{n\times n}, \tag{3.8}$$

in which $\mathbf{Q}_n$ is a matrix with orthonormal columns and $\mathbf{R}_n$ is upper triangular. Since $\underline{\mathbf{H}}_n$ only has one nonzero subdiagonal, the QR decomposition of $\underline{\mathbf{H}}_n$ is typically computed using Givens rotations. The solution to (3.4) is then given by

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

which must be combined with (3.2) to give the approximate solution vector $\underline{\mathbf{x}}_n$.

The equations outlined above completely define the approximate solution vector, and can in principle be used to compute this vector. However, the direct naïve application of these equations to compute $\underline{\mathbf{x}}_n$ requires that $\mathbf{G}_n$ is explicitly available: the resulting algorithm would be a long recurrence algorithm in which the storage requirement and work per iteration grow with the number of iterations.

Next we will explain how the approximate solution vector

$$\underline{\mathbf{x}}_n = \mathbf{G}_n\mathbf{U}_n\mathbf{R}_n^{-1}\underline{\mathbf{Q}}_n^{\mathsf{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\| = \mathbf{V}_n\mathbf{R}_n^{-1}\underline{\mathbf{Q}}_n^{\mathsf{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|. \tag{3.9}$$

can be computed using short recurrences, in a way such that the storage requirements and work per iteration are constant. As was remarked before, the matrix $\underline{\mathbf{H}}_n$ is an extended upper Hessenberg matrix with upper bandwidth s. From this nonzero pattern immediately follows that $\underline{\mathbf{h}}_i \perp \underline{\mathbf{h}}_j, |i-$

$j| > s+1$, with $\underline{\mathbf{h}}_i$ and $\underline{\mathbf{h}}_j$ the ith and jth column of $\underline{\mathbf{H}}_n$, respectively. 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 $\mathbf{w}_n$, the nth column of $\mathbf{W}_n$, we write

$$\mathbf{W}_n \mathbf{R}_n \mathbf{e}_n = \mathbf{G}_n \mathbf{U}_n \mathbf{e}_n = \mathbf{V}_n \mathbf{e}_n, \tag{3.10}$$

in which $\mathbf{e}_n$ is the nth canonical basis vector of dimension n. The vector $\mathbf{G}_n \mathbf{U}_n \mathbf{e}_n = \mathbf{V}_n \mathbf{e}_n$ is explicitly available: it is the vector $\mathbf{v}_n$, cf. Eqn. (2.4). Equation (3.10) can therefore be rewritten as

$$\sum_{i=n-l-1}^{n} \mathbf{w}_i \mathbf{R}_n(i,n) = \mathbf{v}_n, \tag{3.11}$$

in which $\mathbf{R}_n(i,n)$ is entry (i,n) of the matrix $\mathbf{R}_n$. From (3.11) it follows that the update formula for $\mathbf{w}_n$ is

$$\mathbf{w}_n = \Big(\mathbf{v}_n - \sum_{i=n-s-1}^{n-1} \mathbf{w}_i \mathbf{R}_n(i,n)\Big) \cdot \frac{1}{\mathbf{R}_n(n,n)}. \tag{3.12}$$

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

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

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

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

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

$$\underline{\mathbf{x}}_n = \underline{\mathbf{x}}_{n-1} + \mathbf{w}_n \boldsymbol{\phi}_n(n), \tag{3.13}$$

in which $\boldsymbol{\phi}_n(n)$ is the nth 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 [19]. More recent variants include flexible CG [15], flexible QMR [28], and flexible BiCG and flexible BiCGStab [34]. 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[1], we allow for a different preconditioning matrix in every step: let $\mathbf{P}_n$ be the preconditioning matrix that may be different in every iteration n. A generalized Hessenberg relation is derived by replacing equation (2.6) by

$$\mathbf{t} = (\mathbf{A}\mathbf{P}_n^{-1} - \mu_j \mathbf{I})\mathbf{v}_n. \tag{4.1}$$

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. [19])

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

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

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

$$\underline{\mathbf{x}}_n = \widehat{\mathbf{V}}_n \underline{\mathbf{z}}_n. \tag{4.2}$$

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 (3.3), which again we solve by quasi minimizing. Proceeding in the same way as in (3.9), we compute the 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 nth 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 nth 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

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

Finally, the solution vector $\underline{\mathbf{x}}_n$ is computed using Eqn. (3.13).

From the outline above 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 above is in that case mathematically equivalent with FGMRES [19]. 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.

5. Multi-shift QMRIDR(s). Multi-shift methods have been considered in [3, p. 230–231], see also [10]. Many multi-shift Krylov subspace methods exists, we mention multi-shift QMR and multi-shift TFQMR [6], multi-shift GMRES(k) [9], multi-shift FOM(k) [22], and multi-shift BICGSTAB(ℓ) [8]. For some general comments and applications we refer to [14]. Implementation details may play a vital rôle, see the accurate multi-shift CG implementation in [29].

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 nth iteration we have available matrices $\mathbf{G}_n$, $\mathbf{U}_n$, $\mathbf{G}_{n+1}$ and $\underline{\mathbf{H}}_n$ that satisfy (2.13), 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 σ 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

$$\underline{\mathbf{x}}_n^\sigma = \mathbf{G}_n\mathbf{U}_n\underline{\mathbf{z}}_n^\sigma = \mathbf{V}_n\underline{\mathbf{z}}_n^\sigma. \tag{5.1}$$

Note that we can also write this as

$$\underline{\mathbf{x}}_n^\sigma = \mathbf{G}_{n+1}\underline{\mathbf{U}}_n\underline{\mathbf{z}}_n^\sigma, \tag{5.2}$$

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

$$\big\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}}_n^\sigma)\big\| = \min_{\underline{\mathbf{z}}\in\mathbb{C}^n}\big\|\mathbf{G}_{n+1}(\underline{\mathbf{e}}_1\|\mathbf{r}_0\| - (\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n)\underline{\mathbf{z}})\big\|. \tag{5.3}$$

In order to 'quasi-minimize' (5.3) we solve the least-squares problems

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

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

$$\underline{\mathbf{H}}_n - \sigma\underline{\mathbf{U}}_n = \underline{\mathbf{Q}}_n^\sigma\mathbf{R}_n^\sigma, \qquad \underline{\mathbf{Q}}_n^\sigma \in \mathbb{C}^{(n+1)\times n}, \quad \mathbf{R}_n^\sigma \in \mathbb{C}^{n\times n}. \tag{5.4}$$

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

$$\boldsymbol{\phi}_n^\sigma = (\underline{\mathbf{Q}}_n^\sigma)^{\mathsf{H}}\underline{\mathbf{e}}_1\|\mathbf{r}_0\|, \tag{5.5}$$

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

$$\mathbf{w}_n^\sigma = \Big(\mathbf{v}_n - \sum_{i=n-s-1}^{n-1}\mathbf{w}_i^\sigma\mathbf{R}_n^\sigma(i,n)\Big)\cdot\frac{1}{\mathbf{R}_n^\sigma(n,n)}, \tag{5.6}$$

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

$$\underline{\mathbf{x}}_n^\sigma = \underline{\mathbf{x}}_{n-1}^\sigma + \mathbf{w}_n^\sigma\boldsymbol{\phi}_n^\sigma(n). \tag{5.7}$$

We remark that the last four steps, the only steps actually depending on the shifts σ, i.e., those sketched in Eqns. (5.4)–(5.7), can be carried out in parallel and require exactly the same amount of floating point computations per step for all σ.

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 is stored. Columns of matrices are denoted by small characters, e.g., the variable $\underline{\mathbf{h}}$ as used in the algorithm contains the 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.

Algorithm 1 Flexible QMRIDR(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*
$\mathbf{W} = \mathbf{O} \in \mathbb{C}^{N\times(s+1)}$; $\mathbf{w} = \mathbf{o} \in \mathbb{C}^N$; $\mathbf{cs} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\mathbf{sn} = \mathbf{o} \in \mathbb{C}^{s+2}$;
$\phi = 0$; $\hat{\phi} = \|\mathbf{g}\|$; $\mathbf{g} = \frac{1}{\hat{\phi}}\mathbf{g}$; $n = 0$; $j = 0$; $\rho = \hat{\phi}$; $\rho_0 = \|\mathbf{b}\|$;
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* $\underline{\mathbf{U}}$
    $\mathbf{m} = \widetilde{\mathbf{R}}_0^{\mathsf{H}}\mathbf{g}$; // *Construct* $\mathbf{v} \perp \widetilde{\mathbf{R}}_0$
    if $n > s$ **then**
      Solve γ 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**
      $\beta = \mathbf{G}^{\mathsf{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\beta$;
      $\hat{\beta} = \mathbf{G}^{\mathsf{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\hat{\beta}$;
      $\beta = \beta + \hat{\beta}$;
      $\underline{\mathbf{h}}_{(s+1-k+1:s+1)} = \underline{\mathbf{h}}_{(s+1-k+1:s+1)} + \beta$;
    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

The algorithm for computing μ using the strategy explained in [24] is given as Algorithm 2. Note that with respect to the computation of a stable basis we could take $\mu = 0$ if $|\omega| < \texttt{eps}$, 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, compare with [2]. In the context of IDR methods this stagnation is *incurable*, which is clearly visible in the structure of the matrix $\underline{\mathbf{H}}_n$ once one of the μ'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. (2.15) with $\mu_1 = 0$, explicitely denoting the new zeros thus introduced:

$$\underline{\mathbf{H}}_7 = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & 0 & & & & \\ \beta_{2,1} & \beta_{2,2} & 0 & 0 & & & \\ & \beta_{3,2} & 0 & 0 & 0 & & \\ & & \beta_{4,3} & \beta_{4,4}+\mu_1 & \beta_{4,5} & -\mu_2\gamma_{4,6} & \\ & & & \beta_{5,4} & \beta_{5,5}+\mu_1 & -\mu_2\gamma_{5,6} & -\mu_2\gamma_{5,7} \\ & & & & \beta_{6,5} & \mu_2 & -\mu_2\gamma_{6,7} \\ & & & & & \beta_{7,6} & \beta_{7,7}+\mu_2 \\ & & & & & & \beta_{8,7} \end{pmatrix}. \tag{6.1}$$

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

$$\underline{\mathbf{z}}_{k+1} = \begin{pmatrix} \underline{\mathbf{z}}_k \\ 0 \end{pmatrix}, \quad \underline{\mathbf{x}}_{k+1} = \underline{\mathbf{x}}_k, \quad k \geqslant 2. \tag{6.2}$$

To circumvent this incurable stagnation we use as the default value for μ the cheap estimate $\sqrt{\|\mathbf{A}\|_1 \cdot \|\mathbf{A}\|_\infty}$ for $\|\mathbf{A}\|$. Better choices for the default value of μ exist, e.g., we could use good eigenvalue approximations to the small eigenvalues of $\mathbf{A}$ obtained from the Sonneveld pencil $(\mathbf{H}_n, \mathbf{U}_n)$ [12]; this is an active area of research.

Algorithm 2 COMPMU

INPUT: $\mathbf{t}, \mathbf{v} \in \mathbb{C}^N$;
OUTPUT: μ;
1: $\kappa = 0.7$; // *Value $\kappa = 0.7$ is recommended in [24]*
2: $\omega = (\mathbf{t}^{\mathsf{H}}\mathbf{v})/(\mathbf{t}^{\mathsf{H}}\mathbf{t})$;
3: $\rho = (\mathbf{t}^{\mathsf{H}}\mathbf{v})/(\|\mathbf{t}\|\|\mathbf{v}\|)$;
4: **if** $|\rho| < \kappa$ **then**
5: $\quad \omega = \omega\kappa/|\rho|$;
6: **end if**
7: **if** $|\omega| > \texttt{eps}$ **then**
8: $\quad \mu = 1/\omega$;
9: **else**
10: $\quad \mu = \sqrt{\|\mathbf{A}\|_1 \cdot \|\mathbf{A}\|_\infty}$;
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

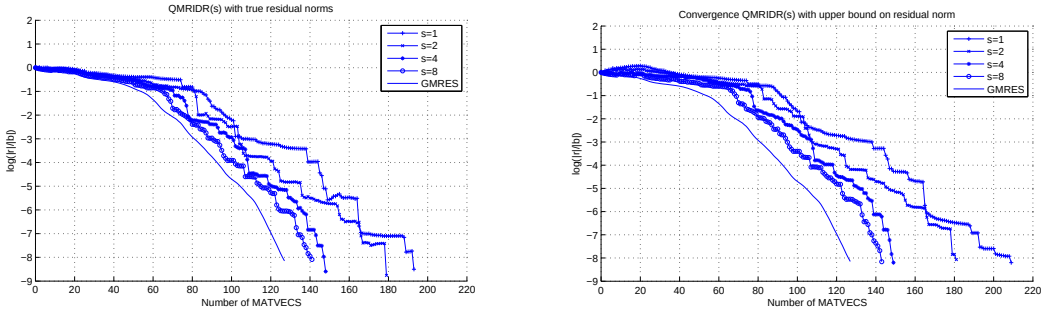
$$\begin{pmatrix} cs & sn \\ -\overline{sn} & cs \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}.$$

The multi-shift QMRIDR algorithm is presented as Algorithm 4. The algorithm is very similar to FQMRIDR, with as most important differences that no preconditioner is used in the multi-shift QMRIDR, and the computation of the update vectors for every shifted system is done in a loop (line 34 to line 45) over the number of shifts.

Algorithm 4 requires, apart from storage for the matrix $\mathbf{A}$ and for scalars, storage for $2s+3+n_\sigma(s+3)$ N-vectors, with n_σ the number of shifts.

Algorithm 3 ROTG

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



(a) Convergence QMRIDR(s), true residual norm.

(b) Convergence QMRIDR(s), upper bound.

FIG. 7.1. *Convergence of* QMRIDR(s)*, left true residual norms, right upper bound*

7. Numerical experiments. The experiments that are presented in this section have been performed on a standard laptop computer running under Linux, with four Intel® Core™ i5 CPU's and 4GB of RAM using MATLAB 7.11.

In all our experiments we take for $\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_s$ the orthogonalization 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 (3.6) 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. The system consists of 1104 equations. 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 $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| < 10^{-8}$.

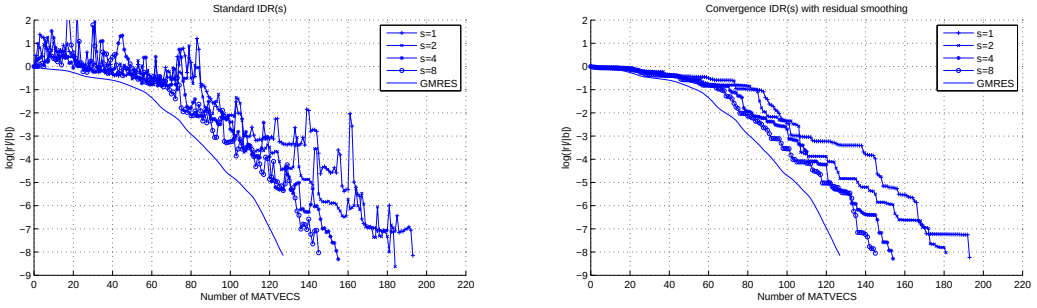
Figure 7.1(a) gives for the four different choices for s for every iteration the true residual norms for QMRIDR(s) and Figure 7.1(b) the upper bound on the residual norm. Also given in these figures are the convergence curves for full GMRES, which shows how close (increasingly closer for larger s) the QMRIDR-convergence curves are from optimal, 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.

Figure 7.2(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 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 [33] incorporates as an option residual smoothing using the technique developed by Hestenes and Stiefel [13, §7, p. 418–419], see also Schönauer and Weiß [21, 35]. Figure 7.2(b) shows the resulting monotonic convergence. Also

Algorithm 4 Multi-shift QMRIDR(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*
$\mu = 0$; $\mathbf{M} = \mathbf{O} \in \mathbb{C}^{s\times s}$; $\mathbf{G} = \mathbf{O} \in \mathbb{C}^{N\times s}$; $\mathbf{w} = \mathbf{o} \in \mathbb{C}^N$;
for $i = 1, n_\sigma$ **do**
  $\mathbf{W}^{\sigma_i} = \mathbf{O} \in \mathbb{C}^{N\times(s+1)}$;
  $\mathbf{cs}^{\sigma_i} = \mathbf{o} \in \mathbb{C}^{s+2}$; $\mathbf{sn}^{\sigma_i} = \mathbf{o} \in \mathbb{C}^{s+2}$;
  $\phi^{\sigma_i} = 0$; $\hat{\phi}^{\sigma_i} = \|\mathbf{g}\|$;
  $n = 0$; $j = 0$;
end for
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* $\mathbf{U}$
    $\mathbf{m} = \widetilde{\mathbf{R}}_0^{\mathsf{H}}\mathbf{g}$; // *Construct* $\mathbf{v} \perp \widetilde{\mathbf{R}}_0$
    if $n > s$ **then**
      Solve γ 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**
      $\beta = \mathbf{G}^{\mathsf{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\beta$;
      $\hat{\beta} = \mathbf{G}^{\mathsf{H}}_{(:,s-k+1:s)}\mathbf{g}$; $\mathbf{g} = \mathbf{g} - \mathbf{G}_{(:,s-k+1:s)}\hat{\beta}$;
      $\beta = \beta + \hat{\beta}$; $\underline{\mathbf{h}}_{(s+1-k+1:s+1)} = \underline{\mathbf{h}}_{(s+1-k+1:s+1)} + \beta$;
    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
      $[\mathbf{cs}^{\sigma_i}_{(s+2)}, \mathbf{sn}^{\sigma_i}_{(s+2)}, \mathbf{r}_{(s+2)}] = \mathrm{ROTG}(\mathbf{r}_{(s+2)}, \mathbf{r}_{(s+3)})$;
      $\phi^{\sigma_i} = \mathbf{cs}^{\sigma_i}_{(s+2)}\hat{\phi}^{\sigma_i}$; $\hat{\phi}^{\sigma_i} = -\overline{\mathbf{sn}}^{\sigma_i}_{(s+2)}\hat{\phi}^{\sigma_i}$; $\mathbf{cs}^{\sigma_i}_{(:,1:s+1)} = \mathbf{cs}^{\sigma_i}_{(:,2:s+2)}$; $\mathbf{sn}^{\sigma_i}_{(:,1:s+1)} = \mathbf{sn}^{\sigma_i}_{(:,2:s+2)}$;
      $\mathbf{w} = (\mathbf{v} - \mathbf{W}^{\sigma_i}\mathbf{r}_{(1:s+1)})/\mathbf{r}_{(s+2)}$; $\mathbf{W}^{\sigma_i}_{(:,1:s)} = \mathbf{W}^{\sigma_i}_{(:,2:s+1)}$; $\mathbf{W}^{\sigma_i}_{(:,s+1)} = \mathbf{w}$;
      $\underline{\mathbf{x}}^{\sigma_i} = \underline{\mathbf{x}}^{\sigma_i} + \phi^{\sigma_i}\mathbf{w}$;
      $\rho = \max(\rho, |\hat{\phi}|\sqrt{j + 1})$; // *Compute upper bound for maximum residual norm*
    end for
  end for
end while



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

FIG. 7.2. *Convergence of IDR(s) without and with residual smoothing*

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 7.1.

s	QMRIDR(s) True residual norm	QMRIDR(s) Upper bound	IDR(s) No smoothing	IDR(s) Smoothing
1	193	209	193	193
2	179	181	184	181
4	148	149	155	154
8	141	143	145	145

TABLE 7.1
Sherman 4: Iterations for the different (QMR)IDR variants

7.2. Example 2: SHERMAN 2. Next we consider the matrix SHERMAN 2 with corresponding right-hand-side, consisting of 1180 equation. This problem is very ill conditioned and as a result IDR(s) requires a high choice for s to converge [33]. The purpose of this example is to illustrate that for high values of s QMRIDR(s) may converge considerably faster than 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 7.2 gives for QMRIDR(s) and IDR(s) for the different values of s the number of iterations to reach the required accuracy. As can be seen from

Method	Iterations	Method	Iterations
QMRIDR(20)	1100	IDR(20)	1983
QMRIDR(40)	431	IDR(40)	945
QMRIDR(60)	294	IDR(60)	802
QMRIDR(80)	137	IDR(80)	779
QMRIDR(100)	131	IDR(100)	624
QMRIDR(120)	119	IDR(120)	337
QMRIDR(140)	119	IDR(140)	136

TABLE 7.2
Sherman 2: Iterations for increasing s

the table, IDR(s) requires for all choices of s, except $s = 140$, considerably more iterations than QMRIDR(s). 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 7.3,

which shows the convergence for QMRIDR(s) and IDR(s) for $s = 140$, and of full GMRES. Note that the required number of iterations for GMRES is 119, which is smaller than s. Since also the upper bound on the QMRIDR 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

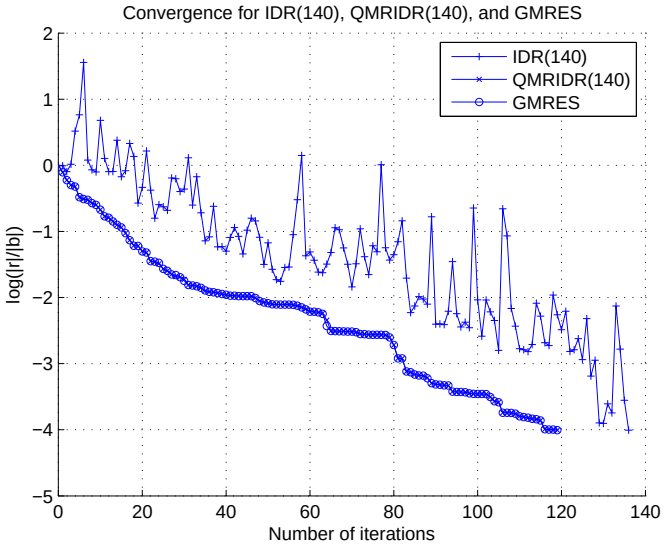


FIG. 7.3. *SHERMAN 2: Convergence of* QMRIDR(140), IDR(140), *and of full* GMRES

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, 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 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 $u(x, y, z) = x(1 - x)y(1 - y)z(1 - z)$. The problem is discretized using central differences with as grid size $h = 0.025$. The resulting linear system consists of approximately 60,000 equations. We take the following values for the parameters: $\epsilon = 1$ (diffusion), $\vec{\beta} = (0/\sqrt{5} \quad 250/\sqrt{5} \quad 500/\sqrt{5})^T$ (convection). The value for r (reaction) depends on the experiment. The resulting matrices are highly nonsymmetric and for $r > 0$ 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 $r = 0$. We use in every iteration 20 steps of full GMRES as preconditioner.

Figure 7.4 gives for every iteration the upper bound on the residual norms of QMRIDR(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.

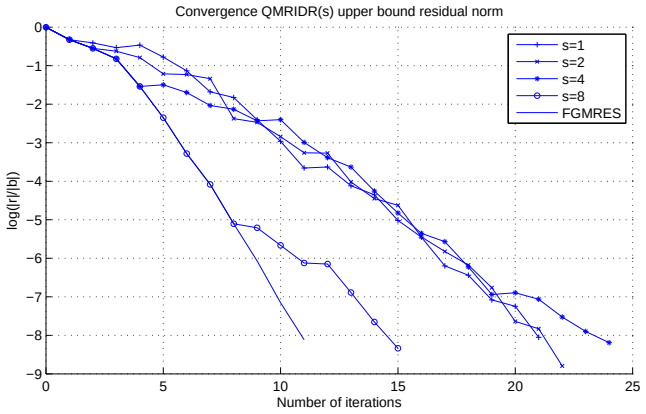


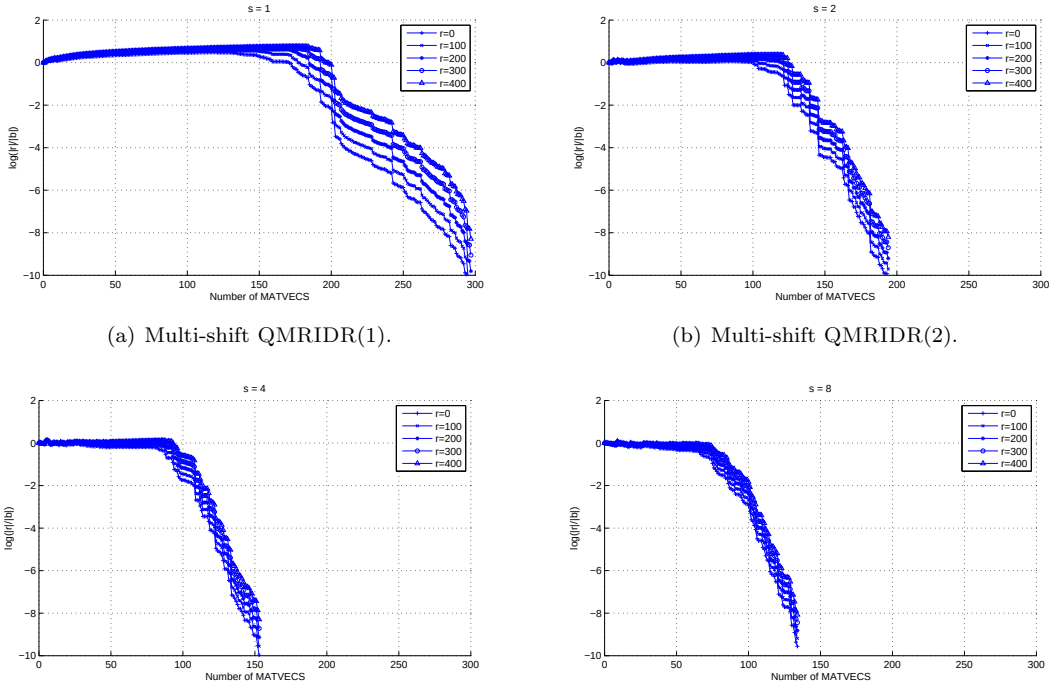
FIG. 7.4. *Convergence of* QMRIDR(s) *with a variable preconditioner*

The first observation is that all four QMRIDR variants achieve the required accuracy: the algorithm is in this respect robust for variations in the preconditioner. The second observations is that the rate of convergence for $s = 1$, $s = 2$, and $s = 4$ is almost the same, with a trend towards a *lower* rate of convergence for higher s. This tendency is stronger if less GMRES iterations are performed. The explanation is that with a variable preconditioner the $\mathbf{g}$-vectors are no longer in a sequence of nested subspaces, since the IDR-theorem does not hold any more. If less GMRES iterations are performed, the preconditioner becomes more variable. The third observation is QMRIDR(8) performs considerably better than the other QMRIDR variants. This illustrates that for larger values of s the quasi-minimization of the residual is close to a true minimization. As a result, the convergence of QMRIDR(8) is closer to the optimal convergence of FGMRES. Also, the convergence curves of the two methods coincide for the first s iterations. The convergence of QMRIDR(16) (not shown) completely coincides with the convergence of FGMRES, since the required number of iterations of FGMRES and of QMRIDR(16) is 12, which is smaller than $s = 16$.

In general, we have observed in many experiments that choosing a large value of s (i.e., in the order of the number of FGMRES iterations) can bring the convergence of QMRIDR(s) with a variable preconditioner arbitrarily close to the (optimal) convergence of full FGMRES. If for reasons of memory usage a relatively small value for s is used, it is most of the time best to choose $s = 1$ if a strongly variable preconditioner is being used.

7.3.3. Multi-shift QMRIDR(s). In the following experiments we take five different shifts: $r = 0$, $r = 100$, $r = 200$, $r = 300$, and $r = 400$, and study the simultaneous solution of the five shifted systems. Figure 7.3.3 shows in four different subplots the convergence curves for every shifted system for QMRIDR(1), QMRIDR(2), QMRIDR(4), and QMRIDR(8). Note that in QMRIDR(s) the (upper bound) on the residual norms is available for all shifted systems at no extra cost.

Although the multi-shift QMRIDR(s) algorithm 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 since many of these operations are related to solving the (projected) shifted system. Moreover, convergence is faster for systems with a smaller shift, while in the multi-shift algorithm termination occurs once the 'slowest' system has converged, which means that unnecessary operations are performed for the other systems. So the question arises how much more efficient the multi-shift algorithm is compared to the solution of the shifted 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 7.3 the number of iterations and CPU time for the multi-shift algorithm, and the accumulated numbers of



(a) Multi-shift QMRIDR(1).

(b) Multi-shift QMRIDR(2).

(c) Multi-shift QMRIDR(4).

(d) Multi-shift QMRIDR(8).

FIG. 7.5. *Convergence for the simultaneous solution of five shifted systems*

iterations and CPU time if the systems are solved subsequently. The results in Table 7.3 show that

s	Simultaneous solution Iterations (CPU time)	One system at a time Iterations (CPU time)
1	297 (8.2s)	1450 (21.6s)
2	194 (6.6s)	928 (15.2s)
4	153 (7.4s)	742 (15.3s)
8	134 (9.1s)	659 (25.9s)

TABLE 7.3
Convection-diffusion-reaction problem: iterations and CPU-time for the solution of the five shifted systems

the saving in CPU-time of multi-shift QMRIDR(s) over the subsequent solution of the shifted system with QMRIDR(s) for a single shift is significant.

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 flexible QMRIDR(s) in combination with a variable multigrid preconditioner. Then we apply multi-shift QMRIDR(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 equation:

$$-\Delta p - \left(\frac{2\pi f}{c(\mathbf{x})}\right)^2 p = s \qquad \text{in} \quad \Omega = (0,600)\times(0,1000), \tag{7.1}$$

$$s = \delta(x_1 - 300, x_2) \qquad \text{for} \quad x_1 = (0,600),\ x_2 = (0,1000), \tag{7.2}$$

$$\frac{\partial p}{\partial n} = 0 \qquad \text{on} \quad \partial\Omega. \tag{7.3}$$

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 7.6.

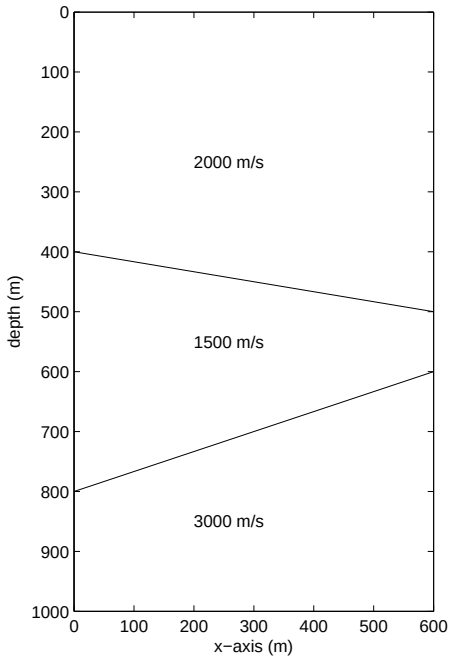


FIG. 7.6. *Problem geometry with sound velocity profile*

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 $\mathbf{M}$ a mass matrix. The gridsize is $h = 12.5m$ which yields a system of almost 3700 equations. 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 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 approximate by one multigrid cycle. In the experiments we use $z_2 = -iz_1$ [5, 32]. The action of $\mathbf{P}^{-1}$ is approximated by one cycle of AGMG, the AGregation based algebraic MultiGrid method proposed by Notay [16]. 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 QMRIDR(s) with one cycle of AGMG as preconditioner is displayed in Figure 7.7. This figure gives for every iteration the upper bound on the residual norms of QMRIDR(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, also for small s. 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

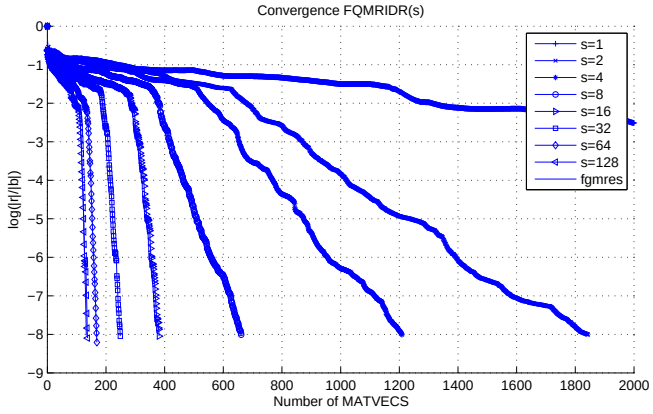
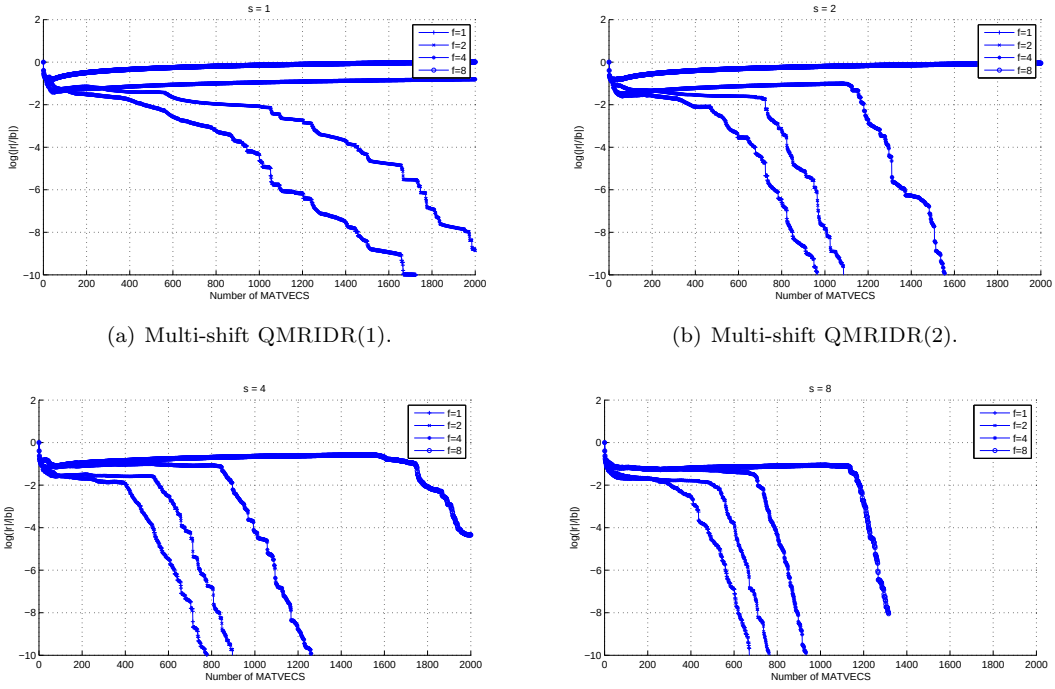


FIG. 7.7. *Convergence of* QMRIDR(s) *with a variable preconditioner*



(a) Multi-shift QMRIDR(1).

(b) Multi-shift QMRIDR(2).

(c) Multi-shift QMRIDR(4).

(d) Multi-shift QMRIDR(8).

FIG. 7.8. *Convergence for the simultaneous solution of five shifted systems*

in Section 7.3.2. For $s = 128$, the convergence curves of FGMRES and QMRIDR(s) coincide almost completely, since the two methods are mathematically equivalent in the first s iterations. The computing time for QMRIDR(16) is about the same as for full FGMRES, both take about 4s, but QMRIDR(16) uses of course much less memory than FGMRES.

7.4.3. Multi-shift QMRIDR(s). In the next experiments we use the multi-shift QMRIDR(s) algorithm to simultaneously solve four shifted systems, for the frequencies $f = 1$, $f = 2$, $f = 4$, and $f = 8$. Figure 7.8 shows in four different subplots the convergence curves for every shifted system for QMRIDR(1), QMRIDR(2), QMRIDR(4), and QMRIDR(8). Only for $s = 8$ all systems are solved to the required accuracy within 2000 iterations. Note that no preconditioner is used in these experiments. 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.

For this example the multi-shift algorithm is only slightly faster than for solving the systems consecutively: multi-shift QMRIDR(8) takes 6.2s and solving the systems one by one with QMRIDR(8) takes 7.2s.

8. Concluding remarks. We have presented two new members from the family of IDR methods highlighting the rôle of the underlying generalized Hessenberg decomposition. The derivation of the flexible and the multi-shift QMRIDR should allow others to easily incorporate ideas from the context of Krylov subspace methods to the slightly more sophisticated 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[2] given in the abstract of [38]:

> "... 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.

9. Acknowledgements. The authors would like to thank Olaf Rendel for pointing out the proof for the upper bound in Eqn. (3.6) and the occurrence of incurable breakdowns in case of $\mu = 0$.

REFERENCES

[1] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. H. Tong, *A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems*, SIAM J. Sci. Comput., 15 (1994), pp. 338–347. Iterative methods in numerical linear algebra (Copper Mountain Resort, CO, 1992).

[2] Jane Cullum and Anne Greenbaum, *Relations between Galerkin and norm-minimizing iterative methods for solving linear systems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 223–247.

[3] Biswa Nath Datta and Youcef Saad, *Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment*, Linear Algebra Appl., 154/156 (1991), pp. 225–244.

[4] Lei Du, Tomohiro Sogabe, and Shao-Liang Zhang, *Quasi-minimal residual smoothing technique for the IDR(s) method*, JSIAM Letters, 3 (2011), pp. 13–16.

[5] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492 (electronic).

[6] Roland W. Freund, *Solution of shifted linear systems by quasi-minimal residual iterations*, in Numerical linear algebra (Kent, OH, 1992), de Gruyter, Berlin, 1993, pp. 101–121.

[7] Roland W. Freund and Noël M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[8] A. Frommer, *BiCGStab(ℓ) for families of shifted linear systems*, Computing, 70 (2003), pp. 87–109.

[9] Andreas Frommer and Uwe Glässner, *Restarted GMRES for shifted linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 15–26 (electronic). Special issue on iterative methods (Copper Mountain, CO, 1996).

[10] C. W. Gear and Y. Saad, *Iterative solution of linear equations in ODE codes*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 583–601.

[11] Martin H. Gutknecht, *IDR explained*, Electron. Trans. Numer. Anal., 36 (2009/10), pp. 126–148.

[12] Martin H. Gutknecht and Jens-Peter M. Zemke, *Eigenvalue computations based on IDR*, Bericht 145, TUHH, Institute of Numerical Simulation, May 2010. Online available at http://doku.b.tu-harburg.de/volltexte/2010/875/.

[13] Magnus R. Hestenes and Eduard Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436 (1953).

[14] B. Jegerlehner, *Krylov space solvers for shifted linear systems*, arXiv.org HEP-LAT preprint, technical report IUHET-353, Indiana University, Department of Physics, 1996. Online available at arXiv.org: http://arXiv.org/abs/hep-lat/9612014, see also http://cdsweb.cern.ch/record/316892/files/9612014.pdf.

[15] Yvan Notay, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460 (electronic).

[16] ———, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.

[17] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[18] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[19] Youcef Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[20] Youcef Saad and Martin H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[21] W. Schönauer, *Scientific Computing on Vector Computers*, Elsevier, Amsterdam, 1987.

[2] This remark is about ML(k)BiCGStab, a predecessor of IDR(s), in some sense the first IDR(s) variant.

[22] V. Simoncini, *Restarted full orthogonalization method for shifted linear systems*, BIT, 43 (2003), pp. 459–466.
[23] Gerard L.G. Sleijpen, Peter Sonneveld, and Martin B. van Gijzen, *Bi-CGSTAB as an induced dimension reduction method*, Appl. Numer. Math., 60 (2010), pp. 1100–1114.
[24] Gerard L.G. Sleijpen and Henk A. van der Vorst, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numer. Algorithms, 10 (1995), pp. 203–223.
[25] Gerard L.G. Sleijpen and Martin B. van Gijzen, *Exploiting BiCGstab(ℓ) strategies to induce dimension reduction*, SIAM J. Sci. Comput., 32 (2010), pp. 2687–2709.
[26] Peter Sonneveld, *On the convergence behaviour of IDR(s)*, Technical Report 10-08, Department of Applied Mathematical Analysis, Delft University of Technology, Delft, 2010.
[27] Peter Sonneveld and Martin B. van Gijzen, *IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 31 (2008/09), pp. 1035–1062.
[28] Daniel B. Szyld and Judith A. Vogel, *FQMR: a flexible quasi-minimal residual method with inexact preconditioning*, SIAM J. Sci. Comput., 23 (2001), pp. 363–380. Copper Mountain Conference (2000).
[29] Jasper van den Eshof and Gerard L.G. Sleijpen, *Accurate conjugate gradient methods for families of shifted systems*, Appl. Numer. Math., 49 (2004), pp. 17–37.
[30] H. A. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[31] H. A. van der Vorst and P. Sonneveld, *CGSTAB, a more smoothly converging variant of CG-S*, Report 90-50, Department of Mathematics and Informatics, Delft University of Technology, 1990.
[32] M. B. van Gijzen, Y. A. Erlangga, and C. Vuik, *Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian*, SIAM J. Sci. Comput., 29 (2007), pp. 1942–1958 (electronic).
[33] Martin B. van Gijzen and Peter Sonneveld, *An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties.*, Technical Report 10-16, Department of Applied Mathematical Analysis, Delft University of Technology, Delft, 2010. (revised version of report 08-21).
[34] Judith A. Vogel, *Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems*, Appl. Math. Comput., 188 (2007), pp. 226–233.
[35] Rüdiger Weiss, *Convergence Behavior of Generalized Conjugate Gradient Methods*, Dissertation, Universität Karlsruhe, 1990.
[36] N. Nakashima Y. Onoue, S. Fujino, *A difference between easy and profound preconditionings of IDR(s) method*, Transactions of the Japan Society for Computational Engineering and Science, (2008). (in Japanese).
[37] ———, *An overview of a family of new iterative methods based on IDR theorem and its estimation.*, in Proceedings of the International Multi-Conference of Engineers and Computer Scientists 2009, vol. II, IMECS 2009, Hong Kong, March 18–20 2009, pp. 129–2134.
[38] Man-Chung Yeung and Tony F. Chan, *ML(k)BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors*, SIAM J. Sci. Comput., 21 (1999/00), pp. 1263–1290 (electronic).