SIAM J. SCI. COMPUT.
Vol. 32, No. 5, pp. 2687–2709

EXPLOITING BICGSTAB(ℓ) STRATEGIES TO INDUCE DIMENSION REDUCTION*

GERARD L. G. SLEIJPEN† AND MARTIN B. VAN GIJZEN‡

Abstract. IDR(s) [P. Sonneveld and M. B. van Gijzen, *SIAM J. Sci. Comput.*, 31 (2008), pp. 1035–1062] and BiCGstab(ℓ) [G. L. G. Sleijpen and D. R. Fokkema, *Electron. Trans. Numer. Anal.*, 1 (1993), pp. 11–32] are two of the most efficient short-recurrence iterative methods for solving large nonsymmetric linear systems of equations. Which of the two is best depends on the specific problem class. In this paper we describe IDRstab, a new method that combines the strengths of IDR(s) and BiCGstab(ℓ). To derive IDRstab we extend the results that we reported on in [G. L. G. Sleijpen, P. Sonneveld, and M. B. van Gijzen, *Appl. Numer. Math.*, (2009), DOI: 10.1016/j.apnum.2009.07.001], where we considered Bi-CGSTAB as an induced dimension reduction (IDR) method. We will analyze the relation between hybrid Bi-CG methods and IDR and introduce the new concept of the Sonneveld subspace as a common framework. Through numerical experiments we will show that IDRstab can outperform both IDR(s) and BiCGstab(ℓ).

Key words. Bi-CGSTAB, Bi-CG, iterative linear solvers, Krylov subspace methods, induced dimension reduction

AMS subject classifications. 65F15, 65N25

DOI. 10.1137/090752341

1. Introduction.

Bi-CGSTAB [14] is the most popular short-recurrence method for solving large nonsymmetric systems of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

of order n. Bi-CGSTAB can be seen as a combination of two different techniques: Bi-CG on the one hand and a one-step minimal residual method (such as GMRES(1)) on the other hand. The Bi-CG part of the algorithm ensures, at least in exact arithmetic, termination at the exact solution in a finite number of steps. This property causes superlinear convergence during the iterative process. Bi-CG, on the other hand, does not possess an error minimization property; the residual norm may go up during the process. To introduce some kind of error minimization, Bi-CGSTAB performs a minimal residual norm step in every iteration. This residual minimization step is performed without extra cost, thus using the computational work more efficiently than in Bi-CG, resulting in a faster convergence. The combination of these two complementary methods explains to a large extent the success of Bi-CGSTAB.

IDR(s) [12] is a new short-recurrence Krylov subspace method for solving large nonsymmetric linear systems. The induced dimension reduction (IDR) method generates residuals that are forced to be in subspaces $\mathcal{G}_k$ of decreasing dimension. These nested subspaces are related by $\mathcal{G}_{k+1} = (\mathbf{I} - \omega_{k+1}\mathbf{A})(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp})$, where $\widetilde{\mathbf{R}}_0^{\perp}$ is the orthogonal complement of the range of a fixed $n \times s$ matrix $\widetilde{\mathbf{R}}_0$, and the ω_k's are nonzero

*Received by the editors March 11, 2009; accepted for publication (in revised form) June 7, 2010; published electronically August 31, 2010. Part of this research has been funded by the Dutch BSIK/BRICKS project.

http://www.siam.org/journals/sisc/32-5/75234.html

†Mathematical Institute, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, The Netherlands (sleijpen@math.uu.nl).

‡Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands (m.b.vanGijzen@tudelft.nl).

scalars. This working principle is different from the more conventional Krylov subspace methods like Bi-CGSTAB that construct residuals in the *growing* Krylov subspace

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) \equiv \text{span}\left(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\right).$$

The numerical examples that are described in [12] show that IDR(s), with $s > 1$ and not too large, is quite competitive and outperforms Bi-CGSTAB for important problem classes.

Although the ideas behind IDR(s) and Bi-CGSTAB are quite different, the two methods are mathematically closely related. Specifically, IDR(1) is mathematically equivalent to Bi-CGSTAB, and IDR(s) with $s > 1$ is related (but not mathematically equivalent) to the Bi-CGSTAB generalization ML(k)BiCGSTAB [17] of Yeung and Chan. This latter method is a block version of Bi-CGSTAB for systems with one right-hand-side vector, which uses multiple initial shadow residuals. We refer to [12], [7], and [3] for the details.

Like Bi-CGSTAB, IDR(s) lacks a global error minimization property. In order to smooth the convergence, the ω_k's are chosen such that the next residual is minimized in norm. As is explained in [12], a new ω can be chosen every $s+1$ step (matrix-vector multiplication (MV)). Note that for $s = 1$ this is exactly the same as in Bi-CGSTAB, where a new ω is selected after every second MV such that the next residual is minimized in norm.

For problems with a strongly nonsymmetric matrix and in which all the data are real, a linear minimal residual step does not work well. It is easy to see that the ω's are zero for skew-symmetric matrices and must be small for nearly skew-symmetric matrices, which will lead to (near) breakdown of the algorithm. Consequently, Bi-CGSTAB is less suited for this class of problems. In order to overcome this problem, Gutknecht proposed to combine Bi-CG with quadratic stabilization polynomials (i.e., with GMRES(2)), which led to BiCGStab2 [2]. Sleijpen and Fokkema combined Bi-CG with stabilization polynomials of degree ℓ, which yields BiCGstab(ℓ) [6].

The linear minimal residual step in IDR(s) also makes this method less suited for problems with a strongly nonsymmetric matrix. Example 3 in [12] presents such a problem. This example shows a very poor performance for IDR(1). The performance of the method can be improved significantly by choosing s larger than 1. However, BiCGstab(ℓ), with $\ell = 2$ or 4, shows a vastly superior convergence and outperforms IDR(s) irrespective of the choice of s.

It is therefore a natural idea to combine IDR(s) and BiCGstab(ℓ) into a new method. In order to derive such a method, we will extend the results in [7], where we studied the mathematical relation between Bi-CGSTAB and IDR. We will call the resulting method IDR(s)stab(ℓ), or IDRstab for short. IDRstab unifies IDR(s) and BiCGstab(ℓ): by choosing $\ell = 1$ it is mathematically equivalent to IDR(s), and by choosing $s = 1$ it is mathematically equivalent to BiCGstab(ℓ).

We present numerical experiments that demonstrate that the combined method is more efficient than either IDR(s) or BiCGstab(ℓ) for certain classes of problems. In particular, we show that IDRstab with s and ℓ larger than 1 can be significantly faster than both IDR(s) and BiCGstab(ℓ).

This paper is organized as follows. The next section reviews the IDR principle and its relation to hybrid Bi-CG methods. It also provides new insights. In particular, it introduces the concept of the Sonneveld subspace. Section 3 explains how BiCGstab(ℓ) is derived from Bi-CG, and section 4 presents an IDR algorithm. We include the derivation of both BiCGstab(ℓ) and IDR in this paper for completeness and to enhance

readability and also because the derivations and the resulting schemes are slightly different from the standard ones in the literature. Section 5 explains that these new variants facilitate easy incorporation of ℓth-degree stabilization polynomials. To this end we give insightful schemes on how the different vector quantities in IDRstab are related. The excellent performance of the new method is illustrated with numerical experiments in section 6.

Some remarks on the notation.

Notation 1.1. If $\widetilde{\mathbf{R}}$ is an $n \times s$ matrix and $\mathbf{v}$ is an n-vector, then we put $\mathbf{v} \perp \widetilde{\mathbf{R}}$ if $\mathbf{v}$ is orthogonal to all column vectors of $\widetilde{\mathbf{R}}$, and we say that $\mathbf{v}$ is *orthogonal to* $\widetilde{\mathbf{R}}$. The linear subspace of all vectors $\mathbf{v}$ that are orthogonal to $\widetilde{\mathbf{R}}$ is denoted by $\widetilde{\mathbf{R}}^{\perp}$.

If $\mathbf{V}_1, \ldots, \mathbf{V}_k$ are matrices with column vectors of size n, then $\mathrm{span}(\mathbf{V}_1, \ldots, \mathbf{V}_k)$ is the subspace spanned by all column vectors of all $\mathbf{V}_j$. We put $\mathbf{V}_1, \ldots, \mathbf{V}_k \perp \widetilde{\mathbf{R}}$ if $\mathrm{span}(\mathbf{V}_1, \ldots, \mathbf{V}_k) \subset \widetilde{\mathbf{R}}^{\perp}$. n-Vectors $\mathbf{v}$ are identified with the $n \times 1$ matrices $[\mathbf{v}]$.

Notation 1.2. Updates of the form $\mathbf{v} - \mathbf{C}\beta$ will play a crucial role in this paper. Here, $\mathbf{v}$ is an n-vector and $\mathbf{C}$ is an $n \times s$ matrix. When considering such updates, both $\mathbf{v}$ and $\mathbf{C}$ are available. Often, the s-vector β is determined by an orthogonality requirement $\mathbf{v} - \mathbf{C}\beta \perp \widetilde{\mathbf{R}}$, where $\widetilde{\mathbf{R}}$ is some given $n \times s$ matrix. For ease of notation, we will simply put

$$\text{“}\mathbf{v} - \mathbf{C}\beta \perp \widetilde{\mathbf{R}}\text{”} \quad \text{if we mean} \quad \text{“let } \beta \text{ be such that } \mathbf{v} - \mathbf{C}\beta \perp \widetilde{\mathbf{R}}.\text{”}$$

Note that, with $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{C}$, β can be formally computed as $\beta = \sigma^{-1}(\widetilde{\mathbf{R}}^{*}\mathbf{v})$ (in practice, a more stable variant should be used). The operator $\mathbf{I} - \mathbf{C}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$ is an oblique projection onto the orthogonal complement of $\widetilde{\mathbf{R}}$.

Notation 1.3. We will follow a number of MATLAB conventions:

— if $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_s]$, then $\mathbf{W}_{(:,1:q)} \equiv [\mathbf{w}_1, \ldots, \mathbf{w}_q]$ and $\mathbf{W}_{(:,q)} \equiv \mathbf{w}_q$ $(q \leq s)$;

— $[\mathbf{U}_0; \ldots; \mathbf{U}_j] \equiv [\mathbf{U}_0^{\mathrm{T}}, \ldots, \mathbf{U}_j^{\mathrm{T}}]^{\mathrm{T}}$ (if sizes permit).

2. The Sonneveld subspaces. The aim is to derive a method that combines the features of IDR(s) and BiCGstab(ℓ). In order to derive such a method we first look at the relation between the spaces in which these methods construct the residuals. Using the new concept of the Sonneveld subspace we will bring IDR-type methods and hybrid Bi-CG methods (also called Lanczos-type product methods) into a common framework.

The IDR method exploits the following result, from which we learn how to construct a strictly increasing sequence of nested linear subspaces. For a proof we refer to [12, 7].

THEOREM 2.1. *Let* $\widetilde{\mathbf{R}}_0 = [\widetilde{\mathbf{r}}_1, \ldots, \widetilde{\mathbf{r}}_s]$ *be an* $n \times s$ *matrix, and let* (μ_k) *be a sequence in* $\mathbb{C}$. *With* $\mathcal{G}_0 \equiv \mathbb{C}^n$, *define*

$$\mathcal{G}_{k+1} \equiv (\mu_{k+1}\mathbf{I} - \mathbf{A})(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}) \quad (k = 0, 1, \ldots).$$

If $\widetilde{\mathbf{R}}_0^{\perp}$ *does not contain an eigenvector of* $\mathbf{A}$, *then, for all* $k = 0, 1, \ldots$, *we have that*

$$(1)\ \mathcal{G}_{k+1} \subset \mathcal{G}_k \quad \textit{and} \quad (2)\ \dim \mathcal{G}_{k+1} < \dim \mathcal{G}_k \ \textit{unless}\ \mathcal{G}_k = \{\mathbf{0}\}.$$

IDR updates an approximation with residual in $\mathcal{G}_k$ to an approximation with residual in $\mathcal{G}_{k+1}$. In view of the above theorem, we know that the residual eventually will be zero and the exact solution will be detected. Fortunately, in practice convergence is much faster: the residual becomes sufficiently small in norm for most problems long before this moment.

We will relate the "IDR" spaces $\mathcal{G}_k$ to Krylov subspaces through the new concept of the Sonneveld space that we will define next.

DEFINITION 2.2. *For a polynomial P_k and an $n \times s$ matrix $\widetilde{\mathbf{R}}_0$, consider the linear subspace*

$$\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0) \equiv \{P_k(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)\}.$$

Here, k is the (exact) degree of the polynomial P_k, and $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$ is the kth (block) Krylov subspace generated by $\mathbf{A}^*$ and $\widetilde{\mathbf{R}}_0$ as follows:

$$\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{R}}_0) = \left\{ \sum_{j<k} (\mathbf{A}^*)^j \widetilde{\mathbf{R}}_0 \gamma_j \mid \gamma_j \in \mathbb{C}^s \right\}.$$

For reasons explained in Note 2.5 *we call $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$ the (P_k)* Sonneveld subspace *(generated by $\mathbf{A}$ and $\widetilde{\mathbf{R}}_0$); k is the* order *of the Sonneveld subspace.*

Note 2.3. $\mathbf{v} \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$ if and only if $\mathbf{v}, \mathbf{A}\mathbf{v}, \ldots, \mathbf{A}^{k-1}\mathbf{v} \perp \widetilde{\mathbf{R}}_0$.

To increase the order of the Sonneveld subspace, the order of the *shadow* Krylov subspace $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$ has to be increased by one, which requires s MVs. An additional MV is needed to increase the degree of the polynomial P_k.

The following result relates the "IDR" spaces $\mathcal{G}_k$ to the Sonneveld subspaces. For a proof, we refer to [7].

THEOREM 2.4. *With $\widetilde{\mathbf{R}}_0$, (μ_k), and $\mathcal{G}_k$ as in Theorem* 2.1*, and with P_k defined by $P_k(\lambda) \equiv \prod_{j=1}^{k}(\mu_j - \lambda)$ $(\lambda \in \mathbb{C})$, we have that*

$$\mathcal{G}_k = \mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0). \tag{2.1}$$

Since IDR methods are characterized by the fact that they compute residuals $\mathbf{r}^{\mathrm{IDR}} \in \mathcal{G}_k$, we have

$$\mathbf{r}^{\mathrm{IDR}} \in \mathcal{G}_k = \mathcal{S}(P_k^{\mathrm{IDR}}, \mathbf{A}, \widetilde{\mathbf{R}}_0),$$

with $P_k^{\mathrm{IDR}}(\lambda) \equiv \prod_{j \le k}(1 - \omega_j \lambda)$ $(\lambda \in \mathbb{C})$. The $\omega_j = 1/\mu_j$ are selected to minimize the residual norm. Note that by defining the polynomial in terms of ω's, we automatically generate a residual polynomial. Of course, the ω's should not be zero to avoid breakdown.

Hybrid methods produce residuals that can be expressed as $\mathbf{r}^{\mathrm{hybrid}} = P_k^{\mathrm{hybrid}}(\mathbf{A})\mathbf{r}_k^{\mathrm{BiCG}}$, where $\mathbf{r}_k^{\mathrm{BiCG}} \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$ is the kth Bi-CG residual. Hence, we have

$$\mathbf{r}^{\mathrm{hybrid}} \in \mathcal{S}(P_k^{\mathrm{hybrid}}, \mathbf{A}, \widetilde{\mathbf{r}}_0)\ .$$

The polynomial P_k^{hybrid} determines the specific hybrid variant. The selection $P_k(\lambda) = \prod_{j \le k}(1 - \omega_j \lambda)$ yields Bi-CGSTAB. The above framework makes it clear that Bi-CGSTAB is equivalent to IDR(1); see also [12], [7]. For BiCGstab(ℓ) we have

$$P_k^{\mathrm{BiCGstab}(\ell)}(\lambda) = \prod_{j \le k/\ell} (1 - \sum_{i=1}^{\ell} \gamma_{i,j\ell} \lambda^i);$$

that is, BiCGstab(ℓ) uses a product of ℓ-degree polynomials. The $\gamma_{i,j\ell}$ are selected to minimize the residual norm every ℓth step.

With the above terminology and definitions we can now give the Sonneveld subspace in which the residuals of our new method IDRstab should lie as follows:

$$\mathbf{r}^{\mathrm{IDRstab}} \in \mathcal{S}(P_k^{\mathrm{BiCGstab}(\ell)}, \mathbf{A}, \widetilde{\mathbf{R}}_0).$$

IDRstab can therefore be viewed as BiCGstab(ℓ) with an s-dimensional initial shadow residual $\widetilde{\mathbf{R}}_0$ instead of a one-dimensional one $\widetilde{\mathbf{r}}_0$.

In practical implementations of IDR methods as well as of hybrid Bi-CG methods, intermediate residuals will be generated to bring a residual from a kth order Sonneveld subspace to one of order $k+1$. We will refer to the residuals that "arrive first" in a Sonneveld subspace as *primary* residuals; the others are called *secondary* residuals.

Note 2.5. Given the fact that (sub)spaces are named after mathematicians, we feel that the name Sonneveld subspace is appropriate. Sonneveld introduced the IDR concept of Theorem 2.1 in a joint paper with Wesseling [16]. In this paper from 1980, he also gave an algorithm that is equivalent to Bi-CGSTAB. The relation to hybrid Bi-CG expressed in Theorem 2.4 and the fundamental principle in hybrid Bi-CG that any polynomial of appropriate degree can be used to avoid multiplication by $\mathbf{A}^*$ were introduced in Sonneveld's CGS paper [11] from 1989.

3. From Bi-CG to BiCGstab(ℓ). In this section, we recall the derivation of Bi-CGSTAB and BiCGstab(ℓ). Our derivation here slightly differs from the standard one and leads to slightly more expensive (but equivalent) algorithms: they require a few vector updates more per step. However, we feel that this modified form helps to clarify the derivation of our IDR variants in sections 4–5. It also explains how the IDR variants relate to BiCGstab(ℓ).

3.1. Bi-CG. Bi-CG uses coupled two term recurrences,

$$\mathbf{u}_k = \mathbf{r}_k - \beta_{k-1}\mathbf{u}_{k-1}, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{u}_k, \tag{3.1}$$

to produce a residual $\mathbf{r}_k$ at step k that is orthogonal to the kth *shadow* Krylov subspace $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$. It exploits the fact that, to achieve this orthogonality, it suffices to put the vectors $\mathbf{A}\mathbf{u}_k$ and $\mathbf{r}_{k+1}$ orthogonal to one vector only: with $\widetilde{\mathbf{r}}_0, \ldots, \widetilde{\mathbf{r}}_j$ spanning $\mathcal{K}_{j+1}(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$ for all $j \le k$, the coefficients β_{k-1} and α_k are such that $\mathbf{A}\mathbf{u}_k \perp \widetilde{\mathbf{r}}_{k-1}$ and $\mathbf{r}_{k+1} \perp \widetilde{\mathbf{r}}_k$. The approximate solutions $\mathbf{x}_{k+1}$ are obtained by updating $\mathbf{x}_k$ by $+\alpha_k \mathbf{u}_k$: here, as elsewhere in this paper (and in hybrid Bi-CG methods), residuals are updated by vectors of the form $-\mathbf{A}\mathbf{u}$ with $\mathbf{u}$ always explicitly available.

The following recurrences are equivalent to the ones in (3.1):

$$\begin{array}{rcll} \mathbf{r}_{k+1} & = & \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{u}_k & \perp \widetilde{\mathbf{r}}_k, \\ \mathbf{A}\mathbf{u}_{k+1} & = & \mathbf{A}\mathbf{r}_{k+1} - \beta_k \mathbf{A}\mathbf{u}_k & \perp \widetilde{\mathbf{r}}_k, \qquad \mathbf{u}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{u}_k. \end{array} \tag{3.2}$$

The first two updates in (3.2) require orthogonality to the same vector $\widetilde{\mathbf{r}}_k$. This common orthogonality property will turn out to be convenient for our approach. The vectors $\mathbf{A}\mathbf{u}_k$ in (3.2) are obtained by recursive update rather than by MV as in the standard approach (3.1), where, first, $\mathbf{u}_{k+1}$ is computed by $\mathbf{u}_{k+1} = \mathbf{r}_{k+1} - \beta_k \mathbf{u}_k$ and then an MV is used to obtain $\mathbf{A}\mathbf{u}_k$. Nevertheless, this formulation does not save MVs: (3.2) requires the computation of $\mathbf{A}\mathbf{r}_{k+1}$. Actually, the coupled recurrences in (3.2) are slightly more expensive than the ones in (3.1): they require one vector update more per step (per MV). The third recurrence relation provides the update $\mathbf{u}_{k+1}$ for the approximate solution $\mathbf{x}_{k+1}$.

3.2. Hybrid Bi-CG. Since $\widetilde{\mathbf{r}}_k \in \mathcal{K}_{k+1}(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$, there are polynomials P_k of degree k (possibly with complex coefficients) such that $\widetilde{\mathbf{r}}_k = \overline{P_k}(\mathbf{A}^*)\widetilde{\mathbf{r}}_0$. For notational convenience, we write $\mathbf{P}_k \equiv P_k(\mathbf{A})$. $\mathbf{P}_k$ is a matrix: it is the value of the matrix polynomial P_k at $\mathbf{A}$. Multiplying the relations in (3.2) by $\mathbf{P}_k$ leads to

$$\begin{aligned} \mathbf{P}_k\mathbf{r}_{k+1} &= \mathbf{P}_k\mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{P}_k\mathbf{u}_k \qquad \perp \widetilde{\mathbf{r}}_0, \\ \mathbf{A}\mathbf{P}_k\mathbf{u}_{k+1} &= \mathbf{A}\mathbf{P}_k\mathbf{r}_{k+1} - \beta_k\mathbf{A}\mathbf{P}_k\mathbf{u}_k \perp \widetilde{\mathbf{r}}_0, \quad \mathbf{P}_k\mathbf{u}_{k+1} = \mathbf{P}_k\mathbf{r}_{k+1} - \beta_k\mathbf{P}_k\mathbf{u}_k. \end{aligned} \tag{3.3}$$

For the orthogonality, we used relations such as $\widetilde{\mathbf{r}}_k^*\mathbf{r}_{k+1} = (\overline{P_k}(\mathbf{A}^*)\widetilde{\mathbf{r}}_0)^*\mathbf{r}_{k+1} = \widetilde{\mathbf{r}}_0^*\mathbf{P}_k\mathbf{r}_{k+1}$.

If, for each k, $P_k(0) = 1$, then $\mathbf{r}'_k \equiv \mathbf{P}_k\mathbf{r}_{k+1}$ (and $\mathbf{r}_k^{\text{hybrid}} \equiv \mathbf{P}_k\mathbf{r}_k$) is a residual; i.e., $\mathbf{r}'_k = \mathbf{b} - \mathbf{A}\mathbf{x}'_k$ for some approximate solution $\mathbf{x}'_k$. The $\mathbf{x}'_k$ can be computed by recursive updating using the update vector $\alpha_k\mathbf{P}_k\mathbf{u}_k$. Note that (3.3) allows us to compute the residuals as $\mathbf{r}'_k$ without computing the individual factor $\mathbf{P}_k$ or $\mathbf{r}_k$. Actually, none of the individual factors will show up in algorithmic formulations of hybrid Bi-CG methods. Also note that $\mathbf{r}_k^{\text{hybrid}}$ belongs to the Sonneveld subspace $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{r}}_0)$. It is a primary residual, while $\mathbf{r}'_k$ is secondary.

The relations in (3.3) give a general scheme for hybrid Bi-CG methods. The scheme is slightly different from the standard one to accommodate for the common orthogonality to $\widetilde{\mathbf{r}}_0$. The strategy for selecting the polynomials P_k, or rather for selecting the update strategy for the polynomials, determines the method. There are a number of effective strategies; see e.g., [11], [14], [2], [6], [1], [18].

3.3. Bi-CGSTAB and BiCGstab(ℓ). In Bi-CGSTAB, the polynomials P_k satisfy the two-term relation $P_{k+1}(\lambda) = (1 - \omega_{k+1}\lambda)\,P_k(\lambda)$: it requires the selection of scalars ω_{k+1}. Hence, $\mathbf{P}_{k+1} = (1 - \omega_{k+1}\mathbf{A})\,\mathbf{P}_k$ and

$$\begin{aligned} \mathbf{P}_{k+1}\mathbf{r}_{k+1} &= \mathbf{P}_k\mathbf{r}_{k+1} - \omega_{k+1}\mathbf{A}\mathbf{P}_k\mathbf{r}_{k+1}, \\ \mathbf{P}_{k+1}\mathbf{u}_{k+1} &= \mathbf{P}_k\mathbf{u}_{k+1} - \omega_{k+1}\mathbf{A}\mathbf{P}_k\mathbf{u}_{k+1}. \end{aligned} \tag{3.4}$$

A combination of these relations with the ones in (3.3) forms an update of the primary residual $\mathbf{r}_k^{\text{BiCGSTAB}} \equiv \mathbf{P}_k\mathbf{r}_k$ in $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{r}}_0)$ to the primary residual $\mathbf{r}_{k+1}^{\text{BiCGSTAB}}$ in $\mathcal{S}(P_{k+1}, \mathbf{A}, \widetilde{\mathbf{r}}_0)$. To complete the Bi-CGSTAB loop consisting of (3.3) and (3.4), $\mathbf{A}\mathbf{P}_{k+1}\mathbf{u}_{k+1}$ has to be computed. This can be done by multiplying $\mathbf{P}_{k+1}\mathbf{u}_{k+1}$ by $\mathbf{A}$, or by multiplying $\mathbf{A}\mathbf{P}_k\mathbf{u}_{k+1}$ by $\mathbf{A}$ after the updates in (3.3), followed by the update

$$\mathbf{A}\mathbf{P}_{k+1}\mathbf{u}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{u}_{k+1} - \omega_{k+1}\mathbf{A}^2\mathbf{P}_k\mathbf{u}_{k+1}. \tag{3.5}$$

This last approach makes each loop one vector update more expensive: the number of MVs (two per loop) is not affected, but this approach fits best to the one in sections 4–5.

Bi-CGSTAB selects ω_{k+1} to minimize the norm of the right-hand side of the first expression in (3.4). Note that this minimizing scalar is real if $\mathbf{A}$ is real and the residuals are real. BiCGstab(ℓ) can be described by stating that it selects ω_j in $\mathbb{C}$ such that at the end of each cycle of ℓ of these loops, the norm of the expression

$$\mathbf{P}_k\mathbf{r}_{k+\ell} - \gamma_{1,k}\mathbf{A}\mathbf{P}_k\mathbf{r}_{k+\ell} - \cdots - \gamma_{\ell,k}\mathbf{A}^\ell\mathbf{P}_k\mathbf{r}_{k+\ell} \tag{3.6}$$

is minimal with respect to the scalars $\gamma_{i,k}$ and $\mathbf{P}_{k+\ell}\mathbf{r}_{k+\ell}$ equals this expression. Here,

$$1 - \gamma_{1,k}\lambda - \cdots - \gamma_{\ell,k}\lambda^\ell = (1 - \omega_{k+1}\lambda)\cdot\cdots\cdot(1 - \omega_{k+\ell}\lambda) \qquad (\lambda \in \mathbb{C}),$$

Select an initial guess $\mathbf{x}$ and an $\widetilde{\mathbf{r}}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \star \mathbf{x}$
$\mathbf{r} = [\mathbf{r}_0]$, $\mathbf{u} = [\mathbf{r}_0; \mathbf{A} \star \mathbf{r}_0]$
while $\|\mathbf{r}_0\| > tol$
  for $j = 1, \ldots, \ell$
    % *The Bi-CG step*
    $\sigma = \widetilde{\mathbf{r}}_0^* \mathbf{u}_j$, $\alpha = \sigma^{-1}(\widetilde{\mathbf{r}}_0^* \mathbf{r}_{j-1})$
    $\mathbf{x} = \mathbf{x} + \mathbf{u}_0 \alpha$, $\mathbf{r} = \mathbf{r} - [\mathbf{u}_1; \ldots; \mathbf{u}_j] \alpha$, $\mathbf{r} = [\mathbf{r}; \mathbf{A} \star \mathbf{r}_{j-1}]$
    $\beta = \sigma^{-1}(\widetilde{\mathbf{r}}_0^* \mathbf{r}_j)$, $\mathbf{u} = \mathbf{r} - \mathbf{u} \beta$, $\mathbf{u} = [\mathbf{u}; \mathbf{A} \star \mathbf{u}_j]$
  end for
  % *The polynomial step*
  $\gamma = [\gamma_1; \ldots; \gamma_\ell] = \mathtt{argmin}_\gamma \|\mathbf{r}_0 - [\mathbf{r}_1, \ldots, \mathbf{r}_\ell] \gamma\|$
  $\mathbf{x} = \mathbf{x} + [\mathbf{r}_0, \ldots, \mathbf{r}_{\ell-1}] \gamma$, $\mathbf{r} = \mathbf{r}_0 - [\mathbf{r}_1, \ldots, \mathbf{r}_\ell] \gamma$
  $\mathbf{u} = [\mathbf{u}_0 - \sum_{j=1}^{\ell} \gamma_j \mathbf{u}_j; \mathbf{u}_1 - \sum_{j=1}^{\ell} \gamma_j \mathbf{u}_{j+1}]$
end while

ALGORITHM 3.1. BiCGstab(ℓ). *The $\mathbf{u}_j$ and $\mathbf{r}_j$ in the computation of the scalars σ, α, and β, respectively, are related to $\mathbf{u}$, and $\mathbf{r}$ according to $\mathbf{u} = [\mathbf{u}_0; \ldots; \mathbf{u}_j]$ and $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_j]$, respectively. MVs are denoted by statements like $\mathbf{A} \star \mathbf{v}$. Note that the sizes of $\mathbf{r}$ and $\mathbf{u}$ change during the loop: at the start of the jth step of the "*for $j = \ldots$*" loop, $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_{j-1}]$ and $\mathbf{u} = [\mathbf{u}_0; \ldots; \mathbf{u}_j]$. γ is an ℓ-vector.*

relates the γ's and ω's. Note that k is a multiple of ℓ. In actual computations, the γ's are computed. Since they can only be computed at the end of the cycle of ℓ loops, the ω's are not available in intermediate loops. Therefore, to make this ℓth-degree minimization applicable, the cycle of ℓ loops needs some rearrangement as is done in [6] in the derivation of a BiCGstab(ℓ) algorithm. For completeness, we include the resulting algorithm; see Algorithm 3.1. This variant is equivalent to the original one in [6]. But, as explained above, in contrast to the original version, the new one explicitly puts vectors orthogonal to $\widetilde{\mathbf{r}}_0$ (see (3.3)).

The polynomials P_k in CGS [11], GCGS [1], and GPBiCG [18] are generated by three term recurrences $\beta_k P_{k+1}(\lambda) = (\lambda - \alpha_k) P_k(\lambda) - \beta_{k-1} P_{k-1}(\lambda)$ (with $\beta_k + \alpha_k + \beta_{k-1} = 1$ to have "residual" polynomials P_k; i.e., $P_k(0) = 1$). Therefore, although P_k can be expressed as $P_k(\lambda) = \Pi_{j \le k}(1 - \omega_j \lambda)$, these hybrid variants cannot be described in the above Bi-CGSTAB fashion: the ω_j's ($j \le k$) change as k increases ($\omega_j = \omega_{j,k}$).

4. IDR. In this section, we formulate a variant of IDR that, in section 5, allows easy incorporation of degree ℓ polynomial minimization for $\ell > 1$. In this variant, we distinguish two parts in one cycle of the method.

Suppose that at the beginning of the kth cycle, $n \times s$ "update" matrices $\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_s]$ and $\mathbf{AU}_k \equiv [\mathbf{Au}_1, \ldots, \mathbf{Au}_s]$ are available. In addition, we also have an approximate solution $\mathbf{x}_k$ and its associated residual $\mathbf{r}_k \equiv \mathbf{b} - \mathbf{Ax}_k$ (assuming exact arithmetic). *(For $s = 1$, the $\mathbf{r}_k$ and $\mathbf{U}_k$ correspond to the $\mathbf{P}_k \mathbf{r}_k$ and $\mathbf{P}_k \mathbf{u}_k$, respectively, of* (3.3)*. Note that, in this section, the notation of $\mathbf{r}_k$ and $\mathbf{U}_k$ does show the polynomial dependence. References to corresponding formulas in section* 3 *are italic.)*

- In the first part (see section 4.1), which we call *the IDR step*, we update the residual $\mathbf{r}_k$ in $\mathcal{G}_k$ (a primary residual) and the $n \times s$ matrix $\mathbf{AU}_k$ with columns

also in $\mathcal{G}_k$ to a (secondary) residual $\mathbf{r}'$ in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$ and an $n \times s$ matrix $\mathbf{AV}$ with columns in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$.

- Then in the second part (see section 4.2), *the polynomial part*, we select a degree one polynomial factor $Q(\lambda) \equiv 1 - \omega_{k+1}\lambda$ and update $\mathbf{r}'$ to the (next primary) residual $\mathbf{r}_{k+1} \equiv \mathbf{r}' - \omega_{k+1}\mathbf{A}\mathbf{r}'$ in $\mathcal{G}_{k+1} \equiv (\mathbf{I} - \omega_{k+1}\mathbf{A})(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp})$ and $\mathbf{AV}$ to $\mathbf{AU}_{k+1}$ with columns also in $\mathcal{G}_{k+1}$. $\mathbf{V}$ is updated to $\mathbf{U}_{k+1}$.

Although the structure of the algorithm that we describe here seems similar to the ones described in [12] and [15], there is an essential difference. The IDR(s) algorithms given in [12] and [15] compute in the "IDR step" vectors in $\mathcal{G}_k$, which requires multiplication with the polynomial factor $(\mathbf{I} - \omega_k\mathbf{A})$. The "polynomial step" consists of the computation of a new polynomial factor $(\mathbf{I} - \omega_{k+1}\mathbf{A})$ (i.e., the computation of a new value for ω) and of the first residual in $\mathcal{G}_{k+1}$. In the algorithm we describe here, the IDR step involves computation of vectors in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$, which only requires multiplications with $\mathbf{A}$. The update of the vectors in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$ to vectors in $\mathcal{G}_{k+1}$ is done in the polynomial step. The fact that the minimization polynomial is only applied explicitly in the polynomial step of the algorithm makes the extension to higher order minimization polynomials natural.

4.1. The IDR step. Let Π_i be the projections defined by

$$\Pi_i \equiv \mathbf{I} - \mathbf{A}^i\mathbf{U}_k\sigma^{-1}\widetilde{\mathbf{R}}_0^*\mathbf{A}^{1-i} \quad \text{with} \quad \sigma \equiv \widetilde{\mathbf{R}}_0^*\mathbf{A}\mathbf{U}_k \quad (i = 0, 1). \tag{4.1}$$

Note that

$$\widetilde{\mathbf{R}}_0^*\Pi_1 = \mathbf{0} \quad \text{and} \quad \Pi_1\mathbf{A} = \mathbf{A}\Pi_0. \tag{4.2}$$

In particular, we have that $\Pi_1\mathbf{w} \perp \widetilde{\mathbf{R}}_0$ for all n-vectors $\mathbf{w}$.

With $\rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{r}_k$ and $\alpha \equiv \sigma^{-1}\rho$, we generate $\mathbf{r}'$ and $\mathbf{x}'$ by *(cf.* (3.3)*)*

$$\mathbf{r}' \equiv \Pi_1\mathbf{r}_k = \mathbf{r}_k - \mathbf{A}\mathbf{U}_k\alpha \quad \text{and} \quad \mathbf{x}' \equiv \mathbf{x}_k + \mathbf{U}_k\alpha. \tag{4.3}$$

Then we have that $\mathbf{b} - \mathbf{A}\mathbf{x}' = \mathbf{r}' \perp \widetilde{\mathbf{R}}_0$. Now, we obtain the vector $\mathbf{A}\mathbf{r}'$ by matrix multiplication.

Next, we update $\mathbf{U}_k$ to $\mathbf{V}$, and $\mathbf{AU}_k$ to $\mathbf{AV}$ such that $\mathbf{AV}$ is orthogonal to $\widetilde{\mathbf{R}}_0$ *(cf.* (3.3)*)* and we compute $\mathbf{A}^2\mathbf{V}$. We generate $\mathbf{AV}$ such that the columns form a basis of the Krylov subspace $\mathcal{K}_s(\Pi_1\mathbf{A}, \Pi_1\mathbf{A}\mathbf{r}')$. As a "side product" we obtain $\mathbf{A}^2\mathbf{V}$ and $\mathbf{V}$. For instance (for variants, see section 4.4), if

$$\mathbf{AV} = [\Pi_1\mathbf{A}\mathbf{r}', (\Pi_1\mathbf{A})^2\mathbf{r}', \ldots, (\Pi_1\mathbf{A})^s\mathbf{r}'] \perp \widetilde{\mathbf{R}}_0, \tag{4.4}$$

then, with $\mathbf{s} \equiv \mathbf{AV}e_q$ for some $q < s$, the vector $\mathbf{AV}e_{q+1}$ can be computed as $\mathbf{c} \equiv \mathbf{A}\mathbf{s}$, which gives the qth column of $\mathbf{A}^2\mathbf{V}$,

$$\mathbf{A}^2\mathbf{V}e_q = \mathbf{c}, \quad \text{where} \quad \mathbf{c} \equiv \mathbf{A}\mathbf{s}, \tag{4.5}$$

followed by the projection:

$$\mathbf{AV}e_{q+1} = \mathbf{c} - \mathbf{A}\mathbf{U}_k\beta, \quad \text{where} \quad \beta \equiv \sigma^{-1}\rho \quad \text{and} \quad \rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{c}. \tag{4.6}$$

Now, the $(q+1)$th column of $\mathbf{V}$ can be computed as

$$\mathbf{V}e_{q+1} = \mathbf{s} - \mathbf{U}_k\beta. \tag{4.7}$$

(The vector $\mathbf{c}$ *relates to residuals as we will indicate in section* 4.4.3*, and* (4.6) *corresponds to the second relation in* (3.3) *while* (4.7) *corresponds to the third relation.)*

The following scheme summarizes this IDR step

$$
\begin{array}{ccccccccccc}
\mathbf{U}_k & & \mathbf{x}_k & \to & \mathbf{x}' & & \mathbf{V}e_1 & & \mathbf{V}e_2 & \dots & \mathbf{V}e_s \\
 & & & & & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{AU}_k & & \mathbf{r}_k & \overset{\Pi_1}{\to} & \mathbf{r}' & & \mathbf{AV}e_1 & & \mathbf{AV}e_2 & \dots & \mathbf{AV}e_s \\
 & & & & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
 & & & & \boxed{\mathbf{A}\mathbf{r}'} & & \boxed{\mathbf{A}^2\mathbf{V}e_1} & & \boxed{\mathbf{A}^2\mathbf{V}e_2} & \dots & \boxed{\mathbf{A}^2\mathbf{V}e_s}
\end{array}
\tag{4.8}
$$

The boxed vectors have been obtained by explicit multiplication by the matrix $\mathbf{A}$. The other "new" vectors have been obtained by vector updating as indicated in (4.3), (4.6), and (4.7).

The new vectors on the second row of vectors in scheme (4.8), the vectors $\mathbf{r}', \mathbf{AV}e_1, \dots, \mathbf{AV}e_s$, are orthogonal to $\widetilde{\mathbf{R}}_0$. To use IDR terminology, if $\mathbf{r}_k$ and $\mathbf{AU}_k$ belong to $\mathcal{G}_k$, then (see Lemma 4.1 below) these new vectors belong to $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$. Multiplication by $\mathbf{A}$ leads to the vectors on the last row of (4.8). What $\mathcal{G}_{k+1}$ will be depends on the choice of μ_{k+1}: $\mathcal{G}_{k+1} \equiv (\mu_{k+1}\mathbf{I} - \mathbf{A})(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp})$. Hence, μ_{k+1} times a vector on the second row plus the corresponding vector on the third row belongs to $\mathcal{G}_{k+1}$ (for general μ_{k+1}). In the next subsection, we discuss the choice $\mu_{k+1} = 1/\omega_{k+1}$.

LEMMA 4.1. *Assume* $\mathrm{span}(\mathbf{AU}_k, \mathbf{r}_k) \subset \mathcal{G}_k$. *Then* $\mathrm{span}(\mathbf{AV}, \mathbf{r}') \subset \mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$.

Proof. By definition of Π_1 and (4.2), $\Pi_1\mathbf{t} \in \mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$ if $\mathbf{t} \in \mathcal{G}_k$. In particular, $\mathbf{r}' \in \mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$. Consider an $\mathbf{s} \in \mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$. Apply (1) of Theorem 2.1 (with $\mu_{k+1} = 0$) to see that $\mathbf{A}(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}) \subset \mathcal{G}_k$. Hence, $(\mathbf{t} \equiv)\mathbf{As} \in \mathcal{G}_k$ and $\Pi_1\mathbf{As} \in \mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$. Taking $\mathbf{s} = \mathbf{r}'$ and $\mathbf{s} = \mathbf{AV}e_q$ for $q = 1, \dots, s$ completes the proof. ∎

4.2. The polynomial step. Now, we have the ingredients to perform the minimization step to obtain our next approximate solution, residual, and update matrices, denoted by $\mathbf{x}_{k+1}$, $\mathbf{r}_{k+1}$, $\mathbf{U}_{k+1}$, and $\mathbf{AU}_{k+1}$, respectively.

Select a complex scalar ω_{k+1}. Then

$$
\begin{aligned}
&\mathbf{x}_{k+1} \equiv \mathbf{x}' + \omega_{k+1}\mathbf{r}', \quad \mathbf{r}_{k+1} \equiv \mathbf{r}' - \omega_{k+1}\mathbf{A}\mathbf{r}', \\
&\mathbf{U}_{k+1} \equiv \mathbf{V} - \omega_{k+1}\mathbf{AV}, \quad \mathbf{AU}_{k+1} \equiv \mathbf{AV} - \omega_{k+1}\mathbf{A}^2\mathbf{V}
\end{aligned}
$$

(cf. (3.4) *and* (3.5)*).*

This step can be viewed as a Richardson step with parameter ω_{k+1}. In practice, ω_{k+1} is selected to minimize the norm of the new residual $\mathbf{r}_{k+1}$ and then the step can be viewed as one step in restarted GMRES with restart every step (GMRES(1)). But other choices are possible as well: instead, for minimal residuals, one can also aim for accurate IDR updates similar to the strategy for Bi-CGSTAB as explained in [8].

Note 4.2. If, in scheme (4.8), the column with $\mathbf{x}'$, $\mathbf{r}'$, and $\mathbf{A}\mathbf{r}'$ is replaced by $\mathbf{x}_{k+1}$ and $\mathbf{r}_{k+1}$ ($\mathbf{r}_{k+1}$ at the location of $\mathbf{A}\mathbf{r}'$), and a similar replacement for $\mathbf{AV}_k$ is included, then the scheme for the subsequential steps can be chained. The horizontal movement (from left to right) can be viewed as representing the update of the order k of the shadow Krylov subspace $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$, while the vertical movement (from top to bottom) represents the increase of the polynomial degree (cf. [5], [4]).

```
Select an initial guess x and an n × s matrix R̃_0.
Compute r = b − A ⋆ x
% Generate an initial U; AU
for q = 1, ..., s
   if q = 1, v_0 = r, else, v_0 = v_1
   v_1 = A ⋆ v_0
   U_(:,q) = v_0,  AU_(:,q) = v_1
end for
while ‖r‖ > tol
    % The IDR step
    σ = R̃_0^* AU, α = σ^{-1}(R̃_0^* r)
    x = x + Uα, r = r − AUα, Ar = A ⋆ r
    for q = 1, ..., s
       if q = 1, v_0 = r, v_1 = Ar, else, v_0 = v_1,  v_1 = v_2
       β = σ^{-1}(R̃_0^* v_1),  v_0 = v_0 − Uβ, v_1 = v_1 − AUβ, v_2 = A ⋆ v_1
       V_(:,q) = v_0, AV_(:,q) = v_1, A^2 V_(:,q) = v_2;
    end for
    % The polynomial step
    ω = argmin_ω ‖r − ωAr‖
    x = x + ωr,  r = r − ωAr
    U = V − ωAV, AU = AV − ωA^2 V
end while
```

ALGORITHM 4.2. IDR. *The* $\mathbf{A}$ *in quantities as* $\mathbf{AU}$ *is "part of the name." MVs are indicated by a "*$\star$*" as, for example,* $\mathbf{v}_1 = \mathbf{A} \star \mathbf{v}_0$ *(line* 6*).*

Algorithm 4.2 summarizes the above. In the algorithm, we replace old quantities by the corresponding new ones whenever possible. In particular, there is no reference to the step number k and $\mathbf{x}'$ is absorbed by $\mathbf{x}$ and $\mathbf{r}'$ by $\mathbf{r}$. This basic algorithm can be improved in several ways as we will indicate in section 4.4.

4.3. The relation to IDR. A combination of Theorem 2.4 and the following proposition implies that the above approach defines an IDR method.

PROPOSITION 4.3. *Consider one cycle of the method as described above in sections* 4.1 *and* 4.2*. Suppose that at the beginning of the cycle the residual* $\mathbf{r}_k$ *as well as the columns of the matrix* $\mathbf{AU}_k$ *belong to a Sonneveld subspace* $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$ *with* P_k *a polynomial of exact degree* k*. Then, with* $P_{k+1}(\lambda) \equiv (1 - \omega_{k+1}\lambda)P_k(\lambda)$*, the residual* $\mathbf{r}_{k+1}$ *and the columns of* $\mathbf{AU}_{k+1}$ *at the end of the cycle belong to* $\mathcal{S}(P_{k+1}, \mathbf{A}, \widetilde{\mathbf{R}}_0)$*.*

Proof. Combine Theorem 2.4 with the argument at the end of section 4.1. ∎

4.4. Alternative formulations of the IDR step. In (4.8) above, we constructed the basis vectors of $\mathcal{AU} \equiv \mathcal{K}_s(\Pi_1\mathbf{A}, \Pi_1\mathbf{Ar}')$ ($\mathbf{AV}$ is the matrix of basis vectors) by multiplication by the operator $\Pi_1\mathbf{A}$, thus generating a "power basis." With larger s, this approach will be unstable, and a more stable approach should be used such as Arnoldi's method for generating an orthonormal set of basis vectors. Then the matrix of basis vectors can be expressed as $\mathbf{AV}T$, with T some nonsingular $s \times s$ matrix. For instance, with Arnoldi's, we find the $n \times s$ orthonormal matrix $\mathbf{Q}$ from the QR-decomposition of $\mathbf{AV}$: $\mathbf{AV} = \mathbf{Q}R$, the $s \times s$ matrix R is upper triangular,

and $T = R^{-1}$. Herewith associated, we should compute $\widetilde{\mathbf{V}} \equiv \mathbf{V}T$ and $\mathbf{A}^2\widetilde{\mathbf{V}} \equiv \mathbf{A}^2\mathbf{V}T$ instead of $\mathbf{V}$ and $\mathbf{A}^2\mathbf{V}$, respectively. Then $\mathbf{Q} = \mathbf{A}\widetilde{\mathbf{V}}$.

In practice, whenever we compute $\mathbf{A}\widetilde{\mathbf{V}}e_q$ by $\sum_{j<q} \gamma_j \mathbf{A}\widetilde{\mathbf{V}}e_j + \gamma_q \mathbf{A}\mathbf{V}e_q$, we immediately compute $\mathbf{A}^i\widetilde{\mathbf{V}}e_q$ (for $i = 0, 2$) by $\sum_{j<q} \gamma_j \mathbf{A}^i\widetilde{\mathbf{V}}e_j + \gamma_q \mathbf{A}^i\mathbf{V}e_q$, and we put the new values at the location of the old ones.

Below we discuss three stable approaches for generating a set of basis vectors.

4.4.1. Orthogonalization. In this approach, Arnoldi's approach, $\mathbf{A}\widetilde{\mathbf{V}}$ is orthonormal: $\mathbf{A}\widetilde{\mathbf{V}}e_{q+1}$ is obtained by orthogonalizing $\Pi_1\mathbf{A}^2\widetilde{\mathbf{V}}e_q$ against $\mathbf{A}\widetilde{\mathbf{V}}_{(:,1:q)}$. Note that $\mathbf{A}^2\widetilde{\mathbf{V}}e_q$ is obtained by multiplying $\mathbf{A}\widetilde{\mathbf{V}}e_q$ by $\mathbf{A}$. The alternative of orthogonalizing $\mathbf{A}\Pi_1\mathbf{A}^2\widetilde{\mathbf{V}}e_q$ against $\mathbf{A}^2\widetilde{\mathbf{V}}_{(:,1:q)}$ is comparably stable, but is slightly more expensive, since now the computation of $\mathbf{A}^2\widetilde{\mathbf{V}}e_q$ requires vector updates as well.

4.4.2. Biorthogonalization. The columns of $\mathbf{A}^2\widetilde{\mathbf{V}}$ and $\widetilde{\mathbf{R}}_0$ satisfy a biorthogonality relation: $\widetilde{\sigma} \equiv \widetilde{\mathbf{R}}_0^*\mathbf{A}^2\widetilde{\mathbf{V}}$ is lower triangular. The vector $\mathbf{A}^2\widetilde{\mathbf{V}}e_{q+1}$ is obtained by (bi)orthogonalizing $\mathbf{s} \equiv \mathbf{A}\Pi_1\mathbf{A}^2\widetilde{\mathbf{V}}e_q$ against $\widetilde{\mathbf{R}}_{0(:,1:q)}$ with $\mathbf{W} \equiv \mathbf{A}^2\widetilde{\mathbf{V}}_{(:,1:q)}$: $\mathbf{A}^2\widetilde{\mathbf{V}}e_{q+1} = (\mathbf{I} - \mathbf{W}\widetilde{\sigma}_q^{-1}\widetilde{\mathbf{R}}^*_{0(:,1:q)})\mathbf{s}$, where $\widetilde{\sigma}_q \equiv \widetilde{\mathbf{R}}^*_{0(:,1:q)}\mathbf{W}$. The matrix $\widetilde{\sigma}_q$ is $q \times q$ lower triangular; it is the left upper block of $\widetilde{\sigma}$.

This approach appears to be stable and is quite efficient: for instance, the $\widetilde{\sigma}$ is the σ of the next IDR step. This variant is similar to the one described in [15]. For details, we refer to this reference.

4.4.3. Projected systems. Krylov subspace methods, such as ORTHODIR and GCR, for solving the projected system

$$\Pi_1\mathbf{A}\mathbf{y} = \mathbf{r}' \tag{4.9}$$

(with initial guess $\mathbf{0}$ or, equivalently, to the system $\Pi_1\mathbf{A}\mathbf{y} = \mathbf{b}$ with initial guess $\mathbf{x}$) produce a (stable) set of basis vectors for $\mathcal{AU}$ as well. These methods can also be used. As a side product they allow termination before finishing the IDR step: if, say, the GCR residual is small enough for $q < s$, then the GCR approximation for $\mathbf{y}$ is an appropriate update for $\mathbf{x}$. For a further discussion we refer to [10, section 4.4].

4.4.4. Saving storage. Note that the ω_{k+1} can be determined after the computation of $\mathbf{r}'$ and $\mathbf{A}\mathbf{r}'$, but before the formation of $\mathbf{V}e_1$: $\mathbf{x}_{k+1}$ and $\mathbf{r}_{k+1}$ can be computed before forming $\mathbf{V}e_2$ and $\mathbf{A}\mathbf{V}e_2$. Similarly, $\mathbf{U}_{k+1}e_q$ and $\mathbf{A}\mathbf{U}_{k+1}e_q$ can be computed before forming $\mathbf{V}e_{q+2}$ and $\mathbf{A}\mathbf{V}e_{q+2}$. These vectors can be stored at the location of storage of $\mathbf{V}e_q$ and $\mathbf{A}\mathbf{V}e_q$. This saves the storage of an $n \times s$ matrix (the matrix $\mathbf{A}^2\mathbf{V}$).

4.4.5. Saving vector updates. For the computation of σ (cf. (4.1)), the matrix $\mathbf{A}\mathbf{U}_k$ is not needed: σ can be computed as $(\mathbf{A}^*\widetilde{\mathbf{R}}_0)^*\mathbf{U}_k$. The matrix $\mathbf{A}^*\widetilde{\mathbf{R}}_0$ has to be computed once and stored. This allows us to compute $\mathbf{r}'$, $\mathbf{V}$, and $\mathbf{A}\mathbf{V}$ without $\mathbf{A}\mathbf{U}_k$ and without computing $\mathbf{A}^2\mathbf{V}$ in the IDR step. This saves storage, as well as vector updates. For details, see [10, section 4.5].

The approaches in sections 4.4.2 and 4.4.4–4.4.5 can be combined. However, we will not elaborate on this here; it will be the subject of a future publication.

5. IDRstab. As explained in section 4.2, after computation of $\mathbf{x}'$, $\mathbf{r}'$, $\mathbf{A}\mathbf{r}'$, $\mathbf{V}$, $\mathbf{A}\mathbf{V}$, and $\mathbf{A}^2\mathbf{V}$, a scalar ω_{k+1} can be determined and the vectors can be updated to $\mathbf{x}_{k+1}$, $\mathbf{r}_{k+1} = \mathbf{r}' - \omega_k\mathbf{A}\mathbf{r}'$, $\mathbf{U}_{k+1} = \mathbf{V} - \omega_{k+1}\mathbf{A}\mathbf{V}$, etc. As an alternative, the IDR step can be repeated ℓ times before selecting appropriate scalars ω_{k+1} (see section 5.1 and Lemma 5.2 below). This leads to a BiCGstab(ℓ) version of IDR (see section 5.2). We will refer to this variant as IDR(s)stab(ℓ), or IDRstab for short.

5.1. The IDR step that allows degree ℓ minimization. We give details on how to "repeat" an IDR step without polynomial updates.

Performing $j-1$ repetitions makes a residual $\mathbf{r}$, say, available with associated approximation $\mathbf{x}$, plus the vectors $\mathbf{Ar}$, ..., $\mathbf{A}^{j-1}\mathbf{r}$, and the $n \times s$ matrices $\mathbf{U}$, $\mathbf{AU}$, and $\mathbf{A}^2\mathbf{U}$, ..., $\mathbf{A}^j\mathbf{U}$. If $j=1$, then $\mathbf{r}=\mathbf{r}_k$, $\mathbf{x}=\mathbf{x}_k$, and $\mathbf{U}=\mathbf{U}_k$ (with k an integer multiple of ℓ).

The next "repetition step" can be described by the projections Π_i defined by

$$\Pi_i \equiv \mathbf{I} - \mathbf{A}^i\,\mathbf{U}\sigma^{-1}\widetilde{\mathbf{R}}_0^*\mathbf{A}^{j-i} \quad (i=0,1,\ldots,j) \quad \text{with} \quad \sigma \equiv \widetilde{\mathbf{R}}_0^*\mathbf{A}^j\mathbf{U}. \tag{5.1}$$

Note that

$$\widetilde{\mathbf{R}}_0^*\Pi_j = \mathbf{0} \quad \text{and} \quad \Pi_{i+1}\mathbf{A} = \mathbf{A}\Pi_i \quad (i=0,1,\ldots,j-1). \tag{5.2}$$

First, we obtain $\mathbf{A}^i\mathbf{r}'$ for $i<j$ by projection: with $\alpha \equiv \sigma^{-1}(\widetilde{\mathbf{R}}_0^*\mathbf{A}^{j-1}\mathbf{r})$,

$$\mathbf{A}^i\mathbf{r}' \equiv \Pi_{i+1}(\mathbf{A}^i\mathbf{r}) = \mathbf{A}^i\mathbf{r} - \mathbf{A}^{i+1}\mathbf{U}\alpha \quad (i<j), \quad \mathbf{x}' = \mathbf{x} - \mathbf{U}\alpha. \tag{5.3}$$

Next, we obtain $\mathbf{A}^j\mathbf{r}'$ by multiplying $\mathbf{A}^{j-1}\mathbf{r}'$ by $\mathbf{A}$:

$$\mathbf{A}^j\mathbf{r}' = \mathbf{A}(\mathbf{A}^{j-1}\mathbf{r}'). \tag{5.4}$$

Then we update $\mathbf{A}^i\mathbf{U}$ to $\mathbf{A}^i\mathbf{V}$ $(i=0,\ldots,j)$ such that the columns of $\mathbf{A}^j\mathbf{V}$ are orthogonal to $\widetilde{\mathbf{R}}_0$ and form a basis of the Krylov subspace $\mathcal{K}_s(\Pi_j\mathbf{A}, \Pi_j\mathbf{A}^j\mathbf{r}')$. As a side product we obtain $\mathbf{A}^i\mathbf{V}$ for $i=0,\ldots,j-1,j+1$. For instance, with $\mathbf{s}_j \equiv \mathbf{A}^j\mathbf{V}e_q$ for some $q<s$, the vector $\mathbf{A}^j\mathbf{V}e_{q+1}$ can be computed as $\mathbf{As}_j$, which gives the qth column of $\mathbf{A}^{j+1}\mathbf{V}$, followed by the projection Π_j: $\mathbf{A}^j\mathbf{V}e_{q+1} \equiv \mathbf{As}_j - \mathbf{A}^j\mathbf{U}\beta$, where $\beta \equiv \sigma^{-1}\rho$ and $\rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{As}_j$. Then the $(q+1)$th column of $\mathbf{A}^i\mathbf{V}$ can be computed as $\mathbf{A}^i\mathbf{V}e_{q+1} = \mathbf{s}_{i+1} - \mathbf{A}^i\mathbf{U}\beta$ $(i=0,\ldots,j-1)$, involving vector updates only (no additional MVs, no additional dot products). Here, $\mathbf{s}_i \equiv \mathbf{A}^i\mathbf{V}e_q$. The following scheme summarizes this IDR step in case $j=2$:

$$\begin{array}{ccccccccccc}
\mathbf{U} & & \mathbf{x} & \to & \mathbf{x}' & & \mathbf{V}e_1 & & \mathbf{V}e_2 & \ldots & \mathbf{V}e_s \\
 & & & & & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{AU} & & \mathbf{r} & \overset{\Pi_1}{\to} & \mathbf{r}' & & \mathbf{AV}e_1 & & \mathbf{AV}e_2 & \ldots & \mathbf{AV}e_s \\
 & & & & & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{A}^2\mathbf{U} & & \mathbf{Ar} & \overset{\Pi_2}{\to} & \mathbf{Ar}' & & \mathbf{A}^2\mathbf{V}e_1 & & \mathbf{A}^2\mathbf{V}e_2 & \ldots & \mathbf{A}^2\mathbf{V}e_s \\
 & & & & \downarrow \mathbf{A} & \overset{\Pi_2}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_2}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
 & & & & \boxed{\mathbf{A}^2\mathbf{r}'} & & \boxed{\mathbf{A}^3\mathbf{V}e_1} & & \boxed{\mathbf{A}^3\mathbf{V}e_2} & \ldots & \boxed{\mathbf{A}^3\mathbf{V}e_s}
\end{array} \tag{5.5}$$

As before, the boxed vectors have been obtained by explicit multiplication by the matrix $\mathbf{A}$, while the other "new" vectors have been obtained by vector updates.

The new vectors on the next to last row of scheme (5.5)—the vectors $\mathbf{Ar}'$, $\mathbf{A}^2\mathbf{V}e_1$, ..., $\mathbf{A}^2\mathbf{V}e_s$—are orthogonal to $\widetilde{\mathbf{R}}_0$. Actually, all new vectors have this orthogonality property except for the ones at the last and at the first rows (see also Note 2.3).

LEMMA 5.1. *Assume* $\mathbf{AU}, \mathbf{r} \perp \mathcal{K}_{j-1}(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$. *Then* $\mathbf{AV}, \mathbf{r}' \perp \mathcal{K}_j(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$.

Proof. Consider an $\mathbf{s} \perp \mathcal{K}_j(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$. We will show that $\Pi_i \mathbf{A}^i \mathbf{s} \perp \widetilde{\mathbf{R}}_0$ and $\mathbf{A}^{i-1}\mathbf{r}' \perp \widetilde{\mathbf{R}}_0$ for $i = 1, \ldots, j$. Then an induction argument (increasing q with $\mathbf{s} = \mathbf{r}'$ and $\mathbf{s} = \mathbf{A}\mathbf{V}e_q$) completes the proof for $\mathbf{A}\mathbf{V}$.

Since $\Pi_i \mathbf{A}^i \mathbf{s} = \mathbf{A}^i \mathbf{s} - \mathbf{A}^i \mathbf{U} \sigma^{-1} \widetilde{\mathbf{R}}_0^* \mathbf{A}^j \mathbf{s}$, the assumption implies that $\Pi_i \mathbf{A}^i \mathbf{s} \perp \widetilde{\mathbf{R}}_0$ for $i = 1, \ldots, j-1$. Similarly, $\mathbf{A}^{i-1}\mathbf{r}' = \Pi_i \mathbf{A}^{i-1}\mathbf{r} \perp \widetilde{\mathbf{R}}_0$. The fact that $\widetilde{\mathbf{R}}_0^* \Pi_j = 0$ (see (5.2)) implies $\Pi_j \mathbf{A}^j \mathbf{s} \perp \widetilde{\mathbf{R}}_0$ and $\mathbf{A}^{j-1}\mathbf{r}' = \Pi_j \mathbf{A}^{j-1}\mathbf{r} \perp \widetilde{\mathbf{R}}_0$. ∎

The following lemma essentially states that the polynomial step can be postponed.

LEMMA 5.2. *Assume* $\mathbf{A}\mathbf{U}, \mathbf{r} \perp \mathcal{K}_{j-1}(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$. *For a polynomial* Q *of degree* $< j$, *put*

$$\widehat{\Pi}_1 \equiv \mathbf{I} - \mathbf{A}\mathbf{W}\widehat{\sigma}^{-1}\widetilde{\mathbf{R}}_0^* \quad \textit{with} \quad \mathbf{W} \equiv Q(\mathbf{A})\mathbf{U} \quad \textit{and} \quad \widehat{\sigma} \equiv \widetilde{\mathbf{R}}_0^* \mathbf{A}\mathbf{W}.$$

Then $\widehat{\Pi}_1 Q(\mathbf{A})\mathbf{s} = Q(\mathbf{A})\Pi_1 \mathbf{s}$ *for all* $\mathbf{s} \perp \mathcal{K}_{j-1}(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$. *In particular,* $\widehat{\Pi}_1 Q(\mathbf{A})\mathbf{r} = Q(\mathbf{A})\mathbf{r}'$ *and* $\widehat{\Pi}_1 \mathbf{A} Q(\mathbf{A})\mathbf{A}\mathbf{V}e_q = Q(\mathbf{A})\mathbf{A}\mathbf{V}e_{q+1}$.

Proof. The assumption implies that $\widehat{\sigma} \equiv \widetilde{\mathbf{R}}_0^* \mathbf{A} Q(\mathbf{A})\mathbf{U} = \gamma \widetilde{\mathbf{R}}_0^* \mathbf{A}^j \mathbf{U} = \gamma\sigma$, with γ the leading coefficient of Q. Further, if $\mathbf{s} \perp \mathcal{K}_{j-1}(\mathbf{A}^*, \widetilde{\mathbf{R}}_0)$, then $\widetilde{\mathbf{R}}_0^* Q(\mathbf{A})\mathbf{s} = \gamma \widetilde{\mathbf{R}}_0^* \mathbf{A}^{j-1}\mathbf{s}$ whence $\widehat{\Pi}_1 Q(\mathbf{A})\mathbf{s} = Q(\mathbf{A})\Pi_1 \mathbf{s}$.

The last claim follows by an induction argument (for q) and Lemma 5.1. ∎

To prepare for the next repetition of an IDR step, rename $\mathbf{x}'$ to $\mathbf{x}$, $\mathbf{A}^i\mathbf{r}'$ to $\mathbf{A}^i\mathbf{r}$ $(i = 0, \ldots, j)$, and $\mathbf{A}^i\mathbf{V}$ to $\mathbf{A}^i\mathbf{U}$ $(i = 0, \ldots, j+1)$.

The variants in section 4.4 have their analogues for the present situation. In our actual implementation for this paper, we followed Arnoldi's variant of section 4.4.1 to orthonormalize $\mathbf{A}^j\mathbf{V}$ (orthonormalizing the vectors at the next to last row of scheme (5.5)).

5.2. Minimization using polynomials of degree ℓ. After ℓ repetitions of the IDR step and before renaming, we have a residual $\mathbf{r}'$ available, plus the vectors $\mathbf{A}^i\mathbf{r}'$ for $i = 0, \ldots, \ell$, the associated approximation $\mathbf{x}'$, and $\ell+2$ matrices $\mathbf{A}^i\mathbf{V}$ $(i = 0, \ldots, \ell+1)$ of size $n \times s$. Now, to finish one cycle of IDRstab, determine scalars $\gamma_{1,k}, \ldots, \gamma_{\ell,k}$ such that the norm of

$$\mathbf{r}_{k+\ell} \equiv \mathbf{r}' - \gamma_{1,k}\mathbf{A}\mathbf{r}' - \cdots - \gamma_{\ell,k}\mathbf{A}^\ell \mathbf{r}' \tag{5.6}$$

is minimal. Compute

$$\mathbf{x}_{k+\ell} \equiv \mathbf{x}' + \gamma_{1,k}\mathbf{r}' + \cdots + \gamma_{\ell,k}\mathbf{A}^{\ell-1}\mathbf{r}' \tag{5.7}$$

and

$$\mathbf{A}^i\mathbf{U}_{k+\ell} \equiv \mathbf{A}^i\mathbf{V} - \gamma_{1,k}\mathbf{A}^{i+1}\mathbf{V} - \cdots - \gamma_{\ell,k}\mathbf{A}^{i+\ell}\mathbf{V} \quad (i = 0, 1). \tag{5.8}$$

5.3. The relation to IDR. Theorem 2.4 and the following theorem allows IDRstab to be viewed as an IDR method.

THEOREM 5.3. *Consider one cycle of IDRstab as described above in sections* 5.1 *and* 5.2. *Suppose none of the roots* $\lambda_1, \ldots, \lambda_\ell$ *of the polynomial*

$$Q_\ell(\lambda) \equiv 1 - \gamma_{1,k}\lambda - \cdots - \gamma_{\ell,k}\lambda^\ell \qquad (\lambda \in \mathbb{C})$$

is zero. Suppose at the beginning of the cycle the residual $\mathbf{r}_k$ *and the columns of the matrix* $\mathbf{A}\mathbf{U}_k$ *belong to the Sonneveld subspace* $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$, *with* P_k *a polynomial of*

exact degree k. Then the residual $\mathbf{r}_{k+\ell}$ and the columns of $\mathbf{A}\mathbf{U}_{k+\ell}$ at the end of the cycle belong to $\mathcal{S}(Q_\ell P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$: $P_{k+\ell} \equiv Q_\ell P_k$ is of exact degree $k+\ell$.

Proof. To prove the theorem, factorize the polynomial Q_ℓ as

$$Q_\ell(\lambda) = 1 - \gamma_{1,k}\lambda - \cdots - \gamma_{\ell,k}\lambda^\ell = (1 - \omega_{k+1}\lambda) \cdot \cdots \cdot (1 - \omega_{k+\ell}\lambda).$$

Then, with $Q_0 \equiv 1$ and $Q_j(\lambda) \equiv (1 - \omega_{k+j-1}\lambda)Q_{j-1}(\lambda)$, use induction to j, $j = 1, \ldots, \ell$, adding one factor per induction step with the IDR strategy of sections 4.1 and 4.2: from Proposition 4.3 we learn that the resulting residual $\widehat{\mathbf{r}}_{k+\ell}$, say, will be in $\mathcal{S}(Q_\ell P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$. From Lemma 5.2 we learn that $\widehat{\mathbf{r}}_{k+\ell}$ is precisely $\mathbf{r}_{k+\ell}$. A similar statement holds for $\mathbf{A}\mathbf{U}_{k+\ell}$ and, therefore, for $\mathbf{U}_{k+\ell}$ (using that $\mathbf{A}$ is nonsingular). ∎

We actually proved that the quantities $\mathbf{x}_{k+\ell}$, $\mathbf{r}_{k+\ell}$, and $\mathbf{U}_{k+\ell}$ (of (5.6)–(5.8)) would also have been obtained by performing ℓ steps of the IDR variant of section 4, provided the scalars ω_{k+j} would have been appropriately selected in each polynomial step (cf. section 4.2). Obviously, it is impossible in practice to select the ω_{k+j} "appropriately" at step $k+j$, $j < \ell$: the roots of the minimizing polynomial Q are not known at this stage. In particular, in one cycle of IDRstab, we move from the primary residual $\mathbf{r}_k$ to the primary residual $\mathbf{r}_{k+\ell}$ and do not explicitly compute the intermediate primary residual $\mathbf{r}_{k+j}$ ($j = 1, \ldots, \ell-1$). Nevertheless, the theorem is of interest since it shows that IDRstab is an IDR method but with more effective ω_{k+j} selection than in standard IDR.

5.4. The IDRstab algorithm. The procedure described in sections 5.1 and 5.2 leads to Algorithm 5.3. The notation in this algorithm follows the MATLAB conventions of Notation 1.3. The $\mathbf{r}_i$, $\mathbf{U}_i$, and $\mathbf{V}_i$ correspond to the $\mathbf{A}^i\mathbf{r}$, $\mathbf{A}^i\mathbf{U}$, and $\mathbf{A}^i\mathbf{V}$, respectively, of section 5.1.

With $\ell = 1$, this algorithm is (mathematically) equivalent to IDR(s), and with $s = 1$, we have BiCGstab(ℓ).

The lines in the two "`for` $q = \ldots$" loops involving μ form an implementation of Arnoldi's procedure to obtain an orthonormalized matrix $\mathbf{U}_0$ and $\mathbf{V}_j$, respectively. We use Arnoldi's since it is more stable than the power basis. If the power method is sufficiently stable (for instance, if $s < 3$), then these lines can be skipped. However, for larger s, a more stable form of the Gram–Schmidt method, as modified Gram–Schmidt or repeated Gram–Schmidt, may be required. We have good results with $s \leq 8$, and in our experience, for this size of s classical Gram–Schmidt is sufficiently stable.

Actually, Arnoldi's may be applied to obtain an orthonormalized matrix $\mathbf{V}_i (= \mathbf{A}^i\mathbf{V})$ for any $i = 0, \ldots, j+1$. For a further discussion, see [10, section 5.4].

As in BiCGstab(ℓ), the γ can be computed as the solution of the normal equation

$$(\mathbf{R}^*\mathbf{R})\gamma = \mathbf{R}^*\mathbf{r}_0, \quad \text{where} \quad \mathbf{R} \equiv [\mathbf{r}_1, \ldots, \mathbf{r}_\ell],$$

leading to a minimal residual (which can be viewed as the residual after ℓ steps of GMRES with initial residual $\mathbf{r}_0$). As for BiCGstab(ℓ), a suitable combination with the Galerkin residual (that is, the residual after ℓ steps of the full orthogonalization method) may lead to more stability in the IDR step (see [8]).

Assembling matrices such as $\mathbf{U}_0$, $\mathbf{U}_1 = \mathbf{A}\mathbf{U}_0$, …, $\mathbf{U}_j = \mathbf{A}^j\mathbf{U}$ into one tall matrix $\mathbf{U} = [\mathbf{U}_0; \ldots; \mathbf{U}_j]$ and vectors such as $\mathbf{r}_0$, $\mathbf{r}_1 = \mathbf{A}\mathbf{r}_0$, …, $\mathbf{r}_{j-1} = \mathbf{A}^{j-1}\mathbf{r}_0$ into one tall vector $\mathbf{r} = [\mathbf{r}_0; \mathbf{r}_1; \ldots; \mathbf{r}_{j-1}]$ allows the compact representation of vector updates in

Select an initial guess $\mathbf{x}$ and an $n \times s$ matrix $\widetilde{\mathbf{R}}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \star \mathbf{x}$, $\mathbf{r} = [\mathbf{r}_0]$
% *Generate an initial* $\mathbf{U} = [\mathbf{U}_0; \mathbf{U}_1] = [\mathbf{U}_0; \mathbf{A}\mathbf{U}_0]$
for $q = 1, \ldots, s$
  if $q = 1$, $\mathbf{v}_0 = \mathbf{r}_0$, else, $\mathbf{v}_0 = \mathbf{v}_1$
  $\mathbf{v} = [\mathbf{v}_0; \mathbf{A} \star \mathbf{v}_0]$
  $\mu = (\mathbf{U}_{0\,(:,1:q-1)})^* \mathbf{v}_0$, $\mathbf{v} = \mathbf{v} - \mathbf{U}_{(:,1:q-1)}\mu$, $\mathbf{v} = \mathbf{v}/\|\mathbf{v}_0\|$
  $\mathbf{U}_{(:,q)} = \mathbf{v}$
end for
while $\|\mathbf{r}_0\| > tol$
  for $j = 1, \ldots, \ell$
    % *An IDR step*
    $\sigma = \widetilde{\mathbf{R}}_0^* \mathbf{U}_j$, $\alpha = \sigma^{-1}(\widetilde{\mathbf{R}}_0^* \mathbf{r}_{j-1})$
    $\mathbf{x} = \mathbf{x} + \mathbf{U}_0 \alpha$, $\mathbf{r} = \mathbf{r} - [\mathbf{U}_1; \ldots; \mathbf{U}_j]\alpha$, $\mathbf{r} = [\mathbf{r}; \mathbf{A} \star \mathbf{r}_{j-1}]$
    for $q = 1, \ldots, s$
      if $q = 1$, $\mathbf{v} = \mathbf{r}$, else, $\mathbf{v} = [\mathbf{v}_1; \ldots; \mathbf{v}_{j+1}]$
      $\beta = \sigma^{-1}(\widetilde{\mathbf{R}}_0^* \mathbf{v}_j)$, $\mathbf{v} = \mathbf{v} - \mathbf{U}\beta$, $\mathbf{v} = [\mathbf{v}; \mathbf{A} \star \mathbf{v}_j]$
      $\mu = (\mathbf{V}_{j\,(:,1:q-1)})^* \mathbf{v}_j$, $\mathbf{v} = \mathbf{v} - \mathbf{V}_{(:,1:q-1)}\mu$, $\mathbf{v} = \mathbf{v}/\|\mathbf{v}_j\|$
      $\mathbf{V}_{(:,q)} = \mathbf{v}$
    end for
    if $j < \ell$, $\mathbf{U} = \mathbf{V}$
  end for
  % *The polynomial step*
  $\gamma = [\gamma_1; \ldots; \gamma_\ell] = \mathtt{argmin}_\gamma \|\mathbf{r}_0 - [\mathbf{r}_1, \ldots, \mathbf{r}_\ell]\gamma\|$
  $\mathbf{x} = \mathbf{x} + [\mathbf{r}_0, \ldots, \mathbf{r}_{\ell-1}]\gamma$, $\mathbf{r} = \mathbf{r}_0 - [\mathbf{r}_1, \ldots, \mathbf{r}_\ell]\gamma$
  $\mathbf{U} = [\mathbf{V}_0 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_j; \mathbf{V}_1 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_{j+1}]$
end while

ALGORITHM 5.3. IDRstab. *The* $\mathbf{U}_j$, $\mathbf{r}_{j-1}$, $\mathbf{v}_j$, *and* $\mathbf{V}_j$ *in the computation of* σ, α, β, *and* μ, *respectively, are related to* $\mathbf{U}$, $\mathbf{r}$, $\mathbf{v}$, *and* $\mathbf{V}$ *according to* $\mathbf{U} = [\mathbf{U}_0; \ldots; \mathbf{U}_j]$, $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_{j-1}]$, $\mathbf{v} = [\mathbf{v}_0; \ldots; \mathbf{v}_j]$, *and* $\mathbf{V} = [\mathbf{V}_0; \ldots; \mathbf{V}_{j+1}]$, *respectively. Note that the sizes of* $\mathbf{r}$, $\mathbf{U}$, *and* $\mathbf{v}$ *change during the loop: at the beginning of the "*`for` $j = \ldots$*" loop,* $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_{j-1}]$ *and* $\mathbf{U} = [\mathbf{U}_0; \ldots; \mathbf{U}_j]$. *The assignment* $\mathbf{v} = [\mathbf{v}_1; \ldots; \mathbf{v}_{j+1}]$ *shifts the coordinates of* $\mathbf{v}$ n *positions (dropping* $\mathbf{v}_0$*). As before,* $\mathbf{A}$ *times a vector* $\mathbf{v}$ *is denoted by* $\mathbf{A} \star \mathbf{v}$.

the IDR step that is used in Algorithm 5.3. Implementation of this may also lead to more efficiency on certain types of computers.

Table 5.1 summarizes the computational costs. The number of DOTs depends mildly on ℓ: the computation of the coefficients of the minimizing polynomial requires

TABLE 5.1

The table gives the costs of the algorithm averaged per MV: one sweep of IDRstab involves $(s+1)\ell$ *MVs. A DOT is an inner product between two* n*-vectors; an AXPY is a vector update of the form* $\mathbf{x} + \alpha\mathbf{y}$ *with* $\mathbf{x}$ *and* $\mathbf{y}$ n*-vectors. The third column represents the average costs for the orthogonalization using classical Gram–Schmidt.*

DOTs	$2s - 1 + \frac{1}{2}\frac{\ell+5}{s+1}$	$\frac{1}{2}s$
AXPYs	$\frac{1}{2}s(\ell+3) + 2$	$\frac{1}{4}(s-1)(\ell+3) + \frac{1}{4}\frac{\ell+3}{s+1}$

$\approx \frac{1}{2}\ell^2$ inner products per $\ell(s+1)$ MVs. The extra costs for increasing ℓ is mostly in the extra vector updates.

6. Numerical experiments. This section presents numerical experiments on test problems that, for different reasons, are difficult to solve by iterative methods.

As an implementation independent measure for the amount of work to solve the systems, we will give the number of MVs. This measure is realistic for applications where the (preconditioned) MV is far more expensive than a vector operation, which is typically the case, and as long as the number of vector operations per MV is modest. We will also report the actual solution times (CPU times). However, we stress that the actual solution times are implementation dependent and that we have not optimized our code. Further optimizations are possible to limit storage and vector operations, as we have indicated in sections 4.4.4–4.4.5.

We report on results for Bi-CGSTAB, IDR(s), and BiCGstab(ℓ). These results are actually obtained with our IDRstab code, with $s = \ell = 1$ for Bi-CGSTAB, $\ell = 1$ for IDR(s), and $s = 1$ for BiCGstab(ℓ).

For the columns of $\widetilde{\mathbf{R}}_0$ we take the orthogonalization of s real random vectors. No preconditioner is applied. The iterative processes are terminated once the residual norm, divided by the norm of the right-hand-side vector, drops below 10^{-9}.

The experiments are performed with MATLAB 7.5 on a standard PC with an Intel Core 2 duo processor and 4 Gb of RAM.

6.1. A test problem from the MATRIX-MARKET collection. For our first test problem, we consider the SHERMAN5 matrix from the MATRIX-MARKET collection. The size of this matrix is 3312×3312. The right-hand-side vector is chosen such that the solution is the vector consisting of ones.

The SHERMAN5 matrix is, in contrast to the well-known SHERMAN4 matrix from the same collection, notoriously difficult to solve by iterative methods. This is due to the fact that the SHERMAN5 matrix is highly indefinite, as is illustrated by the plot of the spectrum given in Figure 6.1.

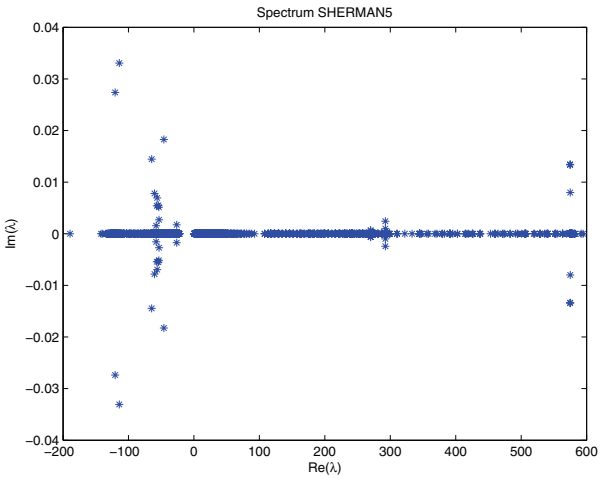


FIG. 6.1. *Spectrum of SHERMAN5.*

The system is solved with IDRstab, with different combinations of the parameters s and ℓ. Figure 6.2 shows the convergence behavior for the following combinations of

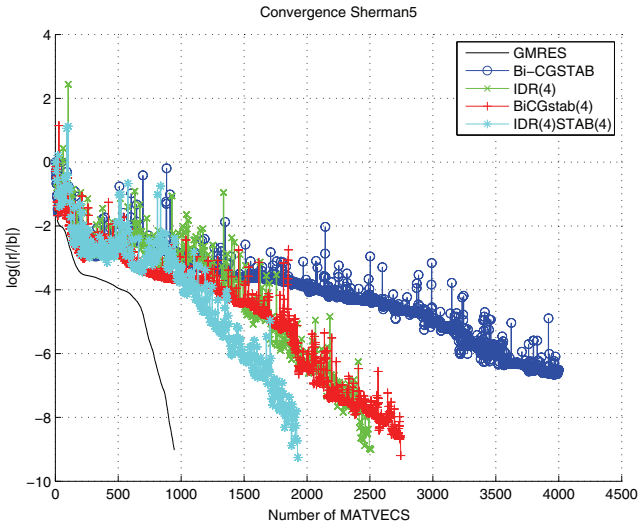


FIG. 6.2. *SHERMAN5: convergence for IDRstab and for GMRES.*

s and ℓ: $s = 1, \ell = 1$ (equivalent with Bi-CGSTAB), $s = 4, \ell = 1$ (equivalent with IDR(4)), $s = 1, \ell = 4$ (equivalent with BiCGstab(4)), and $s = \ell = 4$.

For comparison, we have also included full GMRES in our experimental evaluation. The comparison with GMRES is interesting for theoretical reasons only: it shows how close the required number of IDRstab MVs is to the optimal number of GMRES MVs. For practical applications, where the typical problem size is orders of magnitude larger than for our test problem (millions of unknowns are quite common), it is, of course, impossible to perform hundreds of GMRES iterations due to the excessive memory use and the large overhead of the vector operations. Table 6.1 gives, for different combinations of s and ℓ, the required number of MVs and, within parentheses, the CPU time to achieve the desired accuracy.

TABLE 6.1
SHERMAN5: matrix-vector products and CPU times for IDRstab. Optimal values are in bold.

$\ell \backslash s$	1	2	4	8
1	n.c.	3121 (3.4s)	2508 (2.7s)	2401 (3.1s)
2	3570 (4.2s)	2401 (2.7s)	2198 (2.5s)	1897 (2.7s)
4	2744 (3.3s)	2125 (2.5s)	1928 (**2.4s**)	**1762** (2.9s)
8	2598 (3.5s)	2020 (2.8s)	1848 (2.9s)	1798 (3.9s)

For $s = \ell = 1$ (i.e., Bi-CGSTAB), no convergence (n.c.) occurs within 4000 MVs. For higher values of s or ℓ, IDRstab always converges. Moreover, Table 6.1 shows that the number of MVs is reduced by increasing either s or ℓ, and more importantly, that by increasing *both* s and ℓ the number of MVs is reduced to a level lower than is achieved by IDR(s) and BiCGstab(ℓ). This is also reflected in the CPU times. Despite the fact that our code can be further optimized with respect to vector operations, a significant reduction in CPU time is achieved by choosing both s and ℓ (moderately) larger than 1. We observed that the norm of the true residual for $\ell = 4$ or 8 sometimes stagnated above the norm of the recursively computed residual. This so-called residual

gap can be remedied by the techniques described in [9]. We have not applied these here.

6.2. A two-dimensional (2D) convection-diffusion problem with a positive shift. The second test problem we consider is the finite volume discretization of the following partial differential equations on the unit square $[0\,,\,1] \times [0\,,\,1]$:

$$-u_{xx} - u_{yy} + 1000(xu_x + yu_y) + 10u = f\ .$$

Dirichlet conditions are imposed on the boundaries. This problem is discretized with the finite volume method on a 65×65 grid, which yields a matrix of size 4096×4096. The right-hand-side vector $\mathbf{f}$ of the discrete system is chosen such that the solution is the vector consisting of ones. This problem is taken from [8].

Figure 6.3 shows the convergence of four IDRstab variants and, for comparison, also of full GMRES. This figure shows that also for this problem a significant gain can be achieved by choosing both s and ℓ higher. The convergence curve for IDRstab with $s = 4$ and $\ell = 4$ follows the optimal convergence curve of GMRES closely. Table 6.2 gives, for different combinations of s and ℓ, the required number of MVs to achieve the desired accuracy.

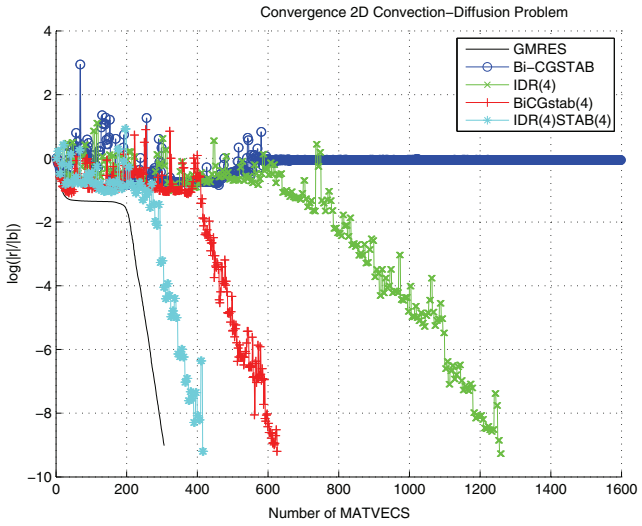


FIG. 6.3. *2D convection-diffusion problem: convergence for IDRstab and for GMRES.*

TABLE 6.2

2D convection-diffusion problem: matrix-vector products and CPU times for IDRstab. Optimal values are in bold.

$\ell \backslash s$	1	2	4	8
1	n.c.	1729 (2.4s)	1258 (1.7s)	898 (1.5s)
2	626 (0.9s)	457 (0.6s)	403 (**0.5s**)	**376** (0.6s)
4	626 (0.9s)	490 (0.7s)	418 (0.6s)	**376** (0.7s)
8	602 (1.0s)	457 (0.8s)	403 (0.8s)	493 (0.8s)

Also for this example, Bi-CGSTAB does not converge within 4000 MVs. IDR(s) (i.e., IDRstab with $\ell = 1$) converges badly for all values of s, which could be expected

since the system matrix is almost skew-symmetric. However, we do remark that the convergence improves significantly by choosing s higher. BiCGstab(ℓ) convergences after around 600 MVs for all values of ℓ larger than 1. The most important conclusion is that by combining IDR(s) and BiCGstab(ℓ), i.e., choosing both s and ℓ larger than 1, it is possible to almost close the gap with GMRES and almost get optimal convergence with respect to MVs. Moreover, the lowest CPU time is achieved by choosing both $s = 4$ and $\ell = 2$, again, showing the potential of the combination of IDR(s) and BiCGstab(ℓ).

6.3. A three-dimensional (3D) convection-dominated problem. The third test problem is taken from [6] and is also included in [12]. This problem was used to illustrate that Bi-CGSTAB does not work well due to the strong nonsymmetry of the system matrix. As was shown in [12], IDR(s) also performs poorly for this problem (if the shadow space $\widetilde{\mathbf{R}}_0$ is chosen real, the bad convergence can be overcome by choosing $\widetilde{\mathbf{R}}_0$ complex).

The test problem is the finite difference discretization of the following partial differential equations on the unit cube $[0\,,\,1] \times [0\,,\,1] \times [0\,,\,1]$:

$$u_{xx} + u_{yy} + u_{zz} + 1000u_x = F.$$

Homogeneous Dirichlet conditions are imposed on the boundaries. The vector F is defined by the solution $u(x, y, z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$. The partial differential equation is discretized using central differences for both the convection and diffusion terms. We take 52 grid points in each direction (including boundary points) which yields a system of 125,000 equations.

Figure 6.4 shows the convergence behavior of four IDRstab variants and of GMRES.

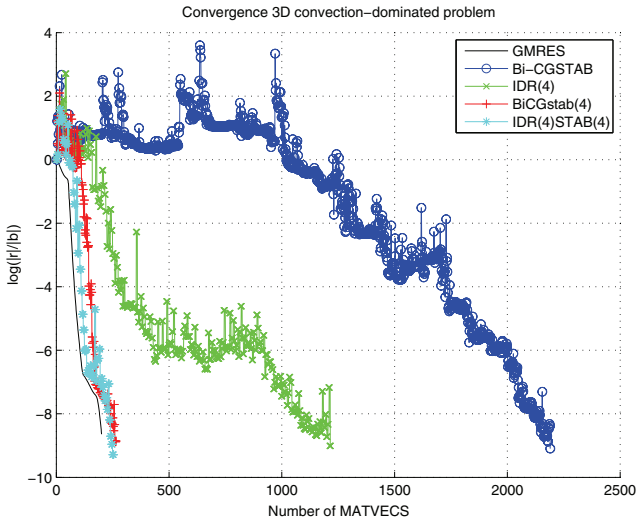


FIG. 6.4. *3D convection-diffusion problem: convergence of IDRstab and of GMRES.*

Figure 6.4 shows that the convergence curve of the IDRstab with $\ell = 4, s = 1$ (i.e., BiCGstab(4)) is close to the optimal convergence curve of full GMRES. As a result, the reduction in MVs that is obtained by increasing both ℓ and s is modest. This

is also apparent from the results tabulated in Table 6.3. The BiCGstab(ℓ) variants are close to optimal; therefore, only a modest improvement can be achieved by also choosing $s > 1$. Still, we remark that the number of MVs for $\ell = 8, s = 8$ is smaller than for the four BiCGstab(ℓ) variants that we investigated.

TABLE 6.3
3D convection-diffusion problem: matrix-vector products and CPU times for IDRstab. Optimal values are in bold.

$\ell \backslash s$	1	2	4	8
1	2190 (136.6s)	2089 (131.7s)	1218 (84.8s)	655 (60.9s)
2	248 (**15.3s**)	265 (16.5s)	253 (18.1s)	250 (24.3s)
4	264 (18.1s)	265 (19.0s)	253 (21.6s)	241 (28.5s)
8	320 (25.6s)	259 (23.1s)	248 (28.0s)	**232** (37.0s)

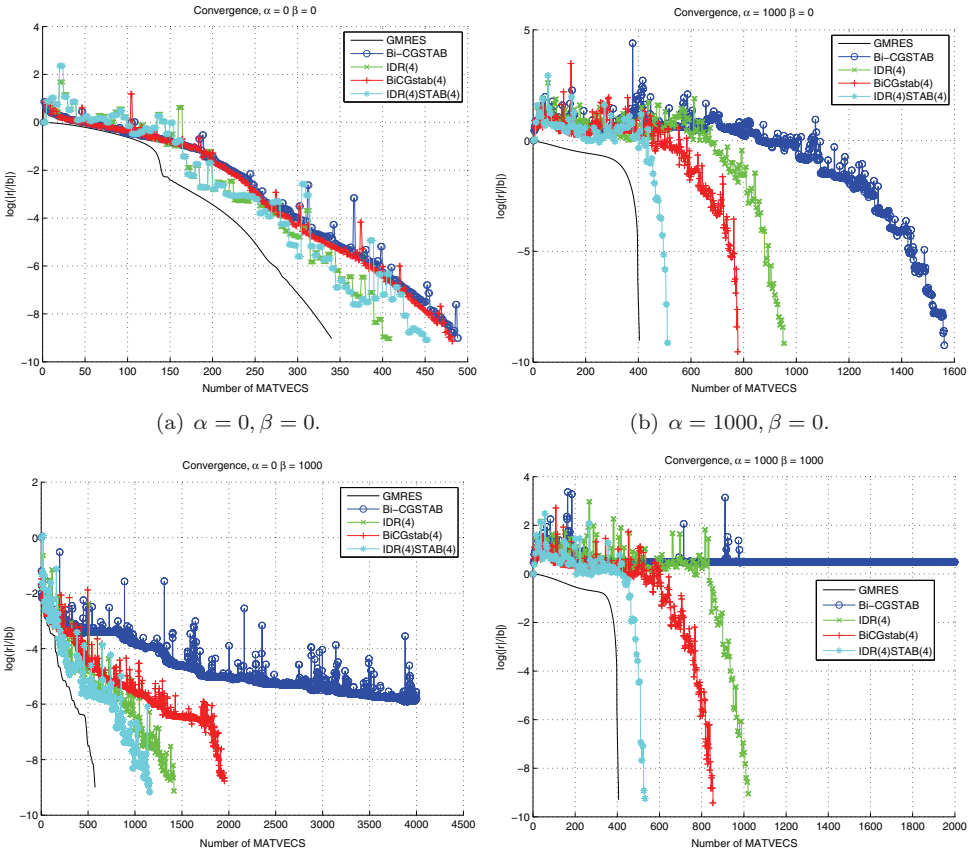
6.4. A parameterized 2D convection-diffusion-reaction equation. The last, parameterized, test problem is the finite difference discretization of the following partial differential equations on the unit square $[0\,,\,1] \times [0\,,\,1]$:

$$-u_{xx} - u_{yy} + \frac{\alpha}{\sqrt{2}}(u_x + u_y) - \beta u = F.$$

Homogeneous Dirichlet conditions are imposed on the boundaries. The vector F is defined by the solution $u(x, y, z) = xy(1 - x)(1 - y)$. The partial differential equation is discretized using central differences. We take 201 grid points in each direction (including boundary points) which yields a system of 39,601 equations.

We consider four different choices for the parameters α and β, which results in test problems with quite different characteristics. Figure 6.5 shows the convergence behavior of four IDRstab variants and of GMRES. We make the following observations:

- $\alpha = \beta = 0$: the matrix is symmetric positive definite for this choice. Figure 6.5(a) shows that the convergence for Bi-CGSTAB is close to the optimal GMRES convergence. As a result, the gain of choosing s and ℓ greater than 1 is limited.
- $\alpha = 1000, \beta = 0$: the system matrix is highly nonsymmetric. One expects BiCGstab(ℓ) to perform better than IDR(s) for such a problem. This is confirmed by the convergence plot given in Figure 6.5(b): IDR(4) performs significantly better than Bi-CGSTAB, and BiCGstab(4), in turn, performs better than IDR(4). But also this example shows that choosing both s and ℓ larger than 1 improves the convergence. The convergence of IDR(4)stab(4) follows the optimal GMRES convergence closely.
- $\alpha = 0, \beta = 1000$: the system matrix is symmetric but indefinite. The convergence for this case is shown in Figure 6.5(c). Here, IDR(4) performs better than BiCGstab(4), and again, the combination IDRstab(4,4) is fastest. For this problem, Bi-CGSTAB does not converge within 4000 MVs.
- $\alpha = 1000, \beta = 1000$: the system matrix is highly nonsymmetric and has eigenvalues in both the right and the left half-plane. The convergence of the IDRstab variants shown in Figure 6.5(d) is similar (apart from the fact that here Bi-CGSTAB stagnates completely) to that of the highly nonsymmetric but definite case that is shown in Figure 6.5(b). IDR(4)stab(4) again closely follows the GMRES convergence.



(a) $\alpha = 0, \beta = 0$. (b) $\alpha = 1000, \beta = 0$.

(c) $\alpha = 0, \beta = 1000$. (d) $\alpha = 1000, \beta = 1000$.

FIG. 6.5. *2D parameterized convection-diffusion-reaction problem: convergence of IDRstab and of GMRES.*

As for the previous examples, we have tested all the combinations $s = 1, 2, 4, 8$ and $\ell = 1, 2, 4, 8$. Table 6.4 shows the optimal combinations of s and ℓ with respect to number of MVs. Table 6.5 gives the combinations that give the lowest CPU time. Also, these results show that choosing both s and ℓ larger than 1 may result in (significantly) less MVs and a lower CPU time than can be obtained with IDR(s) or BiCGstab(ℓ). For example, for the most difficult test case $\alpha = 1000$, $\beta = 1000$, the number of MVs is reduced from 810 for BiCGstab(8) to 523 for IDR(4)stab(2). The CPU time is reduced from 13s for BiCGstab(2) to 9.5s for IDR(4)stab(2).

TABLE 6.4
Optimal combinations of s and ℓ with respect to the number of MVs.

Problem	s	ℓ	MVs	CPU time [s]
$\alpha = 0, \beta = 0$	4	2	403	7.3
$\alpha = 1000, \beta = 0$	8	2	466	12.0
$\alpha = 0, \beta = 1000$	8	1	970	22.0
$\alpha = 1000, \beta = 1000$	4	2	523	9.5

TABLE 6.5
Optimal combinations of s and ℓ with respect to CPU time.

Problem	s	ℓ	MVs	CPU time [s]
$\alpha = 0, \beta = 0$	4	1	408	7.1
$\alpha = 1000, \beta = 0$	4	2	518	9.5
$\alpha = 0, \beta = 1000$	2	4	1132	20.1
$\alpha = 1000, \beta = 1000$	4	2	523	9.5

7. Conclusions. The goal of this research was to unify two of the most powerful short-recurrence Krylov methods for nonsymmetric systems into one method that inherits the strong points of both. To this end we have derived the method IDRstab. We have illustrated the potential of this method with numerical experiments that show that IDRstab for certain problems outperforms both IDR(s) and BiCGstab(ℓ).

The quest for IDRstab has provided us with considerable new insight. In [12] it was postulated that a new IDR theorem was needed to make an IDR method that uses higher order stabilization polynomials. In this paper we have shown that this is not the case.

In the method that we have presented in this paper, many specific choices were made. The resulting algorithm is, for modest values of s and ℓ (e.g., $s = 4, \ell = 2$), in our experience efficient and numerically stable. Nevertheless, further improvements can still be made, and we have indicated some of them.

After submission of our manuscript, we received a new report [13] by Tanio and Sugihara describing a related method.

Acknowledgments. We thank Peter Sonneveld for introducing us to IDR and sharing his deep insight with us. We also thank the referees for their comments.

REFERENCES

[1] D. R. FOKKEMA, G. L. G. SLEIJPEN, AND H. A. VAN DER VORST, *Generalized conjugate gradient squared*, J. Comput. Appl. Math., 71 (1996), pp. 125–146.
[2] M. H. GUTKNECHT, *Variants of BICGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.
[3] M. H. GUTKNECHT, *IDR explained*, Electron. Trans. Numer. Anal., 36 (2010), pp. 126–128.
[4] M. H. GUTKNECHT AND K. J. RESSEL, *Look-ahead procedures for Lanczos-type product methods based on three-term Lanczos recurrences*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1051–1078.
[5] K. J. RESSEL AND M. H. GUTKNECHT, *QMR smoothing for Lanczos-type product methods based on three-term recurrences*, SIAM J. Sci. Comput., 19 (1998), pp. 55–73.
[6] G. L. G. SLEIJPEN AND D. R. FOKKEMA, *BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32.
[7] G. L. G. SLEIJPEN, P. SONNEVELD, AND M. B. VAN GIJZEN, *Bi-CGSTAB as an induced dimension reduction method*, Appl. Numer. Math., (2009), DOI: 10.1016/j.apnum.2009.07.001.
[8] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numer. Algorithms, 10 (1995), pp. 203–223.
[9] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *Reliable updated residuals in hybrid Bi-CG methods*, Comput., 56 (1996), pp. 141–163.
[10] G. L. G. SLEIJPEN AND M. B. VAN GIJZEN, *BiCGstab(ℓ) Strategies to Induce Dimension Reduction*, Report ISSN 1389-6520, Department of Applied Mathematical Analysis, Delft University of Technology, The Netherlands, 2009.
[11] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[12] P. SONNEVELD AND M. 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), pp. 1035–1062.

[13] M. Tanio and M. Sugihara, *GBi-CGSTAB(s,L): IDR(s) with Higher-Order Stabilization Polynomials*, Report METR 2009-16, Department of Mathematical Informatics, University of Tokyo, Japan, 2009.

[14] 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.

[15] M. van Gijzen and P. Sonneveld, *An Elegant IDR(s) Variant That Efficiently Exploits Bi-orthogonality Properties*, Preprint 08-21, Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands, 2008.

[16] P. Wesseling and P. Sonneveld, *Numerical experiments with a multiple grid and a preconditioned Lanczos type method*, in Approximation Methods for Navier-Stokes Problems, Lecture Notes in Math. 771, Springer Verlag, Heidelberg, 1980, pp. 543–562.

[17] M.-C. Yeung and T. F. Chan, *ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple Lanczos starting vectors*, SIAM J. Sci. Comput., 21 (1999), pp. 1263–1290.

[18] S.-L. Zhang, *GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 18 (1997), pp. 537–551.