DELFT UNIVERSITY OF TECHNOLOGY

REPORT 09-02

Exploiting BiCGstab(ℓ) strategies to induce dimension reduction

Gerard L.G. Sleijpen and Martin B. van Gijzen

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2009

Copyright © 2009 by Department of Applied Mathematical Analysis, Delft, The Netherlands.

No part of the Journal may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission from Department of Applied Mathematical Analysis, Delft University of Technology, The Netherlands.

EXPLOITING BICGSTAB(ℓ) STRATEGIES TO INDUCE DIMENSION REDUCTION

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

Abstract. IDR(s) [9] and BiCGstab(ℓ) [5] 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 derive a new method, that we call IDRstab, that that combines the strengths of IDR(s) and BiCGstab(ℓ). To derive IDRstab we extend the results that we reported on in [7] where we considered Bi-CGSTAB as an IDR method. We will analyse the structure of IDR in detail and introduce the new concept of the Sonneveld space. Through numerical experiments we will show that IDRstab can outperform both IDR(s) and BiCGstab(ℓ).

Keywords: Bi-CGSTAB, Bi-CG, iterative linear solvers, Krylov subspace methods, IDR.
AMS(MOS) subject classification: 65F15, 65N25.

1. Introduction. Bi-CGSTAB [10] is the most popular short-recurrence method for solving large nonsymmetric systems of equations

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

of dimension $N \times N$. 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 (or 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 super-linear convergence during the iterative process. Bi-CG, on the other hand, does not poses 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 large extend the success of Bi-CGSTAB.

IDR(s) [9] is a new short-recurrence Krylov subspace method for solving large nonsymmetric linear systems. The IDR method generates residuals that are forced to be in subspaces $\mathcal{G}_j$ of decreasing dimension. These nested subspaces are related by $\mathcal{G}_j = (\mathbf{I} - \omega_j \mathbf{A})(\widetilde{\mathbf{R}}_0^{\perp} \cap \mathcal{G}_{j-1})$, 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_j's are non-zero scalars. This working principle is different from the more conventional Krylov subspace methods like Bi-CGSTAB that construct residuals in the *growing* Krylov subspace

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = \text{Span}\left(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \cdots, \mathbf{A}^{k-1}\mathbf{r}_0\right)$$

The numerical examples that are described in [9] show that IDR(s), with $s > 1$ and not too big (as $s = 2$, 4, or 6) is quite competitive, and outperforms Bi-CGSTAB for important classes of problems.

Although the idea behind IDR(s) is quite different from that of Bi-CGSTAB, the two are mathematically closely related, specifically, IDR(1) is mathematically equivalent with Bi-CGSTAB, and IDR(s) with $s > 1$ is related (but not mathematically

*Mathematical Institute, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, the Netherlands, e-mail: sleijpen@math.uu.nl

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

equivalent) to the Bi-CGSTAB generalisation ML(k)BiCGSTAB [13] 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 shadow residuals. We refer to [9] for the details.

Like Bi-CGSTAB, IDR(s) lacks a global error minimization property. In order to smoothen the convergence, the ω_j's are chosen such that the next residual is minimized in norm. As is explained in [9], a new ω can be chosen every $s+1$-st step (matrix-vector multiplication). Note that for $s = 1$ this is exactly the same as in Bi-CGSTAB, where a new ω is selected after every second matrix-vector multiplication 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 does not work well 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), leading not only to a more effective residual minimization (order ℓ minimization), but also to more stability in the computation of the Bi-CG coefficients of the process [6].

The linear minimal residual step in IDR(s) also makes this method less suited for problems with a strongly non-symmetric matrix. Example 3 in [9] presents such a problem. The 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 $\ell > 1$, shows a vastly superior performance, and outperforms IDR(s) irrespective of the choice of s.

It is therefore a natural idea to combine IDR(s) and BiCGstab(ℓ) to a method that combines the strong points of both. In order to derive such a method, we will extend the results that we reported on in [7]. In this paper, we consider the mathematical relation between Bi-CGSTAB and IDR and propose a block version of Bi-CGSTAB that is mathematically an IDR-method. As we will show, this block Bi-CGSTAB variant can be generalised in a straightforward way to include higher order stabilization polynomials. We will call the resulting method IDRstab. The method unifies IDR(s) and BiCGstab(ℓ): by choosing $\ell = 1$ it is mathematically equivalent to IDR(s), and by choosing $s = 1$ to BiCGstab(ℓ).

To answer the question whether the combination of IDR(s) and BiCGstab(ℓ) yields a method that is more efficient than either IDR(s) or BiCGstab(ℓ) we will present numerical experiments on test problems with different characteristics. We will show that IDRstab with s and ℓ larger than 1 can be significantly faster than both IDR(s) and BiCGstab(ℓ).

This paper is organised as follows: The next section reviews the IDR principle, and also provides new insights. In particular it introduces the concept of the Sonneveld subspace. Section 3 explains how BiCGstab(ℓ) is derived from BiCG. Section 4 presents an IDR variant that allows for easy incorporation of ℓ-th degree stabilization polynomials. In order to derive this variant we give insightful schemes on how the different vector quantities in IDR(s) are related. Section 5 brings together the results of the previous sections to derive the new iterative solver IDRstab. The excellent performance of the 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 a 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 $\widetilde{\mathbf{R}}^{\perp}$.

NOTATION 1.2. When we mention the number of iterations, or the number of steps, we always refer to the number of matrix-vector multiplications (MVs), where each step or iteration always corresponds to one (one-dimensional) MV. A multiplication $\mathbf{AV}$, with $\mathbf{V}$ an $n \times s$ matrix is counted as s MVs.

NOTATION 1.3. Updates of the form $\mathbf{v} - \mathbf{C}\vec{\beta}$ 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 $\vec{\beta}$ is determined by an orthogonality requirement $\mathbf{v} - \mathbf{C}\vec{\beta} \perp \widetilde{\mathbf{R}}$ where $\widetilde{\mathbf{R}}$ is some given $n \times s$ matrix. For ease of notation, we will simply put

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

Note that, with $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{C}$, $\vec{\beta}$ can be computed as $\vec{\beta} = \sigma^{-1}(\widetilde{\mathbf{R}}^{*}\mathbf{v})$ (or with a more stable variant as repeated or modified). The operator $\mathbf{I} - \mathbf{C}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$ is a skew projection onto the orthogonal complement of $\widetilde{\mathbf{R}}$.

NOTATION 1.4. We wil 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 $\mathbf{W}_{(:,q)} \equiv \mathbf{w}_q$ $(q \leq s)$;
— $[\mathbf{U}_0; \ldots; \mathbf{U}_j] \equiv [\mathbf{U}_0^{\mathrm{T}}, \ldots, \mathbf{U}_j^{\mathrm{T}}]^{\mathrm{T}}$.

2. The IDR principle. The IDR method exploits the following result from which we learn how to construct an strictly increasing sequence of nested linear subspaces. For a proof we refer to [9, 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*

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

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. From the statement that the residual is contained in $\mathcal{G}_k$, we can not conclude anything about the size of the norm unless $\mathcal{G}_k = \{\mathbf{0}\}$. In practice, however, the residual norms tend to converge fast (towards zero). The convergence is (often much) faster for larger s.

To update the residual in $\mathcal{G}_k$ to a residual in $\mathcal{G}_{k+1}$ requires $s + 1$ MVs. The following result (Th. 2.3) relates the 'IDR' spaces $\mathcal{G}_k$ to Krylov subspaces and it also explains why $s + 1$ MVs are required. For a proof, we refer to [7]. We first introduce some notation and terminology.

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

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

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

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

If there is no ambiguity, we will write $\mathcal{S}(P)$ or $\mathcal{S}_P$ instead of $\mathcal{S}_P(\mathbf{A}, \widetilde{\mathbf{R}}_0)$. For reasons explained in Note 2.4 we cal $\mathcal{S}_P$ the (Pth) *Sonneveld subspace* (generated by $\mathbf{A}$ and $\widetilde{\mathbf{R}}_0$), k is the *order* of the Sonneveld subspace.

THEOREM 2.3. *With $\widetilde{\mathbf{R}}_0$, (μ_j), and $\mathcal{G}_k$ be as in Theorem* 2.1*, and P_k defined by $P_k(\lambda) \equiv \prod_{j=1}^{k}(\mu_j - \lambda)$ $(\lambda \in \mathbb{C})$, we have the*

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

To increase the order of the Sonneveld space, 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 theorem also shows that, any hybrid Bi-CG method can be viewed as an IDR method: hybrid Bi-CG methods produce residuals that can be expressed as $P_k(\mathbf{A})\mathbf{r}_k^{\mathrm{BiCG}}$, where $\mathbf{r}_k^{\mathrm{BiCG}} \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$ is the kth Bi-CG residual (apply the theorem with $s = 1$ and $\widetilde{\mathbf{R}}_0 = [\widetilde{\mathbf{r}}_0]$), and some polynomial P_k determined by the specific hybrid variant (for more details and references, see §3). In some sense, the polynomials P_k define the hybrid Bi-CG method. The selection $P_k(\lambda) = \prod_{j\leq k}(1 - \omega_j\lambda)$ of Bi-CGSTAB and of BiCGstab(ℓ) will be of special interest to us in this paper.

Alg. 2.1 displays a simple, but elegant, implementation of the IDR method. In practice, additional stability steps may be required, for instance, to cope with nearly singular σ (cf. [9, 7]). Note that this implementation produces the 'primary' residuals mentioned above (the ones that arrive first in the $\mathcal{S}(P_k)$), but also $2s+1$ 'secondary' residuals (denoted by $\mathbf{v}$ and $\mathbf{r}$ in Alg. 2.1): two residuals are computed at every MV. In this algorithm, $\omega_k = 1/\mu_k$ has been selected to minimize the norm of one residual update every $s+1$ MV: to be more precise, the primary residual $\mathbf{r}$ is computed as $\mathbf{r} = \mathbf{v} - \omega\mathbf{A}\mathbf{v}$ with $\omega = \mathrm{minarg}_\omega\|\mathbf{v} - \omega\mathbf{A}\mathbf{v}\|$. This approach of selecting ω_k follows the Bi-CGSTAB strategy: in fact, if $s = 1$, the primary residuals coincide with Bi-CGSTAB residuals (that is, the residuals at the the end of a Bi-CGSTAB cycle). Other selections for ω_k (and μ_k) are possible and can be more effective, as we will learn in this paper.

NOTE 2.4. Given the fact that (sub)spaces are named after mathematicians, we feel that the name 'Sonneveld subspace' is appropriate. Peter Sonneveld introduced the IDR concept of Theorem 2.1 in [12]. In this paper from 1980, hij also gave an algorithm that is equivalent to Bi-CGSTAB. The relation to hybrid Bi-CG expressed in Theorem 2.3, and the other principle in hybrid Bi-CG (see the summary in §3 of hybrid Bi-CG methods), that any polynomial of appropriate degree can be used to avoid multiplication by $\mathbf{A}^*$ was introduced in the CGS paper [8] of 1989. Later more stable and effective variants of Bi-CGSTAB ([10, 2, 5, ...] and CGS [1, 14, ...] were published and made hybrid Bi-CG methods popular solvers for non-symmetric systems.

3. From Bi-CG to BiCGstab(ℓ). In this section, we recall the derivation of BiCGstab(ℓ) in order to clarify the derivation of our IDR variants in §§4–5. It also

Select an $\mathbf{x}_0$.
Select $n \times s$ matrices $\widetilde{\mathbf{R}}_0$ and $\mathbf{U}$
Compute $\mathbf{S} = \mathbf{A}\mathbf{U}$
$\mathbf{x} = \mathbf{x}_0$, $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$i = 1$, $j = 0$
while $\|\mathbf{r}\| > tol$
  $\sigma = \widetilde{\mathbf{R}}_0^*\mathbf{S}$, $\vec{\rho} = \widetilde{\mathbf{R}}_0^*\mathbf{r}$, $\vec{\alpha} = \sigma^{-1}\vec{\rho}$
  $\mathbf{v} = \mathbf{r} - \mathbf{S}\vec{\alpha}$, $\mathbf{c} = \mathbf{A}\mathbf{v}$
  If $j = 0$, $\omega \leftarrow \mathbf{c}^*\mathbf{v}/\mathbf{c}^*\mathbf{c}$,
  $\mathbf{U}_{(:,i)} \leftarrow \mathbf{U}\vec{\alpha} + \omega\mathbf{v}$, $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{U}_{(:,i)}$
  $\mathbf{r}_1 \leftarrow \mathbf{v} - \omega\mathbf{c}$, $\mathbf{S}_{(:,i)} = \mathbf{r} - \mathbf{r}_1$, $\mathbf{r} \leftarrow \mathbf{r}_1$
  $i \leftarrow i + 1$, if $i > s$, $i = 1$
  $j \leftarrow j + 1$, if $j > s$, $j = 0$
end while

Alg. 2.1: IDR(s). The ith column $\mathbf{S}_{(:,i)}$ and $\mathbf{U}_{(:,i)}$ of $\mathbf{S}$ and $\mathbf{U}$ are replaced in the loop by the newly computed vectors. The initial matrices $\mathbf{U}$ and $\widetilde{\mathbf{R}}_0$ have to be such that $\widetilde{\mathbf{R}}_0^*\mathbf{S}$ is non-singular.

explains how the IDR variants relate to BiCGstab (ℓ): our IDR variant can be viewed as BiCGstab(ℓ) with an s dimensional initial shadow residual $\widetilde{\mathbf{R}}_0$ instead of a one dimensional one $\widetilde{\mathbf{r}}_0$.

3.1. Bi-CG. Bi-CG uses coupled two term recurrences to produce a residual $\mathbf{r}_k$ at step k that is orthogonal to the kth 'shadow' Krylov subspace $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$:

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

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 $j \leq k$, 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 mehods), residuals are updated by vectors of the form $-\mathbf{A}\mathbf{u}$ with $\mathbf{u}$ always explicitly available.

The following formulation, equivalent to the one (3.1), will be of interest for later discussion.

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

Here, the vectors $\mathbf{A}\mathbf{u}_k$ are obtained by recursive update rather than by MV. Since $\mathbf{A}\mathbf{r}_k$ is needed the number of MVs is not affected: the coupled recurrences in (3.2) require only one vector update more per step (per MV) than the recurrences in (3.1). Both updates require orthogonality to the same vector $\widetilde{\mathbf{r}}_k$, which will turn out to be convenient. The third recurrence relation provides the update for the approximate solution $\mathbf{x}_{k+1}$. Note that the way of computing β_k is not affected: if $\vartheta_k \in \mathbb{C}$ is such that $\widetilde{\mathbf{r}}_{k+1} - \vartheta_k\mathbf{A}^*\widetilde{\mathbf{r}}_k \in \mathcal{K}_{k+1}(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$, then, with $\sigma_k \equiv \widetilde{\mathbf{r}}_k^*\mathbf{A}\mathbf{u}_k$ and $\rho_j \equiv \widetilde{\mathbf{r}}_j^*\mathbf{r}_j$,

$$\alpha_k = \frac{\rho_k}{\sigma_k} \quad \text{and} \quad \beta_k = \frac{\widetilde{\mathbf{r}}_k^*\mathbf{A}\mathbf{r}_{k+1}}{\widetilde{\mathbf{r}}_k^*\mathbf{A}\mathbf{u}_k} = \frac{\rho_{k+1}}{\vartheta_k\sigma_k}.$$

3.2. Hybrid Bi-CG. For each k, $\widetilde{\mathbf{r}}_k = P_k(\mathbf{A}^*)\widetilde{\mathbf{r}}_0$ for some polynomial P_k of degree k. With $\mathbf{P}_k \equiv P_k(\mathbf{A})$, (3.2) implies

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

There are a number of effective strategies for selecting the polynomials P_k, see. e.g., [8, 10, 2, 5, 1, 14]

3.3. Bi-CGSTAB. In Bi-CGSTAB, the polynomial P_{k+1} is of the form $P_{k+1}(\lambda) = (1 - \omega_{k+1}\lambda)\, P_k(\lambda)$, and a combination of (3.3) and

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

updates the residual $\mathbf{P}_k\mathbf{r}_k$ from $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{r}}_0)$ to the residual $\mathbf{P}_{k+1}\mathbf{r}_{k+1}$ in $\mathcal{S}(P_{k+1}, \mathbf{A}, \widetilde{\mathbf{r}}_0)$. The $\mathbf{P}_k\mathbf{r}_k$ are primary residuals, $\mathbf{P}_k\mathbf{r}_{k+1}$ are secondary ones. To complete the 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. Though a bit more expensive, we feel that this approach helps to clarify the derivation of our IDR variant in §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

$$\mathbf{P}_k\mathbf{r}_{k+\ell} - \gamma_1\mathbf{A}\mathbf{P}_k\mathbf{r}_{k+\ell} - \ldots - \gamma_\ell\mathbf{A}^\ell\mathbf{P}_k\mathbf{r}_{k+\ell} \tag{3.6}$$

is minimal and $\mathbf{P}_{k+\ell}\mathbf{r}_{k+\ell}$ equals this expression. Here, the γ_i are such that

$$1 - \gamma_1\lambda - \ldots - \gamma_\ell\lambda^\ell = (1 - \omega_{k+\ell}\lambda) \cdot \ldots \cdot (1 - \omega_{k+1}\lambda) \qquad (\lambda \in \mathbb{C}).$$

In actual computations, the γ_i are computed. Since they can only be computed at the end of the cycle of ℓ loops, the ω_j 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 [5] in the derivation of a BiCGstab(ℓ) algorithm. For completes, the resulting algorithm is included, see Alg. 3.2.

In our IDR variant we follow a degree ℓ minimization. If $s = 1$, then our variant is equivalent to the original BiCGstab(ℓ) algorithm in [5].

The polynomials P_k in CGS [8], GCGS [1], and GPBiCG [14] 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). Therefore, although P_k can be expressed as $P_k(\lambda) = \Pi_{j\leq k}(1 - \omega_j\lambda)$, these hybrid variants can not be described in the above Bi-CGSTAB fashion: the ω_j ($j \leq k$) change if k increases ($\omega_j = \omega_{j,k}$).

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

Alg. 3.2: **BiCGstab(ℓ)**. The $\mathbf{u}_j$ and $\mathbf{r}_j$ in the computation of 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. 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 $\mathbf{u} = [\mathbf{u}_0; \ldots; \mathbf{u}_j]$.

4. IDR. In this section, we formulate a variant of IDR that, in §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.

• In the first part (see §4.1), which we call *the IDR step*, we update a residual $\mathbf{r}_-$ in $\mathcal{G}_k$ (a primary residual) and an $n \times s$ matrix $\mathbf{A}\mathbf{U}_-$ with columns also in $\mathcal{G}_k$ to a residual $\mathbf{r}$ in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$ and an $n \times s$ matrix $\mathbf{A}\mathbf{U}$ with columns in $\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp}$. We use the subscript '$-$' as a reference to quantities that are available at the end of the past cycle, the kth cycle.
The IDR step corresponds to the Bi-CG part in one cycle of standard Bi-CGSTAB.

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

Suppose, at the beginning of the kth cycle, $n \times s$ 'update' matrices $\mathbf{U}_- = [\mathbf{u}_1, \ldots, \mathbf{u}_s]$ and $\mathbf{A}\mathbf{U}_- \equiv [\mathbf{A}\mathbf{u}_1, \ldots, \mathbf{A}\mathbf{u}_s]$ are available. In addition, we also have an approximate solution $\mathbf{x}_-$ and its associated residual $\mathbf{r}_- \equiv \mathbf{b} - \mathbf{A}\mathbf{x}_-$ (assuming exact arithmetic).

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

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

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 vectors $\mathbf{w}$.

With $\rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{r}_-$ and $\alpha = \sigma^{-1}\rho$, we generate $\mathbf{r}$ and $\mathbf{x}$ by

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

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 and we can form a new residual by determining the scalar ω that minimizes the norm of $\mathbf{r} - \omega\mathbf{A}\mathbf{r}$. This last expression is the new residual associated with the approximate solution $\mathbf{x} + \omega\mathbf{r}$. However, before we perform the minimization step, we first update $\mathbf{U}_-$ to $\mathbf{U}$, and $\mathbf{A}\mathbf{U}_-$ to $\mathbf{A}\mathbf{U}$ such that $\mathbf{A}\mathbf{U}$ is orthogonal to $\widetilde{\mathbf{R}}_0$ and we compute $\mathbf{A}^2\mathbf{U}$. We generate $\mathbf{A}\mathbf{U}$ 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 $\mathbf{A}^2\mathbf{U}$ and $\mathbf{U}$. For instance (for variants, see §4.4), if

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

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

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

followed by the projection:

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

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

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

The following scheme summarizes this IDR step

$$\begin{array}{ccccccccccc}
\mathbf{U}_- & & \mathbf{x}_- & \to & \mathbf{x} & & \mathbf{U}e_1 & & \mathbf{U}e_2 & \ldots & \mathbf{U}e_s \\
 & & & & & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{A}\mathbf{U}_- & & \mathbf{r}_- & \overset{\Pi_1}{\to} & \mathbf{r} & & \mathbf{A}\mathbf{U}e_1 & & \mathbf{A}\mathbf{U}e_2 & \ldots & \mathbf{A}\mathbf{U}e_s \\
 & & & & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
 & & & & \boxed{\mathbf{A}\mathbf{r}} & & \boxed{\mathbf{A}^2\mathbf{U}e_1} & & \boxed{\mathbf{A}^2\mathbf{U}e_2} & \ldots & \boxed{\mathbf{A}^2\mathbf{U}e_s}
\end{array} \tag{4.8}$$

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 $\mathbf{r}, \mathbf{A}\mathbf{U}e_1, \ldots \mathbf{A}\mathbf{U}e_s$, are orthogonal to $\widetilde{\mathbf{R}}_0$. To use IDR-terminology, if $\mathbf{r}_-$ and $\mathbf{A}\mathbf{U}_-$ belong to $\mathcal{G}_k$, then 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): the multiplication maps the vectors to a space $\mathcal{G}_{k+1}$. Of course, what $\mathcal{G}_{k+1}$ will be depends on the choice of μ_{k+1}: $\mathcal{G}_{k+1} \equiv (\mu_{k+1}\mathbf{I} - \mathbf{A})(\mathcal{G}_k \cap \widetilde{\mathbf{R}}_0^{\perp})$. The third row corresponds to $\mu_{k+1} = 0$, which will not be a practical choice. Nevertheless, this observation may clarify how this approach relates to IDR. In the next subsection, we discuss the choice $\mu_{k+1} = 1/\omega_{k+1}$.

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 $\mathbf{x}_+$, $\mathbf{r}_+$, $\mathbf{U}_+$ and $\mathbf{AU}_+$, respectively. Select a complex scalar ω_{k+1}. Then,

$$\mathbf{x}_+ \equiv \mathbf{x} + \omega_{k+1}\mathbf{r}, \quad \mathbf{r}_+ \equiv \mathbf{r} - \omega_{k+1}\mathbf{Ar},$$

$$\mathbf{U}_+ \equiv \mathbf{U} - \omega_{k+1}\mathbf{AU}, \quad \mathbf{AU}_+ \equiv \mathbf{AU} - \omega_{k+1}\mathbf{A}^2\mathbf{U}$$

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 $\mathbf{r}_+$ 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 [6].

NOTE 4.1. If, in Scheme (4.8), the column with $\mathbf{x}$, $\mathbf{r}$ and $\mathbf{Ar}$ is replaced by $\mathbf{x}_+$ and $\mathbf{r}_+$ ($\mathbf{r}_+$ at the location of $\mathbf{Ar}$, and the columns with $\mathbf{x}_-$ and $\mathbf{U}_-$ interchanged), 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., [4, 3]).

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

4.3. The relation to IDR. The following proposition expresses the fact that the above approach defines an IDR method. The proof has been indicated above when combined with Theorem 2.3: we leave details to the reader.

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

4.4. Alternative formulations of the IDR step. In the Scheme (4.8) above, we generated the basis vectors of $\mathcal{AU} \equiv \mathcal{K}_s(\Pi_1\mathbf{A}, \Pi_1\mathbf{Ar})$ by multiplication by the operator $\Pi_1\mathbf{A}$. With larger s, this approach will be unstable. More stable approaches, as Arnoldi, can be exploited. The matrix of basis vectors with such another approach is of the form $\mathbf{AU}T$, where T is some non-singular $s \times s$ matrix. For instance, with Arnoldi, we find the $n \times s$ orthonormal matrix $\mathbf{Q}$ from the QR-decomposition of $\mathbf{AU}$: $\mathbf{AU} = \mathbf{Q}R$, the $s \times s$ matrix R is upper triangular and $T = R^{-1}$. Herewith associated, we should compute $\widetilde{\mathbf{U}} \equiv \mathbf{U}T$ and $\mathbf{A}^2\widetilde{\mathbf{U}} \equiv \mathbf{A}^2\mathbf{U}T$ instead of $\mathbf{U}$ and $\mathbf{A}^2\mathbf{U}$. Then $\mathbf{Q} = \mathbf{A}\widetilde{\mathbf{U}}$. In practise, whenever we compute $\mathbf{A}\widetilde{\mathbf{U}}e_q$ by $\sum_{j<q}\gamma_j\mathbf{A}\widetilde{\mathbf{U}}e_j + \gamma_q\mathbf{AU}e_q$, we immediately compute $\widetilde{\mathbf{U}}e_q$ and $\mathbf{A}^2\widetilde{\mathbf{U}}e_q$ by $\sum_{j<q}\gamma_j\widetilde{\mathbf{U}}e_j + \gamma_q\mathbf{U}e_q$ and $\sum_{j<q}\gamma_j\mathbf{A}^2\widetilde{\mathbf{U}}e_j + \gamma_q\mathbf{A}^2\mathbf{U}e_q$, respectively, and we put the new values at the location of the old ones.

An orthogonal Krylov basis for $\mathcal{AU}$ is generated by s steps of ORTHODIR applied to the equation

$$\Pi_1\mathbf{Ay} = \mathbf{r} \tag{4.9}$$

with initial guess $\mathbf{0}$, or, equivalently, to the equation $\Pi_1\mathbf{A}\mathbf{y} = \mathbf{b}$ with initial guess $\mathbf{x}$. These basis vectors are the ORTHODIR direction vectors that are used to update the residuals. They are equal to the 'Arnoldi' vectors except for scalar multiples. Since $\Pi_1\mathbf{A} = \mathbf{A}\Pi_0$, solving (4.9) is equivalent to solving $\mathbf{A}\Pi_0\mathbf{y} = \mathbf{r}$. Hence, ORTHODIR can be used to update residuals as well as approximate solutions for the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ (with initial guess $\mathbf{x}$). Except for scalar multiples, and assuming that GCR does not break down, ORTHODIR and GCR produce the same orthogonal basis of direction vectors for $\mathcal{AU}$. The direction vectors are obtained in ORTHODIR by multiplication of the preceding direction vector by the operator, $\Pi_1\mathbf{A}$ in this case, whereas in GCR the direction vectors are obtained by multiplication of the present residual by the operator. In both methods, the new direction vectors are obtained then as the orthogonal components by applying Gram-Schmidt.

The IDR(s) variant that is described in [11] follows a different approach. This variant computes basis vectors for $\mathcal{AU}$ that satisfy bi-orthogonality relations with the columns of $\mathbf{R}_0$. This yields a stable and quite efficient IDR-implementation, and this approach can be followed here as well.

4.5. Saving vector updates in the IDR step. Note that for the computation of σ (cf., (4.1)) the matrix $\mathbf{AU}$ is not needed: σ can be computed as $(\mathbf{A}^*\widetilde{\mathbf{R}}_0)^*\mathbf{U}$. The matrix $\mathbf{A}^*\widetilde{\mathbf{R}}_0$ has to be computed once and stored. This allows us to following alternative for performing the IDR step.

With $\rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{r}_-$ and $\alpha = \sigma^{-1}\rho$, we generate $\mathbf{r}$ and $\mathbf{x}$ by first computing $\mathbf{u} \equiv \mathbf{U}_-\alpha$. Then, $\mathbf{r} \equiv \Pi_1\mathbf{r}_- = \mathbf{r}_- - \mathbf{A}\mathbf{u}$ and $\mathbf{x} \equiv \mathbf{x}_- + \mathbf{u}$, where $\mathbf{A}\mathbf{u}$ has been obtained by explicitly multiplying $\mathbf{u}$ by $\mathbf{A}$. As before, we obtain the vector $\mathbf{A}\mathbf{r}$ by matrix multiplication. Now, we update $\mathbf{U}_-$ to $\mathbf{U}$ and compute $\mathbf{AU}$, with $\mathbf{AU}$ as in (4.4), as follows. First, we compute $\mathbf{U}e_1$ and $\mathbf{AU}e_1$: $\mathbf{U}e_1 \equiv \mathbf{r} - \mathbf{U}_-\beta$, where $\beta = \sigma^{-1}((\mathbf{A}^*\widetilde{\mathbf{R}}_0)^*\mathbf{A}\mathbf{r})$. Then $\mathbf{AU}e_1$ is obtained by matrix-vector multiplication. Then, similarly, with $\mathbf{s} \equiv \mathbf{AU}e_q$ for some $q < s$, the vector $\mathbf{U}e_{q+1}$ is computed as $\mathbf{U}e_{q+1} \equiv \mathbf{s} - \mathbf{U}_-\beta$, where now $\beta = \sigma^{-1}(\widetilde{\mathbf{R}}_0^*\mathbf{s})$ and $\mathbf{AU}e_{q+1}$ is obtained by matrix-vector multiplication. The scheme below summarizes this approach.

$$
\begin{array}{lllllll}
\mathbf{U}_- & \mathbf{x}_- & \to \mathbf{x} & \mathbf{U}e_1 & \mathbf{U}e_2 & \dots & \mathbf{U}e_s \\
 & & \downarrow \mathbf{A} \quad \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} \quad \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
 & \mathbf{r}_- & \mathbf{r} & \mathbf{AU}e_1 & \mathbf{AU}e_2 & \dots & \mathbf{AU}e_s \\
 & & \downarrow \mathbf{A} & & & & \\
 & & \boxed{\mathbf{A}\mathbf{r}} & & & &
\end{array}
\tag{4.10}
$$

Note that this approach saves storage. We now have to store the following five $n \times s$ matrices ($\widetilde{\mathbf{R}}_0$, $\mathbf{A}^*\widetilde{\mathbf{R}}_0$, $\mathbf{U}_-$, $\mathbf{U}$, and $\mathbf{AU}$, or the four matrices $\widetilde{\mathbf{R}}_0$, $\mathbf{A}^*\widetilde{\mathbf{R}}_0$, $\mathbf{U}_-$, $\mathbf{U} - \omega\mathbf{AU}$), whereas before we had to store six matrices of this size ($\widetilde{\mathbf{R}}_0$, $\mathbf{U}_-$, $\mathbf{AU}_-$, $\mathbf{U}$, $\mathbf{AU}$, and $\mathbf{A}^2\mathbf{U}$, or, the five matrices $\widetilde{\mathbf{R}}_0$, $\mathbf{U}_-$, $\mathbf{AU}_-$, $\mathbf{U} - \omega\mathbf{AU}$, and $\mathbf{AU} - \omega\mathbf{A}^2\mathbf{U}$). We can save a bit on this: when the last columns of $\mathbf{U}$ and $\mathbf{AU}$ are available, the matrix $\mathbf{U}_-$ (and $\mathbf{AU}_-$) is not needed anymore.

As compared to the previous approach, there are $s^2 + s$ less vector updates needed per full recursion step, but instead, we need an additional matrix vector multiplication per full step.

If $s = 1$, then we can save this additional MV. Because, for the update $\mathbf{r} = \mathbf{r}_- - \mathbf{AU}_-\alpha$ we need $\mathbf{AU}_-$, which is one vector if $s = 1$. This vector can be reused

in the computation of $\mathbf{A}\mathbf{U}e_1 = \mathbf{A}(\mathbf{r} - \mathbf{U}_-\beta) = \mathbf{A}\mathbf{r} - \mathbf{A}\mathbf{U}_-\beta$: in case $s = 1$, both $\mathbf{A}\mathbf{r}$ and $\mathbf{A}\mathbf{U}_-$ are available at this stage.

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

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 make a residual $\mathbf{r}_-$, say, available with associated approximation $\mathbf{x}_-$, plus the vectors $\mathbf{A}\mathbf{r}_-$, ..., $\mathbf{A}^{j-1}\mathbf{r}_-$, and the $n \times s$ matrices $\mathbf{U}_-$, $\mathbf{A}\mathbf{U}_-$, $\mathbf{A}^2\mathbf{U}_-$, ..., $\mathbf{A}^j\mathbf{U}_-$.

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 $\mathbf{A}^i\mathbf{U}$ ($i = 0, \ldots, j$) such that the columns of $\mathbf{A}^j\mathbf{U}$ 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 $\mathbf{A}^i\mathbf{U}$ for $i = 0, \ldots, j-1, j+1$. For instance, with $\mathbf{s}_j \equiv \mathbf{A}^j\mathbf{U}e_q$ for some $q < s$, the vector $\mathbf{A}^j\mathbf{U}e_{q+1}$ can be computed as $\mathbf{A}\mathbf{s}_j$, which gives the qth column of $\mathbf{A}^{j+1}\mathbf{U}$, followed by the projection Π_j: $\mathbf{A}^j\mathbf{U}e_{q+1} \equiv \mathbf{A}\mathbf{s}_j - \mathbf{A}^j\mathbf{U}_-\sigma^{-1}\beta$, where $\beta \equiv \sigma^{-1}\rho$ and $\rho \equiv \widetilde{\mathbf{R}}_0^*\mathbf{A}\mathbf{s}_j$. Then the $(q+1)$th column of $\mathbf{A}^i\mathbf{U}$ can be computed as $\mathbf{A}^i\mathbf{U}e_{q+1} = \mathbf{s}_{i+1} - \mathbf{A}^i\mathbf{U}_-\beta$ ($i = 0, \ldots, j-1$), involving vector updates only (no additional MVs, no additional dot products). Here, $\mathbf{s}_i \equiv \mathbf{A}^i\mathbf{U}e_q$. The following scheme summarizes this IDR step in case $j = 2$.

$$\begin{array}{lllllllll}
\mathbf{U}_- & \mathbf{x}_- & \to & \mathbf{x} & & \mathbf{U}e_1 & & \mathbf{U}e_2 & \ldots & \mathbf{U}e_s \\
 & & & & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_0}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{A}\mathbf{U}_- & \mathbf{r}_- & \overset{\Pi_1}{\to} & \mathbf{r} & & \mathbf{A}\mathbf{U}e_1 & & \mathbf{A}\mathbf{U}e_2 & \ldots & \mathbf{A}\mathbf{U}e_s \\
 & & & & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_1}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
\mathbf{A}^2\mathbf{U}_- & \mathbf{A}\mathbf{r}_- & \overset{\Pi_2}{\to} & \mathbf{A}\mathbf{r} & & \mathbf{A}^2\mathbf{U}e_1 & & \mathbf{A}^2\mathbf{U}e_2 & \ldots & \mathbf{A}^2\mathbf{U}e_s \\
 & & & \downarrow \mathbf{A} & \overset{\Pi_2}{\nearrow} & \downarrow \mathbf{A} & \overset{\Pi_2}{\nearrow} & \downarrow \mathbf{A} & & \downarrow \mathbf{A} \\
 & & & \boxed{\mathbf{A}^2\mathbf{r}} & & \boxed{\mathbf{A}^3\mathbf{U}e_1} & & \boxed{\mathbf{A}^3\mathbf{U}e_2} & \ldots & \boxed{\mathbf{A}^3\mathbf{U}e_s}
\end{array} \tag{5.5}$$

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 last but one row of Scheme (5.5), the vectors $\mathbf{Ar}$, $\mathbf{A}^2\mathbf{U}e_1$, ..., $\mathbf{A}^2\mathbf{U}e_s$, are orthogonal to $\widetilde{\mathbf{R}}_0$, and, to use IDR-terminology, they belong to $\mathcal{G}_{k+j-1} \cap \widetilde{\mathbf{R}}_0^{\perp}$, if $\mathbf{Ar}_-$ and the columns of $\mathbf{A}^2\mathbf{U}_-$ belong to $\mathcal{G}_{k+j-1}$. Multiplication by $\mathbf{A}$ leads to the vectors on the last row of (5.5): they are mapped to a space $\mathcal{G}_{k+j}$ (assuming $\mu_j = 0$).

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 $\mathbf{A}^i\mathbf{U}$ to $\mathbf{A}^i\mathbf{U}_-$ $(i = 0, \ldots, j+1)$.

The variants of §4.2.1, §4.4, and §4.5 have their analogues for the present situation. In our actual implementation, we followed the Arnoldi variant of §4.4 to orthonormalize $\mathbf{A}^j\mathbf{U}$ (orthonormalizing the vectors at the last but one row of Scheme (5.5)).

5.2. Minimization using degree ℓ polynomials. After ℓ repetitions of the IDR step, we have a residual, say $\mathbf{r}$, available, plus the vectors $\mathbf{A}^i\mathbf{r}$ for $i = 0, \ldots, \ell$, the associated approximation $\mathbf{x}$ and $\ell+2$ matrices $\mathbf{A}^i\mathbf{U}$ $(i = 0, \ldots, \ell+1)$ of size $n \times s$. Now, to finish one cycle of IDRstab, determine scalars $\gamma_1, \ldots, \gamma_\ell$, for instance, such that the norm of

$$\mathbf{r}_+ \equiv \mathbf{r} - \gamma_1\mathbf{Ar} - \ldots - \gamma_\ell\mathbf{A}^\ell\mathbf{r} \tag{5.6}$$

is minimal. Compute

$$\mathbf{x}_+ \equiv \mathbf{x} + \gamma_1\mathbf{r} + \ldots + \gamma_\ell\mathbf{A}^{\ell-1}\mathbf{r} \tag{5.7}$$

and

$$\mathbf{A}^i\mathbf{U}_+ \equiv \mathbf{A}^i\mathbf{U} - \gamma_1\mathbf{A}^{i+1}\mathbf{U} - \ldots - \gamma_\ell\mathbf{A}^{i+\ell}\mathbf{U} \quad (i = 0, 1). \tag{5.8}$$

5.3. The relation to IDR. The following theorem allows IDRstab to view as an IDR method.

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

$$Q(\lambda) \equiv 1 - \gamma_1\lambda - \ldots - \gamma_\ell\lambda^\ell \qquad (\lambda \in \mathbb{C})$$

is zero. Suppose at the beginning of the cycle the residual $\mathbf{r}_-$ as well as the columns of the matrix $\mathbf{AU}_-$ belong to the Sonneveld subspace $\mathcal{S}(P_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$, with P_k a polynomial of exact degree k.[1] *Then the residual $\mathbf{r}_+$ and the columns of $\mathbf{AU}_+$ at the end of the cycle belong to $\mathcal{S}(QP_k, \mathbf{A}, \widetilde{\mathbf{R}}_0)$: $P_{k+\ell} \equiv QP_k$ is of exact degree $k + \ell$*

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

$$Q(\lambda) = 1 - \gamma_1\lambda - \ldots - \gamma_\ell\lambda^\ell = (1 - \omega_{k+1}\lambda) \cdot \ldots \cdot (1 - \omega_{k+\ell}\lambda)$$

and use induction to j $(j = 1, \ldots, \ell)$ adding one factor $Q_{j+1}(\lambda) \equiv Q_j(\lambda)(1 - \omega_{k+j}\lambda)$ per induction step; $Q_0 \equiv 1$. We leave details to the reader. ∎

The theorem states that the quantities of (5.6)–(5.8) would also have been obtained by performing ℓ steps of the IDR variant of §4 provided the scalars ω_{k+j} would

[1] The transition of $\mathbf{r}_-$ and $\mathbf{AU}_-$ at the beginning of the cycle to $\mathbf{r}_+$ at the end of the cycle moves through ℓ stages where $\mathbf{r}_-$ is updated to $\mathbf{r}$, $\mathbf{AU}_-$ to $\mathbf{AU}$ as explained in §5.1, and where $\mathbf{r}$ is renamed to $\mathbf{r}_-$ and $\mathbf{AU}$ to $\mathbf{AU}_-$. At the end $\mathbf{r}_+$, ... are formed as explained in §5.2.

```
Select an x_0 and an n × s matrix R̃_0.
Compute r_0 = b − Ax_0
% Generate U = [U_0; U_1] = [U_0; AU_0]
for q = 1, ..., s
   if q = 1, u_0 = r_0, else u_0 = u_1
   μ⃗ = (U_0 (:,1:q−1))* u_0, u = [u_0; Au_0] − U_(:,1:q−1) μ⃗
   u = u/‖u_0‖, U_(:,q) = u
end for
while ‖r_0‖ > tol
    for j = 1, ..., ℓ
       % an IDR step
       σ = R̃_0* U_j, α⃗ = σ^{-1}(R̃_0* r_{j−1})
       x = x + U_0 α⃗, r = r − [U_1; ...; U_j] α⃗, u = [r; Ar_{j−1}]
       for q = 1, ..., s
          if q = 1, u = r, else u = [u_1; ...; u_{j+1}]
          β⃗ = σ^{-1}(R̃_0* u_j), u = u − U β⃗, u = [u; Au_j]
          μ⃗ = (V_j (:,1:q−1))* u_j, u = u − V_(:,1:q−1) μ⃗
          u = u/‖u_j‖, V_(:,q) = u
       end for
       U = V
    end for
    % The polynomial step
    γ⃗ = (γ_1; ...; γ_ℓ) = minarg_γ⃗ ‖r_0 − [r_1, ..., r_ℓ] γ⃗‖
    x = x + [r_0, ..., r_{ℓ−1}] γ⃗, r = r_0 − [r_1, ..., r_ℓ] γ⃗
    U = [U_0 − Σ_{j=1}^{ℓ} γ_j U_j; U_1 − Σ_{j=1}^{ℓ} γ_j U_{j+1}]
end while
```

Alg. 5.3: **IDRstab**. The $\mathbf{U}_j$, $\mathbf{r}_{j-1}$, $\mathbf{u}_j$, and $\mathbf{V}_j$ in the computation of σ, $\vec{\alpha}$, $\vec{\beta}$, and $\vec{\mu}$, respectively, are related to $\mathbf{U}$, $\mathbf{r}$, $\mathbf{u}$, and $\mathbf{V}$ according to $\mathbf{U} = [\mathbf{U}_0; \ldots; \mathbf{U}_j]$, $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_{j-1}]$, $\mathbf{u} = [\mathbf{u}_0; \ldots; \mathbf{u}_j]$ and $\mathbf{V} = [\mathbf{V}_0; \ldots; \mathbf{V}_{j+1}]$, respectively. Note that the sizes of $\mathbf{r}$, $\mathbf{U}$ and $\mathbf{u}$ change during the loop: at the beginning of the '`for` $j = \ldots$' loop, $\mathbf{r} = [\mathbf{r}_0; \ldots; \mathbf{r}_{j-1}]$ and $\mathbf{U} = [\mathbf{U}_0; \ldots; \mathbf{U}_j]$.

have been appropriately selected in each polynomial step (cf. §4.2). Obviously, it is impossible in practice to select the ω_{k+j} 'appropriately' at step $j+k$ ($j < \ell$): the roots of the minimizing polynomial Q are not known at the this stage. In particular, in one cycle of BiCGstab(ℓ), we move from the primary residual $\mathbf{r}_k$ to the primary residual $\mathbf{r}_{k+\ell}$ and do not explicitly compute the intermediate residual $\mathbf{r}_{k+j}$ ($j = 1, \ldots, \ell - 1$). Nevertheless the theorem is of interest: since it shows that IDRstab is an IDR version.

5.4. The IDRstab algorithm. The procedure described in §5.1 and §5.2 leads to the algorithm in Alg. 5.3. The notation in this algorithm follows 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 $\mathbf{W}_{(:,q)} \equiv \mathbf{w}_q$ ($q \leq s$); $[\mathbf{U}_0; \ldots; \mathbf{U}_j] \equiv [\mathbf{U}_0^{\mathrm{T}}, \ldots, \mathbf{U}_j^{\mathrm{T}}]^{\mathrm{T}}$.

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

The line in the two '`for` $q = \ldots$' loops involving $\vec{\mu}$, form an implementation of Arnoldi's procedure to obtain an orthonormalized matrix $\mathbf{U}_0$ and $\mathbf{V}_i$ for $i = j$,

respectively. We use Arnoldi 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, as modified Gram–Schmidt or repeated Gram-Schmidt, may be required. We have good results with $s \leq 8$ and, to our experience, for this size of s classical Gram–Schmidt is sufficiently stable.
Arnoldi may be applied to obtain an orthonormalized matrix $\mathbf{V}_i$ for any $i = 0, \ldots, j+1$. When applied to $i = 1$, we are actually performing ORTHODIR without updating the ORTHODIR residuals and approximations. Of course these ORTHODIR quantities can be computed as well. This can be useful, for instance, to allow for an 'intermediate break' when an ORTHODIR residual is small also for $j < \ell$. Of course updating the ORTHODIR residuals and approximation for each $j = 1, \ldots, \ell$ requires additional vector updates. Hence, this strategy will be attractive only when the matrix vector multiplication form the dominant costs per step.

As in BiCGstab(ℓ), the $\vec{\gamma}$ can be computed as the solution of the normal equation

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

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 FOM) might lead to more stability in the IDR step (see [6]).

Assembling matrices 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 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 vectors updates in the IDR step that is used in Alg. 5.3. Implementation of this, may also lead to more efficiency on certain machines. By solving $\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}$ instead of $\mathbf{A}\mathbf{x} = \mathbf{b}$ and defining $\mathbf{r}_k \equiv \mathbf{b} + \mathbf{A}\mathbf{x}_k$, the two updates $\mathbf{x} = \mathbf{x} - \mathbf{U}_0\vec{\alpha}$ and $\mathbf{r} = \mathbf{r} - [\mathbf{U}_1, \ldots, \mathbf{U}_j]\vec{\alpha}$ can be further combined into one (higher dimensional) update: $[\mathbf{x}; \mathbf{r}] = [\mathbf{x}; \mathbf{r}] - \mathbf{U}\vec{\alpha}$.

6. Numerical experiments. This section presents numerical experiments on three test problems that, for different reasons, are difficult to solve by iterative methods. The difficulty of the first test problem is that the system matrix is highly indefinite, whereas the difficulty in the second and third problem comes form the fact that the system matrix is almost skew symmetric.

As an implementation independent measure for the amount of work (or solution time) to solve the systems we will take the number of matrix-vector operations. This measure is realistic for applications where the (preconditioned) matrix-vector multiplication is far more expensive than a vector operation, which is typically the case, and as long as the number of vector operations per matrix-vector operation is modest. To limit the overhead of the vector operations we only consider relatively small values for s, and ℓ, both ranging between 1 and 8.

The experiments are performed with MATLAB 7.5.

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.

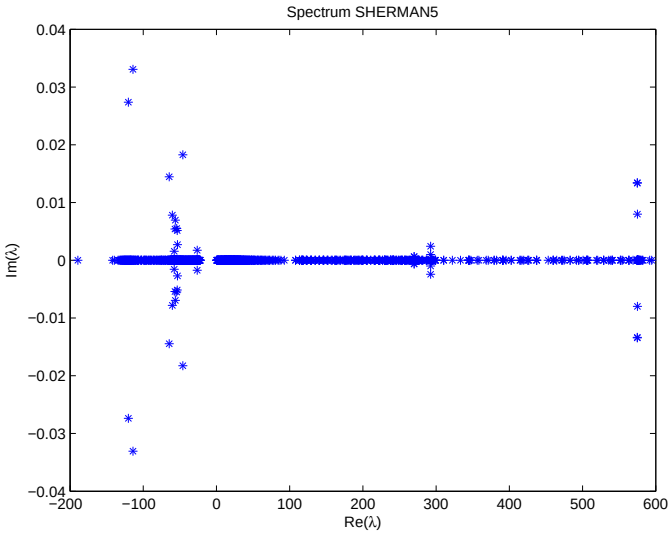

FIG. 6.1: Spectrum of SHERMAN5

The system is solved with IDRstab, with different combinations of the parameters s and l. For the 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}. Figure 6.2 shows the convergence behaviour for the following combinations of s and ℓ: $s = 4, \ell = 1$ (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 matrix-vector multiplications is to the optimal number of GMRES matrix-vector multiplications. For practical applications, where the typical problem size is orders of magnitudes 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 large overhead for vector operations. Table 6.1 gives for different combinations of s and ℓ the required number of matrix-vector multiplications to achieve the desired accuracy.

$s\backslash\ell$	1	2	4	8
1	n.c.	3617	3033	2529
2	2969	2546	2174	2066
4	2734	2154	1944	1884
8	2303	1970	1700	1808

TABLE 6.1: SHERMAN5: matrix-vector products for IDRstab

For $s = \ell = 1$ (i.e. Bi-CGSTAB), no convergence occurs within 4000 matrix-vector multiplications. For higher values of s or ℓ IDRstab always converges. Moreover, Table

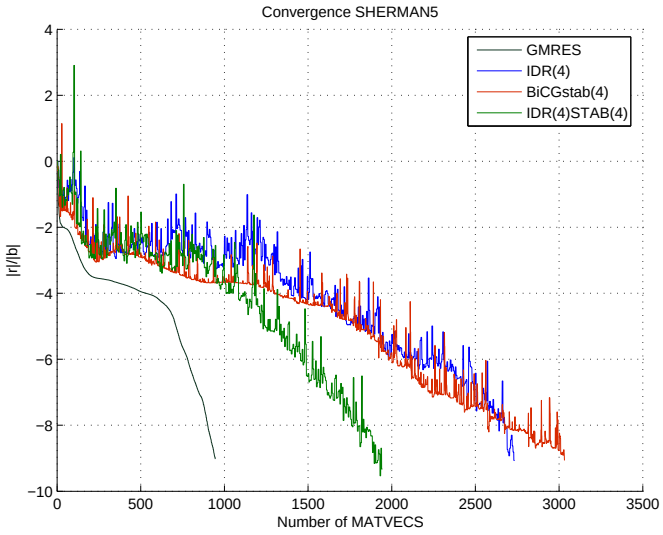

FIG. 6.2: SHERMAN5: convergence for IDRstab and for GMRES

6.1 shows that the number of matrix-vector multiplications is reduced by increasing either s, or ℓ, and, more importantly, that by increasing *both* s and ℓ the number of matrix-vector multiplications is reduced to a level lower than is achieved by IDR(s) and BiCGstab(ℓ).

6.2. A 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 discretised 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 [6].

As in the previous example, $\widetilde{\mathbf{R}}_0$ is taken to be the orthogonalization of s real random vectors, no preconditioner is applied, and the iterative processes are terminated if the relative residual norm drops below 10^{-9}. Figure 6.3 shows the convergence of three IDRstab variants, and for comparison also of full GMRES. This figure shows that also for this problem a significant gain can be achieved by taking 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 matrix-vector multiplications to achieve the desired accuracy.

Also for this example, Bi-CGSTAB does not converge within 4000 matrix-vector multiplications. 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 taking s higher. BiCGstab(ℓ) convergences after around 600 matrix-vector for all values of ℓ larger than 1. The most important conclusion is that by combining IDR and BiCGstab(ℓ), i.e.

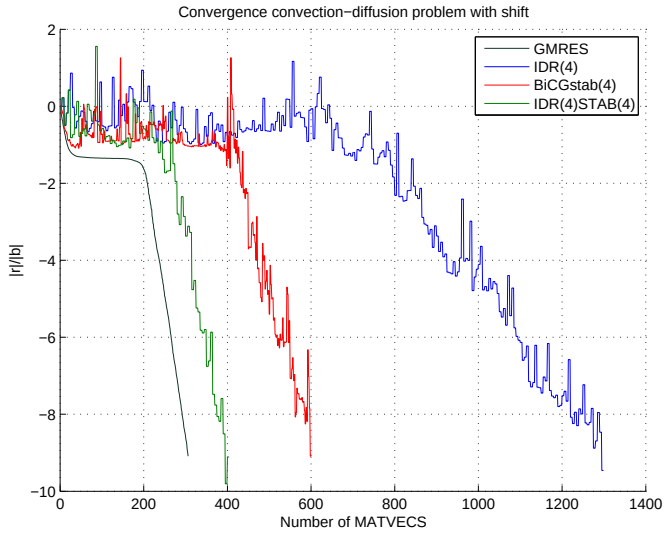

FIG. 6.3: 2D Convection-diffusion problem: convergence for IDRstab and for GMRES

$s\backslash\ell$	1	2	4	8
1	n.c.	625	601	625
2	1718	464	458	458
4	1299	434	404	404
8	917	386	368	440

TABLE 6.2: 2D convection-diffusion problem: matrix-vector products for IDRstab

taking both s and ℓ larger than 1, it is possible to almost close the gap with GMRES, and almost get optimal convergence.

6.3. A 3D convection-dominated problem. The third test problem is taken from [5], and is also included in [9]. This problem was use to illustrate that Bi-CGSTAB does not work well, due to the strong nonsymmetry of the system matrix. As was shown in [9], IDR(s) also performs poorly for this problem (if the shadow space is chosen real, the bad convergence can be overcome by chosing the shadow space complex).

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

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

Homegeneous 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 gridpoints in each directions (including boundary points) which yields a system of 125,000 equations.

As in the previous examples, $\widetilde{\mathbf{R}}_0$ is taken to be the orthogonalization of s real random vectors, no preconditioner is applied, and the iterative processes are terminated

if the relative residual norm drops below 10^{-9}. Figure 6.4 shows the convergence behaviour of three IDRstab variants and of GMRES.

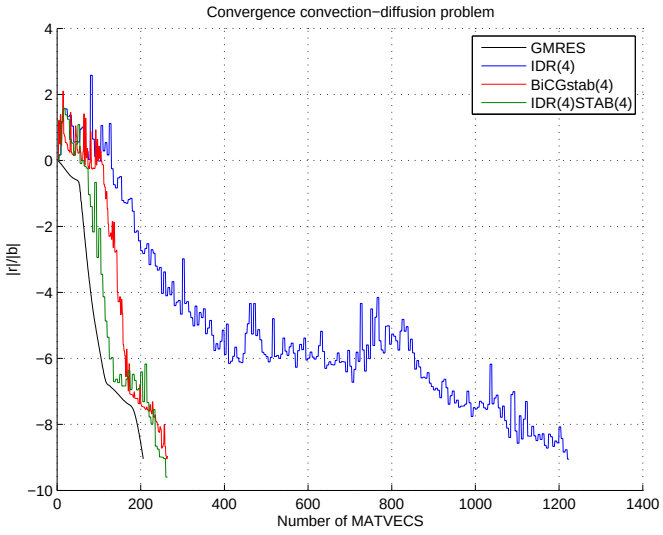

FIG. 6.4: 3D convection-diffusion problem: Convergence of IDRstab and of GMRES

The figure 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 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 remarks that the number of matrix-vector multiplications for $\ell = 2, s = 8$ is smaller than for the four BiCGstab(ℓ) variants that we investigated. Also for this example, Bi-CGSTAB does not converge within 4000 matrix-vector multiplications.

$s\backslash\ell$	1	2	4	8
1	n.c.	321	265	321
2	2087	266	266	266
4	1224	264	264	284
8	638	242	260	296

TABLE 6.3: 3D convection-diffusion problem: matrix-vector products for IDRstab

7. Conclusions. The goal of the research that is described in this paper was to unify two of the most powerful short recursion Krylov method for nonsymmetric systems into one method that inherits the strong points of both. To this end we have derived the method IDRstab. We have illustrate 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 [9], it was assumed that a new IDR-theorem was needed to make an IDR-method that uses

higher order stabilisation polynomials. In this paper, we have shown that this is not the case: any hybrid Bi-CG methods can be considered as an IDR method.

In the algorithm that we have presented in this paper many specific choices were made. The resulting implementation is in our experience efficient and numerically stable. However, we have no doubt that the method can be further optimised. To this end we plan to apply similar ideas as in [11].

Recently we received a private communication that a group at the University of Tokyo are also developing a generalisation of IDR(s) and BiCGstab(ℓ).

Acknowledgements: We thank Peter Sonneveld for introducing us to IDR and sharing his deep insight with us.
Part of this research has been funded by the Dutch BSIK/BRICKS project.

REFERENCES

[1] Diederik R. Fokkema, Gerard L. G. Sleijpen, and Henk A. van der Vorst, *Generalized conjugate gradient squared*, J. Comput. Appl. Math. **71** (1996), no. 1, 125–146. MR 97h:65040

[2] Martin H. Gutknecht, *Variants of BICGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput. **14** (1993), no. 5, 1020–1033. MR 1232173

[3] Martin H. Gutknecht and Klaus J. Ressel, *Look-ahead procedures for Lanczos-type product methods based on three-term Lanczos recurrences*, SIAM J. Matrix Anal. Appl. **21** (2000), no. 4, 1051–1078 (electronic). MR 1 745 519

[4] Klaus J. Ressel and Martin H. Gutknecht, *QMR smoothing for Lanczos-type product methods based on three-term recurrences*, SIAM J. Sci. Comput. **19** (1998), no. 1, 55–73 (electronic), Special issue on iterative methods (Copper Mountain, CO, 1996). MR 99d:65143

[5] Gerard L. G. Sleijpen and Diederik R. Fokkema, *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal. **1** (1993), 11–32 (electronic only). MR 94g:65038

[6] Gerard L. G. Sleijpen and Henk A. van der Vorst, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numer. Algorithms **10** (1995), no. 3-4, 203–223. MR 96g:65037

[7] Gerard L.G Sleijpen, Peter Sonneveld, and Martin B. van Gijzen, *Bi-CGSTAB as an induced dimension reduction method*, Preprint 1369, Department of Mathematics, Utrecht University, Utrecht, the Netherlands, 2008.

[8] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput. **10** (1989), 36–52.

[9] Peter Sonneveld and Martin 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), no. 2, 1035–1062.

[10] Henk 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), no. 2, 631–644. MR 92j:65048

[11] Martin van Gijzen and Peter Sonneveld, *An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties.* Delft University of Technology, Reports of the Department of Applied Mathematical Analysis, Report 08-21, 2008

[12] Piet Wesseling and Peter Sonneveld, *Numerical experiments with a multiple grid- and a preconditioned lanczos type method*, Lecture Notes in Mathematics, vol. 771, pp. 543–562, Springer Verlag, Berlin, Heidelberg, New York, 1980.

[13] Man-Chung Yeung and Tony F. Chan, *Ml(k)bicgstab: A bicgstab variant based on multiple lanczos starting vectors*, SIAM Journal on Scientific Computing **21** (1999), no. 4, 1263–1290.

[14] Shao-Liang Zhang, *GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems*, SIAM J. Sci. Comput. **18** (1997), no. 2, 537–551. MR 97i:65056