

# IDR( $s$ ) AND IDR( $s$ )EIG IN PARALLEL COMPUTING\*

JENS-PETER M. ZEMKE<sup>†</sup>

**Abstract.** In this short summary of a talk held at the University of Tokyo on the 12th February 2010 we consider a certain class of Krylov subspace methods: the IDR methods by Peter Sonneveld, and their recent generalizations to incorporate more shadow vectors, the IDR( $s$ ) methods. We sketch the main ideas behind IDR, IDR( $s$ ), and IDR( $s$ )EIG, and indicate possible approaches for their parallelization.

**Key words.** IDR; PIA; IDR( $s$ ); IDR( $s$ )EIG; Krylov subspace methods; Eigenvalues; Linear Systems; Parallelization

**AMS subject classifications.** 65F10 (primary), 65F15, 65Y05, 65Y20

**1. IDR and IDR( $s$ ).** IDR [16, 17, 24] and IDR( $s$ ) [15, 18, 23, 19, 13, 14, 20, 11] are Krylov subspace methods which have been invented by numerical analyst Peter Sonneveld at the TU Delft in 1976 and 2006, respectively, the latter with co-author Martin van Gijzen. Recently there has been an increased interest in these methods, as they appear to be more stable than other transpose-free Krylov subspace methods based on short recurrences in certain situations. An important predecessor of the new family IDR( $s$ ) is the less well known method ML( $k$ )BiCGSTAB by Man-Chung Yeung and Tony Chan [26].

**1.1. Krylov subspace methods.** IDR and IDR( $s$ ) are *Krylov subspace methods*. The  $m$ th **Krylov subspace**  $\mathcal{K}_m$  is defined for a given square matrix  $\mathbf{A} \in \mathbb{C}^{n \times n}$  and a starting vector  $\mathbf{q} \in \mathbb{C}^n$  as follows,

$$\mathcal{K}_m(\mathbf{A}, \mathbf{q}) := \text{span} \{ \mathbf{q}, \mathbf{A}\mathbf{q}, \dots, \mathbf{A}^{m-1}\mathbf{q} \}. \quad (1.1)$$

As long as the Krylov subspace  $\mathcal{K}_m$  has full dimension  $\dim(\mathcal{K}_m) = m$ , there is a natural isomorphism

$$\mathbf{v} \in \mathcal{K}_m \quad \Leftrightarrow \quad \mathbf{v} = \nu(\mathbf{A})\mathbf{q}$$

between vectors  $\mathbf{v}$  in a Krylov subspace and polynomials  $\nu \in \mathcal{P}_{m-1}$ . Obviously there is a last index  $m \leq n$  with  $\dim(\mathcal{K}_m) = m$ , this index is known as the grade of the vector  $\mathbf{q}$  with respect to the matrix  $\mathbf{A}$ . The Krylov matrices

$$\mathbf{K}_m := \begin{pmatrix} \mathbf{q}, \mathbf{A}\mathbf{q}, \mathbf{A}^2\mathbf{q}, \dots, \mathbf{A}^{m-1}\mathbf{q} \end{pmatrix} \quad (1.2)$$

satisfy the matrix recurrence

$$\begin{pmatrix} \mathbf{q}, \mathbf{A}\mathbf{K}_m \end{pmatrix} = \mathbf{K}_{m+1}. \quad (1.3)$$

---

\*Based in part on the forthcoming paper [5]. The author acknowledges the invitation by Prof. Seiji Fujino of the Research Institute for Information Technology of Kyushu University to Japan in February 2010 where part of this work was written.

<sup>†</sup>Institut für Numerische Simulation, Technische Universität Hamburg-Harburg, D-21073 Hamburg, Germany (zemke@tu-harburg.de).

The  $m$ th Krylov matrix spans a basis of the  $m$ th Krylov space  $\mathcal{K}_m$  if and only if  $m$  is less or equal to the grade of  $\mathbf{q}$ . We assume here that this is always the case. Suppose now that we choose upper triangular basis transformations  $\mathbf{K}_m =: \mathbf{Q}_m \mathbf{R}_m$ ,

$$\left( \mathbf{q}, \mathbf{A} \mathbf{Q}_m \mathbf{R}_m \right) = \mathbf{Q}_{m+1} \mathbf{R}_{m+1}. \quad (1.4)$$

The Krylov matrix recurrence (1.3) can now be reformulated,

$$\left( \mathbf{q}, \mathbf{A} \mathbf{Q}_m \right) = \mathbf{Q}_{m+1} \mathbf{R}_{m+1} \begin{pmatrix} 1 & \mathbf{o}^\top \\ \mathbf{o} & \mathbf{R}_m \end{pmatrix}^{-1}. \quad (1.5)$$

These upper triangular basis transformations are natural in the context of Krylov subspace methods, as in step  $k$  only the first  $k$  columns are available. When we strip off the first column on both sides of Eqn. (1.5), we exhibit the intimate connection of Krylov subspace methods to so-called Hessenberg decompositions.

Let the matrix  $\underline{\mathbf{C}}_m \in \mathbb{C}^{(m+1) \times m}$  be defined by

$$\begin{pmatrix} \star & \underline{\mathbf{C}}_m \\ \mathbf{o} & \end{pmatrix} := \mathbf{R}_{m+1} \begin{pmatrix} 1 & \mathbf{o}^\top \\ \mathbf{o} & \mathbf{R}_m \end{pmatrix}^{-1}. \quad (1.6)$$

By the group structure of regular upper triangular matrices, this matrix is unreduced extended Hessenberg.

In every simple Krylov subspace method, i.e., excluding block variants, we end up with a so-called Hessenberg decomposition<sup>1</sup>

$$\mathbf{A} \mathbf{Q}_m = \mathbf{Q}_{m+1} \underline{\mathbf{C}}_m =: \mathbf{Q}_m \mathbf{C}_m + \mathbf{q}_{m+1} \mathbf{c}_{m+1,m} \mathbf{e}_m^\top, \quad (1.7)$$

where  $\mathbf{C}_m$  is unreduced Hessenberg and measures the ‘‘ratio’’ of the basis transformations.

Krylov subspace methods can be classified into distinct approaches. We first give the matrix based classification. In the following, we always assume that in the context of the approximate solution of linear systems, the starting vector  $\mathbf{q}_1$  has been chosen by  $\|\mathbf{r}_0\|_2 \mathbf{q}_1 = \mathbf{r}_0$ . There are three well-known approaches based on such Hessenberg decompositions, namely,

**QOR:** approximate  $\mathbf{x} = \mathbf{A}^{-1} \mathbf{r}_0$  by  $\underline{\mathbf{x}}_m := \mathbf{Q}_m \mathbf{C}_m^{-1} \mathbf{e}_1 \|\mathbf{r}_0\|_2$ ,

**QMR:** approximate  $\mathbf{x} = \mathbf{A}^{-1} \mathbf{r}_0$  by  $\underline{\mathbf{x}}_m := \mathbf{Q}_m \underline{\mathbf{C}}_m^\dagger \mathbf{e}_1 \|\mathbf{r}_0\|_2$ ,

**Ritz-Gal erkin:** approximate part of  $\mathbf{J} = \mathbf{V}^{-1} \mathbf{A} \mathbf{V}$  by  $\underline{\mathbf{J}}_m := \mathbf{S}_m^{-1} \mathbf{C}_m \mathbf{S}_m$   
and part of  $\mathbf{V}$  by  $\underline{\mathbf{V}}_m := \mathbf{Q}_m \mathbf{S}_m$ , where  $\mathbf{C}_m \mathbf{S}_m = \mathbf{S}_m \mathbf{J}_m$ .

To every method from one class corresponds a method of the other. This fact is used in the forthcoming publication [5] to compute eigenvalues using IDR.

It turns out to be helpful to look at the corresponding polynomial description. We already mentioned that Krylov subspace methods compute elements from the Krylov subspace  $\mathcal{K}_m$  which can also be described as polynomials. The classification

<sup>1</sup>In [5] we name these relations in honor of Karl Hessenberg. He was to our knowledge the first who considered relations of the type  $\mathbf{A} \mathbf{Q}_n = \mathbf{Q}_{n+1} \underline{\mathbf{H}}_n$  with a special unreduced extended Hessenberg matrix  $\underline{\mathbf{H}}_n$ , see [7]. Usually the names of Lanczos [8, 9] or Arnoldi [1] are associated with such relations.

of Krylov methods given above can be rephrased using the language of polynomials. The three classes of methods can be described using certain polynomials and polynomial interpolation, cf. [27, 28] for the general perturbed case. The Ritz values are the eigenvalues of the Hessenberg matrix  $\mathbf{C}_m$  and the harmonic Ritz values are the eigenvalues of the matrix  $(\underline{\mathbf{C}}_m^\dagger \underline{\mathbf{I}}_m)^{-1}$ , where  $\underline{\mathbf{I}}_m$  denotes an extended identity matrix which has an additional row of zeros attached to the bottom. The three approaches implicitly compute polynomials with the following characteristics.

**QOR:**  $\mathbf{r}_m = \mathcal{R}_m(\mathbf{A})\mathbf{r}_0$ , where  $\mathcal{R}_m(z) := \det(\mathbf{I}_m - z\mathbf{C}_m^{-1})$ ,  
 $\mathbf{x}_m = \mathcal{L}_m[z^{-1}](\mathbf{A})\mathbf{r}_0$ , where  $\mathcal{L}_m[z^{-1}](z)$  interpolates  
the function  $z^{-1}$  at the Ritz values,

**QMR:**  $\underline{\mathbf{r}}_m = \underline{\mathcal{R}}_m(\mathbf{A})\mathbf{r}_0$ , where  $\underline{\mathcal{R}}_m(z) := \det(\mathbf{I}_m - z\underline{\mathbf{C}}_m^\dagger \underline{\mathbf{I}}_m)$ ,  
 $\underline{\mathbf{x}}_m = \underline{\mathcal{L}}_m[z^{-1}](\mathbf{A})\mathbf{r}_0$ , where  $\underline{\mathcal{L}}_m[z^{-1}](z)$  interpolates  
the function  $z^{-1}$  at the harmonic Ritz values,

**Ritz-Galérkin:** Unscaled Ritz vectors are given by  $\mathbf{v}_j^{(m)} = \mathcal{A}_m(\theta_j, \mathbf{A})\mathbf{q}_1$ ,  
where  $\mathcal{A}_m(\theta, z) := (\chi_m(\theta) - \chi_m(z))(\theta - z)^{-1}$ ,  $\theta \neq z$ ,  
 $\mathbf{C}_m \mathbf{s}_j = \mathbf{s}_j \theta_j$  and  $\chi_m(z) := \det(z\mathbf{I}_m - \mathbf{C}_m)$ .

These polynomial descriptions play a prominent rôle in determining the rate of convergence and understanding the behavior of Krylov subspace methods in finite precision or subject to more general perturbations, e.g., those, which are introduced by an inexact matrix vector product.

**1.2. 1976–1980: IDR.** In 1976 Sonneveld experimentally observed that for  $\mathbf{B} \in \mathbb{C}^{n \times n}$  and a given starting vector  $\mathbf{f}_0 \in \mathbb{C}^n$  and  $\mathbf{f}_1 := \mathbf{B}\mathbf{f}_0$  the following three-term recurrence

$$\mathbf{f}_{k+1} := \mathbf{B}(\mathbf{f}_k - \gamma_k(\mathbf{f}_k - \mathbf{f}_{k-1})), \quad \gamma_k := \frac{\mathbf{p}^H \mathbf{f}_k}{\mathbf{p}^H (\mathbf{f}_k - \mathbf{f}_{k-1})}$$

almost always terminates after  $2n$  steps with the zero vector  $\mathbf{f}_{2n} = \mathbf{o}_n$  [16, 17].

Analyzing this startling behavior, he discovered that the two consecutive vectors  $\mathbf{f}_{2j}, \mathbf{f}_{2j+1}$  constructed in this manner live in spaces  $\mathcal{G}_j$  of shrinking dimensions. These spaces are nowadays known as ‘‘Sonneveld spaces’’. He called this property ‘‘Induced Dimension Reduction’’ (IDR), and algorithms like the given three-term recurrence ‘‘IDR Algorithms’’. The theorem he proved was the first ‘‘IDR Theorem’’.

Sonneveld first made numerical experiments and then gave a rigorous proof for his first IDR Theorem. It is easy to see that apart from the first two arbitrarily chosen residuals the constructed residuals are in the  $\mathbf{B}$  image of the space  $\mathcal{S} := \mathbf{p}^\perp$ . The same argument proves that in general (observe that the first two residuals  $\mathbf{f}_0, \mathbf{f}_1$  are usually not in  $\mathcal{S}$ ) for  $k \geq 1$

$$\mathbf{f}_{2k}, \mathbf{f}_{2k+1} \in \mathcal{G}_k := \bigcap_{j=1}^k \mathbf{B}^j(\mathcal{S}) = \left( \bigoplus_{j=1}^k \mathbf{B}^{-jH} \{\mathbf{p}\} \right)^\perp = \left( \mathcal{K}_k(\mathbf{B}^{-H}, \mathbf{B}^{-H} \mathbf{p}) \right)^\perp.$$

Sonneveld proved that the dimensions of the spaces constructed are shrinking. This is the essence of the first IDR Theorem. He did not use the description as an orthogonal complement of a Krylov subspace as it is done here and elsewhere [13, 11]. We remark that generically  $\dim(\mathcal{K}_n(\mathbf{B}^{-H}, \mathbf{B}^{-H} \mathbf{p})) = n$ . Using the Krylov subspace point of view and the explicit orthogonalization against  $\mathbf{p}$  before multiplication with  $\mathbf{B}$ , we see that indeed  $\mathbf{f}_{2n} = \mathbf{B}\mathbf{o}_n = \mathbf{o}_n$ .

This IDR Theorem can be turned into many IDR Algorithms. The three-term recurrence

$$\mathbf{f}_{k+1} = \mathbf{B}(\mathbf{f}_k - \gamma_k(\mathbf{f}_{k-1} - \mathbf{f}_k)), \quad \text{where} \quad \gamma_k = \frac{\mathbf{p}^H \mathbf{f}_k}{\mathbf{p}^H(\mathbf{f}_{k-1} - \mathbf{f}_k)},$$

is an “implementation” of the Induced Dimension Reduction (IDR) Theorem. The vectors constructed live in spaces of shrinking dimensions. Methods like this are called “IDR Algorithms”.

Another implementation by Sonneveld can be used to solve “genuine” linear systems. The idea is to rewrite the linear system to Richardson’s iteration form,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b} =: \mathbf{B}\mathbf{x} + \mathbf{b}.$$

The classical Richardson’s iteration with a starting guess  $\mathbf{x}_0$  is then given by

$$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}_k + \mathbf{b}.$$

With  $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ , the Richardson’s iteration is carried out as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{r}_k, \quad \mathbf{r}_{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{r}_k.$$

In a Richardson-type IDR Algorithm, the second equation is replaced by the update

$$\mathbf{r}_{k+1} = (\mathbf{I} - \mathbf{A})(\mathbf{r}_k + \gamma_k(\mathbf{r}_k - \mathbf{r}_{k-1})), \quad \gamma_k = \frac{\mathbf{p}^H \mathbf{r}_k}{\mathbf{p}^H(\mathbf{r}_{k-1} - \mathbf{r}_k)}.$$

The update of the iterates has to be modified accordingly,

$$\begin{aligned} -\mathbf{A}(\mathbf{x}_{k+1} - \mathbf{x}_k) &= \mathbf{r}_{k+1} - \mathbf{r}_k = (\mathbf{I} - \mathbf{A})(\mathbf{r}_k + \gamma_k(\mathbf{r}_k - \mathbf{r}_{k-1})) - \mathbf{r}_k \\ &= (\mathbf{I} - \mathbf{A})(\mathbf{r}_k - \gamma_k \mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1})) - \mathbf{r}_k \\ &= -\mathbf{A}(\mathbf{r}_k + \gamma_k(\mathbf{I} - \mathbf{A})(\mathbf{x}_k - \mathbf{x}_{k-1})) \\ \Leftrightarrow \mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{r}_k + \gamma_k(\mathbf{I} - \mathbf{A})(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= \mathbf{r}_k + \gamma_k(\mathbf{x}_k - \mathbf{x}_{k-1} + \mathbf{r}_k - \mathbf{r}_{k-1}). \end{aligned}$$

Sonneveld termed the outcome the “Primitive IDR Algorithm” [16]. For simplicity we will refer to it as PIA. A very simple implementation of PIA is given in [Algorithm 1](#).

The algorithm can be restated with less memory requirements when we allow to overwrite the vectors and scalars. This variant can be found in [Algorithm 2](#). In [Figure 1.1](#) we compare Richardson’s iteration and PIA. These pictures gives some impressions of the “finite termination” property at step  $2n$  and a feeling for the acceleration of Richardson’s iteration by PIA in finite precision.

Sonneveld never did use PIA, as he considered it to be too unstable, instead he went on with a corresponding acceleration of the Gauß-Seidel method. In [17] he terms this method “Accelerated Gauß-Seidel” (AGS) and refers to it as “[t]he very first IDR-algorithm [...]”, see page 6, Ibid. This part of the story took place “in the background” in the year 1976. In September 1979 Sonneveld did attend the [IUTAM Symposium on Approximation Methods for Navier-Stokes Problems](#) in Paderborn, Germany. At this symposium he presented a new variant of IDR based on a variable splitting  $\mathbf{I} - \omega_j \mathbf{A}$ , where  $\omega_j$  is fixed for two steps and otherwise could be chosen

```

input :  $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, \mathbf{p}$ 
output:  $\mathbf{r}_k, \mathbf{x}_k, \gamma_k$ 
1  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ 
2  $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{r}_0$ 
3  $\mathbf{r}_1 = \mathbf{r}_0 - \mathbf{A}\mathbf{r}_0$ 
4 for  $k = 1, 2 \dots$  do
5    $\gamma_k = \mathbf{p}^\top \mathbf{r}_k / \mathbf{p}^\top (\mathbf{r}_{k-1} - \mathbf{r}_k)$ 
6    $\mathbf{s}_k = \mathbf{r}_k + \gamma_k (\mathbf{r}_k - \mathbf{r}_{k-1})$ 
7    $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k (\mathbf{x}_k - \mathbf{x}_{k-1}) + \mathbf{s}_k$ 
8    $\mathbf{r}_{k+1} = \mathbf{s}_k - \mathbf{A}\mathbf{s}_k$ 
9 end

```

**Algorithm 1:** The most simple variant of PIA

```

input :  $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, \mathbf{p}$ 
output:  $\mathbf{r}, \mathbf{x}, \gamma$ 
1  $\mathbf{x}_{\text{old}} \leftarrow \mathbf{x}_0$ 
2  $\mathbf{r}_{\text{old}} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_{\text{old}}$ 
3  $\mathbf{x}_{\text{new}} \leftarrow \mathbf{x}_{\text{old}} + \mathbf{r}_{\text{old}}$ 
4  $\mathbf{r}_{\text{new}} \leftarrow \mathbf{r}_{\text{old}} - \mathbf{A}\mathbf{r}_{\text{old}}$ 
5 while not converged do
6    $\gamma \leftarrow \mathbf{p}^\top \mathbf{r}_{\text{new}} / \mathbf{p}^\top (\mathbf{r}_{\text{old}} - \mathbf{r}_{\text{new}})$ 
7    $\mathbf{s} \leftarrow \mathbf{r}_{\text{new}} + \gamma (\mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}})$ 
8    $\mathbf{x}_{\text{tmp}} \leftarrow \mathbf{x}_{\text{new}} + \gamma (\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}}) + \mathbf{s}$ 
9    $\mathbf{r}_{\text{tmp}} \leftarrow \mathbf{s} - \mathbf{A}\mathbf{s}$ 
10   $\mathbf{x}_{\text{old}} \leftarrow \mathbf{x}_{\text{new}}, \mathbf{x}_{\text{new}} \leftarrow \mathbf{x}_{\text{tmp}}$ 
11   $\mathbf{r}_{\text{old}} \leftarrow \mathbf{r}_{\text{new}}, \mathbf{r}_{\text{new}} \leftarrow \mathbf{r}_{\text{tmp}}$ 
12 end

```

**Algorithm 2:** A better implementation of PIA

freely, but non-zero. This algorithm with a minimization of every second residual to determine the new  $\omega_j$  is included in the proceedings from 1980 [24]. The connection to Krylov methods, e.g., to BICG and Lanczos, is also given there. We refer to it as “Classical IDR” or simply “IDR”. The pseudo-code for the algorithm of IDR is given in Algorithm 3.

Algorithm 3 is the original IDR Algorithm from page 551 of [24]. It uses ORTHORES(1) in the first step and a residual minimization every second step. The residuals are the vectors  $-\mathbf{f}_{2j}$ . The finite termination property follows from a generalization of the IDR Theorem based on the commutativity of the linear polynomials  $\mathbf{I} - \omega_j \mathbf{A}$ . A numerical comparison of Richardson’s iteration, original IDR, and PIA is given in Figure 1.2.

As is obvious from the derivation of PIA, AGS, and IDR, there are many brothers of classical and primitive IDR. In 1976 Sonneveld already considered the acceleration of Gauß-Seidel (AGS). Similarly, the IDR philosophy can be used as an accelerator for the other classical splitting methods like Jacobi, SOR, SSOR, or Chebyshev, or even more general semi-iterative methods. Thus, IDR is a whole family of methods instead of a single instant. We remark that the well known method BICGSTAB [22, 21] is

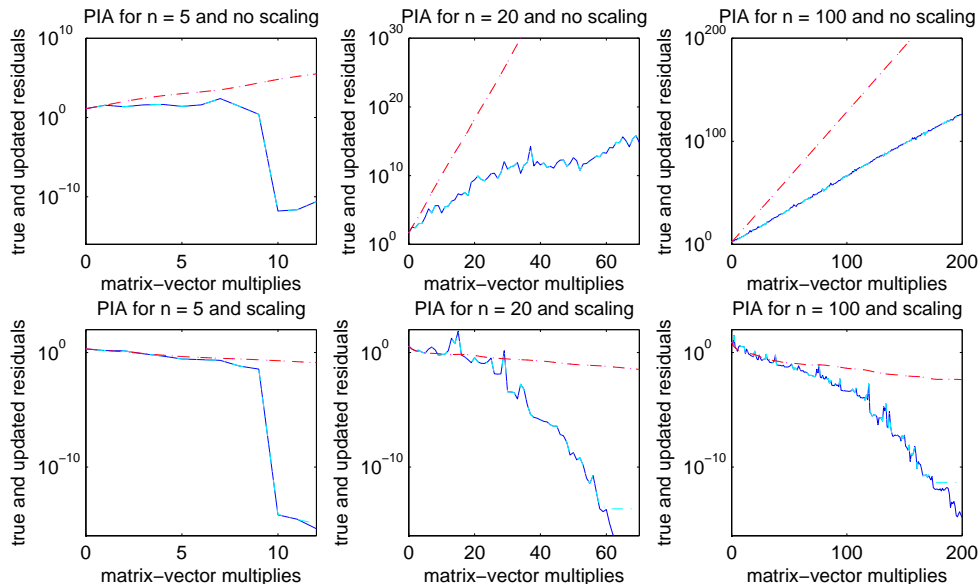


FIGURE 1.1. A numerical comparison of PIA and Richardson's iteration. The convergence curves of PIA are in all cases, after some possible initial erratic behavior, below the convergence curves of Richardson's iteration. For small  $n$ , PIA manages to converge when Richardson's iteration actually diverges.

mathematically merely a reformulation of IDR and thus also an IDR algorithm.

Some of the methods based on IDR acceleration of classical splitting and semi-iterative methods have been considered much more recently by Seiji Fujino et al. under the names ISOR, IJacobi, IGS. The generalization of these methods to incorporate more than one shadow vector seems very promising on distributed memory computers. One has to look careful at the difference between preconditioning and using a variable splitting before applying IDR acceleration or afterwards. The numerical behavior is not very promising. But this picture changes, when we use more shadow vectors . . .

**1.3. 2006–2010: IDR( $s$ ).** In 2006 the rebirth of IDR took place. Almost exactly 30 years after inventing IDR, Peter Sonneveld together with Martin van Gijzen reconsidered IDR and introduced more shadow vectors: IDR( $s$ ) was born. They came up with a variant called IDR( $s$ ) that used orthogonalization against a larger space, where  $s$  denotes the dimension of that space. Algorithmically, the transition from IDR to IDR( $s$ ) corresponds to replacing the single vector  $\mathbf{p} \in \mathbb{C}^n$  with a matrix or block vector  $\mathbf{P} \in \mathbb{C}^{n \times s}$ ,  $1 \leq s \leq n$ .

To analyze IDR and IDR( $s$ ), we have to consider generalized Hessenberg decompositions (also referred to as rational Hessenberg decompositions) and to generalize QOR, QMR and Ritz-Galérkin. We have to prove that the expressions for the iterates and residuals based on polynomials are still valid. But: All these approaches extend easily to generalized Hessenberg decompositions. The prototype IDR( $s$ ) algorithm without the recurrences for  $\mathbf{x}_m$  and thus already slightly rewritten is given in [Algorithm 4](#). This variant forms the basis for eigenvalue computations based on IDR in [\[5\]](#).

```

input :  $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, \mathbf{p}$ 
output:  $\mathbf{f}_k, \mathbf{x}_k, \gamma_k, \omega_k$ 
1  $\gamma_0 = 0$ 
2  $\mathbf{f}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$ 
3  $\Delta\mathbf{g}_0 = \mathbf{o}_n$ 
4  $\Delta\mathbf{y}_0 = \mathbf{o}_n$ 
5 for  $k = 1, \dots$  do
6    $\mathbf{s}_k = \mathbf{f}_{k-1} + \gamma_{k-1}\Delta\mathbf{g}_{k-1}$ 
7    $\mathbf{t}_k = \mathbf{A}\mathbf{s}_k$ 
8   if  $k = 1$  or  $k$  is even then
9      $\omega_k = (\mathbf{t}_k^H \mathbf{s}_k) / (\mathbf{t}_k^H \mathbf{t}_k)$ 
10  else
11     $\omega_k = \omega_{k-1}$ 
12  end
13   $\Delta\mathbf{x}_k = \gamma_{k-1}\Delta\mathbf{y}_{k-1} - \omega_k \mathbf{s}_k$ 
14   $\Delta\mathbf{f}_k = \gamma_{k-1}\Delta\mathbf{g}_{k-1} - \omega_k \mathbf{t}_k$ 
15   $\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta\mathbf{x}_k$ 
16   $\mathbf{f}_k = \mathbf{f}_{k-1} + \Delta\mathbf{f}_k$ 
17  if  $k$  is even then
18     $\Delta\mathbf{y}_k = \Delta\mathbf{y}_{k-1}$ 
19     $\Delta\mathbf{g}_k = \Delta\mathbf{g}_{k-1}$ 
20  else
21     $\Delta\mathbf{y}_k = \Delta\mathbf{x}_k$ 
22     $\Delta\mathbf{g}_k = \Delta\mathbf{f}_k$ 
23  end
24   $\gamma_k = -(\mathbf{p}^H \mathbf{f}_k) / (\mathbf{p}^H \Delta\mathbf{g}_k)$ 
25 end

```

**Algorithm 3:** Classical IDR

With mention a few remarks: We can start with any (simple) Krylov subspace method. The steps in the  $s$ -loop only differ from the first block in that no new  $\omega_j$  is computed. IDR( $s$ )ORES is based on oblique projections and  $s + 1$  consecutive multiplications with the same linear factor  $\mathbf{I} - \omega_j \mathbf{A}$ . To understand IDR we have to find the underlying Hessenberg decomposition. We already noted that essential features of Krylov subspace methods can be described by a Hessenberg decomposition

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_{m+1}\underline{\mathbf{H}}_m = \mathbf{Q}_m\mathbf{H}_m + \mathbf{q}_{m+1}h_{m+1,m}\mathbf{e}_m^T. \quad (1.8)$$

Here,  $\mathbf{H}_m$  denotes an unreduced Hessenberg matrix. In the perturbed case, e.g., in finite precision and/or based on inexact matrix-vector multiplies, we obtain a perturbed Hessenberg decomposition

$$\mathbf{A}\mathbf{Q}_m + \mathbf{F}_m = \mathbf{Q}_{m+1}\underline{\mathbf{H}}_m = \mathbf{Q}_m\mathbf{H}_m + \mathbf{q}_{m+1}h_{m+1,m}\mathbf{e}_m^T. \quad (1.9)$$

The matrix  $\mathbf{H}_m$  of the perturbed variant will, in general, still be unreduced. It turns out that in case of IDR we have to consider *generalized* Hessenberg decompositions.

$$\mathbf{A}\mathbf{Q}_m\mathbf{U}_m = \mathbf{Q}_{m+1}\underline{\mathbf{H}}_m = \mathbf{Q}_m\mathbf{H}_m + \mathbf{q}_{m+1}h_{m+1,m}\mathbf{e}_m^T \quad (1.10)$$

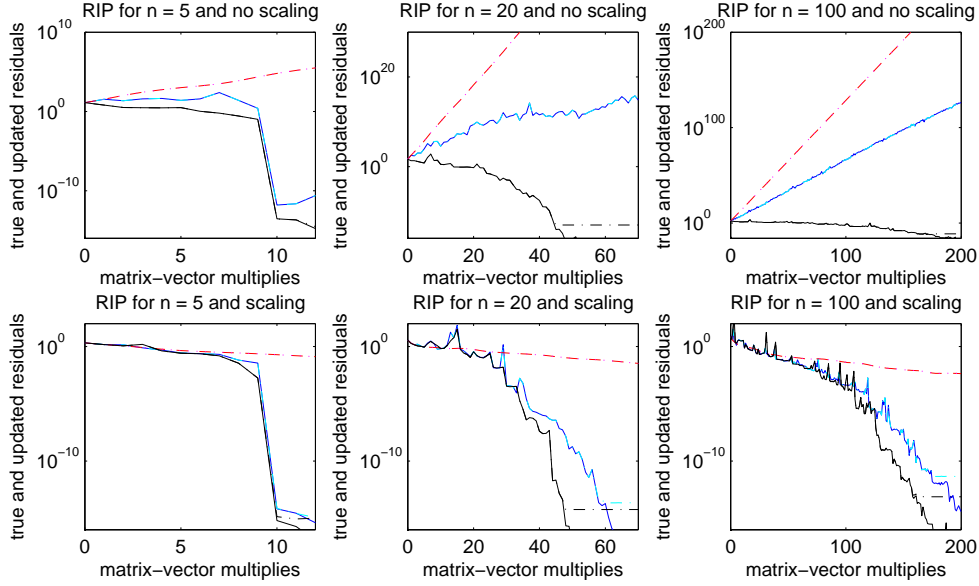


FIGURE 1.2. A numerical comparison of Richardson's iteration, classical IDR, and PIA. Classical IDR performs best for these small-scale test problems and converges in all cases to a reasonable level of accuracy, even when PIA and Richardson's iteration diverge.

and *perturbed* generalized Hessenberg decompositions

$$\mathbf{A}\mathbf{Q}_m\mathbf{U}_m + \mathbf{F}_m = \mathbf{Q}_{m+1}\mathbf{H}_m = \mathbf{Q}_m\mathbf{H}_m + \mathbf{q}_{m+1}h_{m+1,m}\mathbf{e}_m^\top \quad (1.11)$$

with upper triangular (possibly even singular) matrices  $\mathbf{U}_m$ . Generalized Hessenberg decompositions correspond to an oblique projection of the pencil  $(\mathbf{A}, \mathbf{I})$  to the pencil  $(\mathbf{H}_m, \mathbf{U}_m)$  as long as  $\mathbf{Q}_{m+1}$  has full rank,

$$\begin{aligned} \widehat{\mathbf{Q}}_m^H(\mathbf{A}, \mathbf{I})\mathbf{Q}_m\mathbf{U}_m &= \widehat{\mathbf{Q}}_m^H(\mathbf{A}\mathbf{Q}_m\mathbf{U}_m, \mathbf{Q}_m\mathbf{U}_m) \\ &= \widehat{\mathbf{Q}}_m^H(\mathbf{Q}_{m+1}\mathbf{H}_m, \mathbf{Q}_m\mathbf{U}_m) = (\mathbf{I}_m^\top\mathbf{H}_m, \mathbf{U}_m) = (\mathbf{H}_m, \mathbf{U}_m), \end{aligned} \quad (1.12)$$

here  $\widehat{\mathbf{Q}}_m^H := \mathbf{I}_m^\top\mathbf{Q}_{m+1}^\dagger$ .

In understanding IDR, we have to understand the concept of ORTHORES-type methods. The entries of the Hessenberg matrices of Hessenberg decompositions are defined in different variations. Three well-known ways for implementing the QOR/QMR approach are commonly denoted as ORTHORES, ORTHOMIN, and ORTHODIR. The prototype IDR( $s$ ) belongs to the class ORTHORES and uses short recurrences, therefore we refer to it as IDR( $s$ )ORES. ORTHORES-type methods have a generalized Hessenberg decomposition

$$\mathbf{A}\mathbf{R}_m\mathbf{U}_m = \mathbf{R}_{m+1}\mathbf{H}_m^\circ = \mathbf{R}_m\mathbf{H}_m^\circ + \mathbf{r}_{m+1}h_{m+1,m}^\circ\mathbf{e}_m^\top, \quad (1.13)$$

where  $\mathbf{e}_m^\top\mathbf{H}_m^\circ = \mathbf{o}_m^\top$ ,  $\mathbf{e}^\top = (1, \dots, 1)$ , and the matrix

$$\mathbf{R}_{m+1} = (\mathbf{r}_0, \dots, \mathbf{r}_m) = \mathbf{Q}_{m+1} \operatorname{diag} \left( \frac{\|\mathbf{r}_0\|_2}{\|\mathbf{q}_1\|_2}, \dots, \frac{\|\mathbf{r}_m\|_2}{\|\mathbf{q}_{m+1}\|_2} \right) \quad (1.14)$$

```

input :  $\mathbf{A}, \mathbf{b}, \mathbf{x}_0, s, \mathbf{P}$ 
output:  $\mathbf{R}_{m+1}, \mathbf{c}_{s+1}, \mathbf{c}_{s+2}, \dots, \omega_1, \omega_2, \dots$ 
1  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ 
2 compute  $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \dots, \mathbf{r}_s)$  using, e.g., ORTHORES
3  $\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \dots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \dots, \mathbf{r}_s - \mathbf{r}_{s-1})$ 
4  $m \leftarrow s + 1, j \leftarrow 1$ 
5 while not converged do
6    $\mathbf{c}_m = (\mathbf{P}^H \nabla\mathbf{R}_{m-s:m-1})^{-1} \mathbf{P}^H \mathbf{r}_{m-1}$ 
7    $\mathbf{v}_{m-1} = \mathbf{r}_{m-1} - \nabla\mathbf{R}_{m-s:m-1} \mathbf{c}_m$ 
8   compute  $\omega_j$ 
9    $\nabla\mathbf{r}_m = -\nabla\mathbf{R}_{m-s:m-1} \mathbf{c}_m - \omega_j \mathbf{A} \mathbf{v}_{m-1}$ 
10   $\mathbf{r}_m = \mathbf{r}_{m-1} + \nabla\mathbf{r}_m$ 
11   $m \leftarrow m + 1$ 
12   $\nabla\mathbf{R}_{m-s:m-1} = (\nabla\mathbf{r}_{m-s}, \dots, \nabla\mathbf{r}_{m-1})$ 
13  for  $k = 1, \dots, s$  do
14     $\mathbf{c}_m = (\mathbf{P}^H \nabla\mathbf{R}_{m-s:m-1})^{-1} \mathbf{P}^H \mathbf{r}_{m-1}$ 
15     $\mathbf{v}_{m-1} = \mathbf{r}_{m-1} - \nabla\mathbf{R}_{m-s:m-1} \mathbf{c}_m$ 
16     $\nabla\mathbf{r}_m = -\nabla\mathbf{R}_{m-s:m-1} \mathbf{c}_m - \omega_j \mathbf{A} \mathbf{v}_{m-1}$ 
17     $\mathbf{r}_m = \mathbf{r}_{m-1} + \nabla\mathbf{r}_m$ 
18     $m \leftarrow m + 1$ 
19     $\nabla\mathbf{R}_{m-s:m-1} = (\nabla\mathbf{r}_{m-s}, \dots, \nabla\mathbf{r}_{m-1})$ 
20  end
21   $j \leftarrow j + 1$ 
22 end

```

**Algorithm 4:** IDR( $s$ )ORES

is diagonally scaled to be the matrix of residual vectors.

Next we depict the underlying generalized Hessenberg decomposition. We already noted in the remarks following Algorithm 4 that the IDR recurrences of IDR( $s$ )ORES can be summarized by

$$\begin{aligned}
\mathbf{v}_{m-1} &:= \mathbf{r}_{m-1} - \nabla\mathbf{R}_{m-s:m-1} \mathbf{c}_m = \mathbf{R}_{m-s-1:m-1} \mathbf{y}_m \\
&= (1 - \gamma_s^{(m)}) \mathbf{r}_{m-1} + \sum_{\ell=1}^{s-1} (\gamma_{s-\ell+1}^{(m)} - \gamma_{s-\ell}^{(m)}) \mathbf{r}_{m-\ell-1} + \gamma_1^{(m)} \mathbf{r}_{m-s-1}, \\
\mathbf{1} \cdot \mathbf{r}_m &:= (\mathbf{I} - \omega_j \mathbf{A}) \mathbf{v}_{m-1}.
\end{aligned}$$

Here,  $m > s$ , and the index of the scalar  $\omega_j$  is defined by

$$j := \left\lfloor \frac{m}{s+1} \right\rfloor,$$

compare with the so-called “index functions” in [25]. Removing  $\mathbf{v}_{m-1}$  from the recurrence we obtain the generalized Hessenberg decomposition

$$\mathbf{A} \mathbf{R}_m \mathbf{Y}_m \mathbf{D}_\omega^{(m)} = \mathbf{R}_{m+1} \mathbf{Y}_m^\circ, \quad (1.15)$$

where non-zero elements in the upper triangular part of the columns of the upper triangular matrix  $\mathbf{Y}_m$  and the unreduced extended upper Hessenberg matrix  $\mathbf{Y}_m^\circ$

are given by the elements in the vectors  $\mathbf{y}_m$ , the diagonal matrix  $\mathbf{D}_\omega^{(m)}$  has  $s + 1$  repeated copies of  $\omega_j$  along the diagonal and the lower diagonal of the unreduced extended upper Hessenberg matrix  $\underline{\mathbf{Y}}_m^\circ$  is given by minus ones, which makes it an ORTHORES-type matrix, i.e., the columns sum to zero.

**2. IDR( $s$ )Eig.** The IDR( $s$ )EIG approach in [5] is based on the upper unreduced Hessenberg/upper triangular pencil of the generalized Hessenberg decomposition (1.15). For details we refer the reader to the forthcoming technical report [5]. We simply sketch the mathematical transformations with the aid of small pictures based on IDR( $s$ )ORES with  $s = 3$  for 12 steps.

**2.1. Sonneveld pencil.** In this small section we consider the Sonneveld pencil and Sonneveld matrix. The IDR( $s$ )ORES pencil, the so-called Sonneveld pencil  $(\mathbf{Y}_m^\circ, \mathbf{Y}_m \mathbf{D}_\omega^{(m)})$ , can be depicted by

$$\begin{pmatrix} \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ \end{pmatrix}.$$

We observe a nice banded structure of this pencil. The eigenvalues can easily be computed using the QZ algorithm. The data dependencies can be reduced to obtain the approximate eigenvalues in a more stable manner. The upper triangular matrix  $\mathbf{Y}_m \mathbf{D}_\omega^{(m)}$  could be inverted, which results in the Sonneveld matrix, a full unreduced Hessenberg matrix. The QR algorithm on the Sonneveld matrix could be used to obtain the same approximations to eigenvalues.

**2.2. Purified pencil.** We know the eigenvalues  $\approx$  roots of kernel polynomials  $1/\omega_j$ . We are only interested in the other eigenvalues. The purified IDR( $s$ )ORES pencil  $(\mathbf{Y}_m^\circ, \mathbf{U}_m \mathbf{D}_\omega^{(m)})$ , that has only the remaining eigenvalues and some infinite ones as eigenvalues, can be depicted by

$$\begin{pmatrix} \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ \end{pmatrix}.$$

We get rid of the infinite eigenvalues using a change of basis (block Gauß elimination/Schur complement). The resulting deflated purified IDR( $s$ )ORES pencil, after the elimination step  $(\mathbf{Y}_m^\circ \mathbf{G}_m, \mathbf{U}_m \mathbf{D}_\omega^{(m)})$ , where  $\mathbf{G}_m$  denotes the block Gauß-

eliminator, can be depicted by

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ + & \times & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\ \circ & \circ & + & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & + & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \times & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \end{pmatrix}.$$

Using Laplace expansion of the determinant of  $z\mathbf{U}_m\mathbf{D}_\omega^{(m)} - \mathbf{Y}_m^\circ\mathbf{G}_m$  we can get rid of the trivial constant factors corresponding to infinite eigenvalues. This amounts to a deflation.

**2.3. Deflated pencil.** Let  $D$  denote an deflation operator that removes every  $s + 1$ th column and row from the matrix the operator is applied to. The deflated purified IDR( $s$ )ORES pencil, after the deflation step ( $D(\mathbf{Y}_m^\circ\mathbf{G}_m), D(\mathbf{U}_m\mathbf{D}_\omega^{(m)})$ ), can be depicted by

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times & \circ & \circ & \circ \\ + & \times & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & + & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & \circ & + & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & + & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & + & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \times & \times & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times \end{pmatrix}.$$

The block-diagonal matrix  $D(\mathbf{U}_m\mathbf{D}_\omega^{(m)})$  has invertible upper triangular blocks and can be inverted to expose the underlying Lanczos process, a Lanczos process with  $s$  left-hand and one right-hand side. We refer to this process as BiORES( $s,1$ ), as this is a process of ORTHORES-type. In the next section we show that IDR( $s$ ) is a Lanczos process with multiple left-hand sides.

**2.4. BiORES( $s,1$ ).** Inverting the block-diagonal matrix  $D(\mathbf{U}_m\mathbf{D}_\omega^{(m)})$  gives an algebraic eigenvalue problem with a block-tridiagonal unreduced upper Hessenberg matrix

$$\mathbf{L}_m := D(\mathbf{Y}_m^\circ\mathbf{G}_m) \cdot D(\mathbf{U}_m\mathbf{D}_\omega^{(m)})^{-1} = \begin{pmatrix} \times & \times & \times & \times & \times & \times & \circ & \circ & \circ \\ + & \times & \times & \times & \times & \times & \circ & \circ & \circ \\ \circ & + & \times & \times & \times & \times & \times & \times & \times \\ \circ & \circ & + & \times & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & + & \times & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & + & \times & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times \end{pmatrix}.$$

This is the matrix of the underlying BiORES( $s,1$ ) process. The extended matrix version  $\underline{\mathbf{L}}_m$  satisfies

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_{m+1}\underline{\mathbf{L}}_m,$$

where the reduced residuals  $\mathbf{q}_{js+k}$ ,  $k = 0, \dots, s-1$ ,  $j = 0, 1, \dots$ , with  $\Omega_0(z) \equiv 1$  and  $\Omega_j(z) = \prod_{k=1}^j (1 - \omega_k z)$  are given by  $\Omega_j(\mathbf{A})\mathbf{q}_{js+k} = \mathbf{r}_{j(s+1)+k}$ . The reduced residuals

are defined by

$$\Omega_j(\mathbf{A})\mathbf{q}_{js+k} = \mathbf{r}_{j(s+1)+k} = (\mathbf{I} - \omega_j\mathbf{A})\mathbf{v}_{j(s+1)+k-1}$$

and every  $\mathbf{v}_{j(s+1)+k-1}$  is orthogonal to  $\mathbf{P}$ . Thus,  $\mathbf{q}_{js+k} \perp \Omega_{j-1}(\mathbf{A}^H)\mathbf{P}$ . Using induction [13] one can prove that  $\mathbf{q}_{js+k} \perp \mathcal{K}_j(\mathbf{A}^H, \mathbf{P})$ ; thus, this is a two-sided Lanczos process with  $s$  left and one right starting vectors. This can more easily be proven using the representations ( $\mathcal{S} := \mathbf{P}^\perp$ )

$$\begin{aligned} \mathcal{G}_0 &= \mathcal{K}(\mathbf{A}, \mathbf{r}_0), \quad \text{where } \mathcal{K}(\mathbf{A}, \mathbf{r}_0) \text{ denotes the full Krylov subspace,} \\ \mathcal{G}_j &= \bigcap_{k=0}^{j-1} \Omega_k(\mathbf{A})^{-1} \Omega_j(\mathbf{A})(\mathcal{S}) = \left( \bigoplus_{k=0}^{j-1} \Omega_j(\mathbf{A})^{-H} \Omega_k(\mathbf{A})^H \{\mathbf{P}\} \right)^\perp \\ &= \left( \Omega_j(\mathbf{A})^{-H} \mathcal{K}_j(\mathbf{A}^H, \mathbf{P}) \right)^\perp = \Omega_j(\mathbf{A}) \left( \mathcal{K}_j(\mathbf{A}^H, \mathbf{P}) \right)^\perp \end{aligned}$$

of the Sonneveld spaces. This has to be compared with Theorem 4.2 in [13] and with Theorem 4.1 in [11] (here a similar result is obtained; the authors use a slightly different method of proof).

The first equality

$$\mathcal{G}_j = \bigcap_{k=0}^{j-1} \Omega_k(\mathbf{A})^{-1} \Omega_j(\mathbf{A})(\mathcal{S}) = \bigcap_{k=1}^j (\mathbf{I} - \omega_j\mathbf{A}) \cdots (\mathbf{I} - \omega_k\mathbf{A})(\mathcal{S})$$

follows from the observations that the first  $s+1$  residuals obviously are in  $\mathcal{G}_0 := \mathcal{K}(\mathbf{A}, \mathbf{r}_0)$ , the next  $s+1$  residuals (or any other vectors in  $\mathcal{G}_1$ ) are in the  $\mathbf{I} - \omega_1\mathbf{A}$  image of  $\mathcal{S} = \mathbf{P}^\perp$ , the last  $s+1$  residuals are in the  $\mathbf{I} - \omega_j\mathbf{A}$  image of  $\mathcal{S} = \mathbf{P}^\perp$ , and, since they are computed as images of linear combinations of previous information, also images of linear combinations of previously obtained images  $(\mathbf{I} - \omega_{j-1}\mathbf{A}) \cdots (\mathbf{I} - \omega_k\mathbf{A})$  of  $\mathcal{S} = \mathbf{P}^\perp$ .

The second equality

$$\bigcap_{k=0}^{j-1} \Omega_k(\mathbf{A})^{-1} \Omega_j(\mathbf{A})(\mathcal{S}) = \left( \bigoplus_{k=0}^{j-1} \Omega_j(\mathbf{A})^{-H} \Omega_k(\mathbf{A})^H \{\mathbf{P}\} \right)^\perp$$

is based on

$$\mathbf{B}\mathbf{P}^\perp = (\mathbf{B}^{-H}\mathbf{P})^\perp$$

and

$$\mathcal{U}^\perp \cap \mathcal{V}^\perp = (\mathcal{U} \cup \mathcal{V})^\perp = (\mathcal{U} + \mathcal{V})^\perp.$$

The second relations are basic linear algebra. The first relation follows from the observation that

$$\mathbf{P}^\perp = \{ \mathbf{v} \in \mathbb{C}^n \mid \mathbf{P}^H \mathbf{v} = \mathbf{o}_s \} \quad \Rightarrow \quad \mathbf{B}\mathbf{P}^\perp = \{ \mathbf{B}\mathbf{v} \in \mathbb{C}^n \mid \mathbf{P}^H \mathbf{v} = \mathbf{o}_s \},$$

since, for invertible  $\mathbf{B}$ ,

$$\mathbf{y} \in \mathbf{B}\mathbf{P}^\perp \Leftrightarrow \{ \mathbf{y} = \mathbf{B}\mathbf{v} \wedge \mathbf{P}^H \mathbf{v} = \mathbf{o}_s \} \Leftrightarrow \mathbf{P}^H \mathbf{v} = \mathbf{P}^H \mathbf{B}^{-1} \mathbf{y} = (\mathbf{B}^{-H} \mathbf{P})^H \mathbf{y} = \mathbf{o}_s.$$

The third and fourth equality

$$\begin{aligned} \left( \sum_{k=0}^{j-1} \Omega_j(\mathbf{A})^{-\mathbf{H}} \Omega_k(\mathbf{A})^{\mathbf{H}} \{\mathbf{P}\} \right)^\perp &= \left( \Omega_j(\mathbf{A})^{-\mathbf{H}} \mathcal{K}_j(\mathbf{A}^{\mathbf{H}}, \mathbf{P}) \right)^\perp \\ &= \Omega_j(\mathbf{A}) \left( \mathcal{K}_j(\mathbf{A}^{\mathbf{H}}, \mathbf{P}) \right)^\perp \end{aligned}$$

are satisfied

- since the polynomials  $\Omega_k(\mathbf{A})$ ,  $0 \leq k < j$  form a basis of the space of polynomials of degree less  $j$ , and
- by the property proved above, respectively.

**2.5. Generalizations of IDR( $s$ ).** The residuals computed last in a complete cycle are uniquely defined. Based on the analysis of a possible breakdown of IDR( $s$ ), Sonneveld and van Gijzen came up with their new implementation IDR( $s$ )BIO [23]. In this implementation of the IDR Theorem they use basis vectors  $\mathbf{g}_{-1}, \dots, \mathbf{g}_{-s} \in \mathcal{G}_j$ , which are not simply residual differences, but linear combinations of these.

The new vectors  $\mathbf{v}_m$  and  $\mathbf{r}_{m+1}$  are in this general setting given by the updates

$$\begin{aligned} \mathbf{v}_m &= \mathbf{r}_m - \sum_{i=1}^s \mathbf{g}_{m-i} \gamma_i =: \mathbf{r}_m - \mathbf{G}_m \mathbf{c}_m, \quad \text{and thus,} \\ \mathbf{r}_{m+1} &= (\mathbf{I} - \omega \mathbf{A}) \mathbf{v}_m = \mathbf{r}_m - \omega \mathbf{A} \mathbf{v}_m - \sum_{i=1}^s \mathbf{g}_{m-i} \gamma_i, \end{aligned}$$

where  $\mathbf{c}_m$  is determined such that  $\mathbf{P}^{\mathbf{H}} \mathbf{v}_m = \mathbf{o}$ .

Recently, the relations between IDR( $s$ ) and BICGSTAB( $\ell$ ) and combinations of both methods have been investigated.

- In [13] the authors derive different implementations of ML( $k$ )BICGSTAB-like algorithms.
- In [14] the authors combine the IDR philosophy with higher degree stabilization polynomials. The resulting method is named IDR( $s$ )STAB( $\ell$ ). The approach is comparable to the one resulting in BICGSTAB( $\ell$ ).
- In [20] the authors derive the algorithm GBICGSTAB( $s, L$ ), which is similar to IDR( $s$ )STAB( $\ell$ ). In their own words: “Our algorithm is to theirs what the Gauss-Seidel iteration is to the Jacobi iteration.” A predecessor of GBICGSTAB( $s, L$ ) seems to be the method called GIDR( $s, L$ ) in [19].
- In [12] the ideas behind BICGSTAB2 [4] and GPBICG [29] are considered.

The relation of IDR( $s$ ) to Petrov-Galérkin with a rational Krylov space motivated the method IDR-Ritz [11]. Another, simpler motivation is that the residual polynomials should be designed to dampen the spectrum. Using the residual polynomial representation of IDR( $s$ ) we could choose the  $1/\omega_j$  close but not equal to eigenvalues, at least we should choose them in the field of values of  $\mathbf{A}$ .

The minimization used in IDR( $s$ )ORES and IDR( $s$ )BIO results in values  $\omega_j$  which are in the field of values of  $\mathbf{A}^{-\mathbf{H}}$ , thus Simoncini and Szyld suggest to use a few steps of the Arnoldi method to compute some Ritz values, which are then used in some ordering as  $1/\omega_j$  values. For real non-symmetric matrices this typically results in an algorithm based on complex arithmetic in place of real arithmetic.

Last but not least: Certain old ideas have been reactivated. Sonneveld presented the hitherto unpublished Accelerated Gauß-Seidel (AGS) method at the Kyoto Forum on Krylov Subspace Methods in 2008. Based on the algorithm in the proceedings, Seiji Fujino et al. considered the acceleration of the classical splitting methods (Jacobi, Gauß-Seidel and SOR). The resulting methods are called

- IDR( $s$ )-Jacobi (w/o adaptive tuning),
- IDR( $s$ )-GS,
- IDR( $s$ )-SOR.

These approaches result in a “tight packing” of preconditioning and Krylov subspace methods, compare with PIA. In most of these methods the  $\omega_j$  are fixed by the splitting chosen.

**3. Parallelization of IDR( $s$ ) and IDR( $s$ )Eig.** To understand aspects of a possible parallelization, we take a look at the structure of IDR( $s$ ). IDR( $s$ ) is a typical Krylov subspace method. Most known Krylov subspace methods are based on

- matrix-vector products (“black-box” or with a sparse matrix),
- some kind of (bi-)orthogonalization (frequently Gram-Schmidt),
- solution of small linear systems,
- `_dots` (line search minimization),
- `_axpys` or `_gemvs` (updates of vectors).

**3.1. ... a short introduction to IDR( $s$ ) parallelization.** Here, we simply sketch the IDR( $s$ ) variant IDR( $s$ )BIO described in the technical report [23]. Its Matlab source code can be downloaded from

<http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html>.

We remark that this version of IDR( $s$ ) is contained in release 3.0 of the IFISS package. The structure of an unpreconditioned IDR( $s$ )BIO is sketched as a pseudo-code in Algorithm 5. A parallelization is easily possible, we could use, e.g.: {Sca,P}LAPACK, CUBLAS/CUDA, cloud computing, MPI & OpenMP, ...

A naïve CUBLAS implementation for instance would be based on the parallel evaluation of small sized problems and several calls to `cublasSaxpy()`, `cublasSgemv()`, `cublasSdot()`, and `cublasScopy()`. But there is room for improvement. The implementation of the matrix-vector multiplication with  $\mathbf{A}$  should be adjusted to the type of matrix given, e.g.,

- (fully optimized parallel) “black-box”,
- reordering of a given sparse matrix using some heuristics (e.g., reverse Cuthill-McKee, multi-color schemes, ...) for a better block-distribution to the nodes and to reduce communication,
- parallel implementation of an  $\mathcal{H}$ -matrix format.

We could also allow for a (parallel) preconditioner.

Most important is the load balancing, especially in a non-homogeneous environment, since we have several synchronization points in the algorithm, namely the `_dots` from orthogonalization and (possibly) the computation of  $\omega_j$  and (possibly) the computation of the norm of the residual or backward error.

The naïve CUBLAS implementation can be enhanced by using some other IDR( $s$ ) variant. The triangular bi-orthogonalisation scheme could be replaced:

- Instead of modified Gram-Schmidt we could use (iterated) classical Gram-Schmidt.
- Furthermore, we could adopt delayed re-orthogonalization [6] to the case of iterated Gram-Schmidt-like bi-orthogonalisation.
- We could use other triangular basis transformations. This would result in full  $s \times s$ -systems, but the main computational effort takes place elsewhere.

Similarly to the approaches used in parallel implementations of CG [10, 3] we could try to minimize the number of synchronization barriers given by the orthogonalisation against  $\mathbf{P}$ , any other occurring (bi-)orthogonalization and the computation of  $\omega$  by utilization of algebraic rewritings. The recent parallel variant in [2] is based on a minimization of synchronization barriers.

Next we give a few remarks about the general structure of  $\text{IDR}(s)$ . We can quite easily get rid of the synchronization points caused by the computation of the non-zero scalars  $\omega_j$ . One idea is to precompute a certain amount of Ritz values like in [11] or to use the CPU (or some nodes of the GPU) to compute rough approximations to eigenvalues based on the Sonneveld pencil [5].

If we use a method like  $\text{IDR}(s)$ -Jacobi, we typically have a lower convergence rate, but have removed all synchronization points due to (bi-)orthogonalization of the basis vectors in the Sonneveld spaces or their pre-images. This may give a speedup that covers the price paid due to a slower convergence.

But: We can never get rid of the orthogonalization against  $\mathbf{P}$ . This has to be carried out in *every*  $\text{IDR}(s)$  method. This part of the algorithm should be optimized using code adopted to the architecture we are working on.

We give a few remarks on the structure of  $\mathbf{P}$  and the resulting cost of orthogonalization. We could adopt the generic choice for  $\mathbf{P}$ , namely, using randomly generated orthonormal columns to the type of problem. If the problem is the discrete version of a 2D or 3D physical problem, we could use as test vectors  $\mathbf{p}_j$  the discretization of test functions with compact support, e.g., we could use some (non-)overlapping Schwarz-like vectors. These vectors can be stored more compactly and could be distributed to all nodes in a distributed memory architecture. To save memory, we could use randomly generated vectors with only  $\pm 1$  and 0. The orthogonalization against  $\mathbf{P}$  is in this case based on the computation of two sums of subsets of the vectors and finally one subtraction. Additional structure gives additional gain in efficiency. Both ideas can be combined. One has to carefully balance the benefits and the risks of resulting instabilities.

Last we give a few remarks about the structure of  $\text{IDR}(s)\text{EIG}$  and its possible parallelization. As  $\text{IDR}(s)\text{EIG}$  is based on (a given variant) of  $\text{IDR}(s)$ , the same comments apply. We only have to store the vectors defining the orthogonalization against  $\mathbf{P}$  (in every step one vector of length  $s$ ), any triangular basis transformations (in every sweep of  $s + 1$  steps a few  $s \times s$  triangular matrices) and the  $\omega_j$  used for  $s + 1$  consecutive steps. The computation of the eigenvalues should be performed on some of the pencils using an adopted QZ algorithm working near the original band of the banded pencil, the shift strategy should be chosen as to minimize communication between diagonal blocks while retaining favorable convergence properties. It is not known by now, which  $\text{IDR}(s)\text{Eig}$  algorithm is the one most stable. Thus, up to now, nobody did consider to come up with a stable eigenvalue solver designed for the special structure of pencils stemming from  $\text{IDR}(s)$  algorithms. Once a good candidate for an

IDR( $s$ ) algorithm suitable for stable eigenvalue computations is known, one can come up with a parallel variant.

**4. Conclusions.** We gave a short introduction to Krylov subspace methods with special emphasis on IDR. We sketched the IDR and IDR( $s$ ) families and indicated how to compute eigenvalues using one particular instance of IDR( $s$ ), namely, IDR( $s$ )ORES. We sketched the relation between the algorithm IDR( $s$ )ORES for the approximate solution of linear systems, the eigenvalue routine IDR( $s$ )EIG based on it, and a two-sided Lanczos process BIORRES( $s,1$ ). We briefly indicated how to parallelize one member of the IDR( $s$ ) family, namely the more recent variant IDR( $s$ )BiO. We gave only a rather philosophical treatment of a parallel implementation of IDR( $s$ )EIG. Much work, both theoretical and practical, remains to be done.

**Acknowledgments.** I am indebted to my co-author Martin Gutknecht. It was (and still is) a pleasure to write the joint report [5]. Without Peter Sonneveld answering my e-mail dating to 2006 and sending me an e-mail informing me about the DCSE symposium on IDR, June 3, 2009, nothing of this would have happened. I thank Martin van Gijzen for our interesting encounter in Harrachov 2007 and many interesting discussions since then. I have to thank our whole IDR family. I am deeply grateful for the support by Prof. Seiji Fujino from the Research Institute for Information Technology of Kyushu University who enabled my visit to Japan and thus my talks at Kyushu University and at the University of Tokyo. Last but not least I would like to thank Prof. Kengo Nakajima and Prof. Takahiro Katagiri at the Supercomputing Division of the Information Technology Center of the University of Tokyo for making my talk and this extended abstract possible.

**Postscript.** Just after the slides for the talk in Tokyo had been finished (5th February 2010), Tijmen Collignon and Martin van Gijzen published a technical report [2] on parallelization of IDR( $s$ ) (9th February 2010). In this report some of the ideas presented in the talk have been considered, namely, the use of specially structured  $\mathbf{P}$  and some algebraic rewritings to minimize the number of synchronization points.

#### REFERENCES

- [1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.
- [2] Tijmen P. Collignon and Martin B. van Gijzen. Fast solution of nonsymmetric linear systems on Grid computers using parallel variants of IDR( $s$ ). Reports of the Department of Applied Mathematical Analysis Report 10-05, Delft University of Technology, 2010.
- [3] Eduardo F. D’Azevedo, Victor Eijkhout, and Charles H. Romine. A matrix framework for conjugate gradient methods and some variants of CG with less synchronization overhead. In *PPSC*, pages 644–646, 1993.
- [4] Martin H. Gutknecht. Variants of BICGSTAB for matrices with complex spectrum. *SIAM J. Sci. Comput.*, 14(5):1020–1033, September 1993.
- [5] Martin H. Gutknecht and Jens-Peter M. Zemke. Eigenvalue computations based on IDR, 2010. Technical Report (to appear 2010).
- [6] V. Hernández, J. E. Román, and A Tomás. A parallel variant of the Gram-Schmidt process with reorthogonalization. In G. R. Joubert, W. E. Nagel, F. J. Peters, O. G. Plata, P. Tirado, and E. L. Zapata, editors, *Proceedings of the International Conference on Parallel Computing (ParCo 2005)*, volume 33, pages 221–228. Central Institute for Applied Mathematics, Jülich, Germany, 2006.
- [7] Karl Hessenberg. Behandlung linearer Eigenwertaufgaben mit Hilfe der Hamilton-Cayleyschen Gleichung. Numerische Verfahren, Bericht 1, Institut für Praktische Mathematik (IPM),

- Technische Hochschule Darmstadt, July 1940. Scanned report and biographical sketch of Karl Hessenberg's life online available at <http://www.hessenberg.de/karl1.html>.
- [8] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bureau Standards*, 45:255–281, 1950.
  - [9] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bureau Standards*, 49:33–53, 1952.
  - [10] Gérard Meurant. Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Comput.*, 5(3):267–280, 1987.
  - [11] Valeria Simoncini and Daniel Szyld. Interpreting IDR as a Petrov-Galerkin method. Report 09-10-22, Dipartimento di Matematica, Università di Bologna and Department of Mathematics, Temple University, Philadelphia, 2009.
  - [12] Gerard L. G. Sleijpen and Kuniyoshi Abe, 2010. Publication in preparation (January 2010).
  - [13] Gerard L. G. Sleijpen, Peter Sonneveld, and Martin B. van Gijzen. Bi-CGSTAB as an induced dimension reduction method. Reports of the Department of Applied Mathematical Analysis Report 08-07, Delft University of Technology, 2008. ISSN 1389-6520.
  - [14] Gerard L. G. Sleijpen and Martin B. van Gijzen. Exploiting BiCGstab( $\ell$ ) strategies to induce dimension reduction. Reports of the Department of Applied Mathematical Analysis Report 09-02, Delft University of Technology, 2009. ISSN 1389-6520.
  - [15] P. Sonneveld and M. B. van Gijzen. IDR( $s$ ): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. Report 07-07, Department of Applied Mathematical Analysis, Delft University of Technology, 2007.
  - [16] Peter Sonneveld. History of IDR: an example of serendipity. PDF file sent by Peter Sonneveld on Monday, 24th of July 2006, July 2006. 8 pages; evolved into [17].
  - [17] Peter Sonneveld. AGS-IDR-CGS-BiCGSTAB-IDR( $s$ ): The circle closed. A case of serendipity. In *Proceedings of the International Kyoto Forum 2008 on Krylov subspace methods*, pages 1–14, September 2008.
  - [18] Peter Sonneveld and Martin B. van Gijzen. IDR( $s$ ): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing*, 31(2):1035–1062, 2008. Received Mar. 20, 2007.
  - [19] Masaaki Tanio and Masaaki Sugihara. GIDR( $s, L$ ): generalized IDR( $s$ ). In *The 2008 annual conference of the Japan Society for Industrial and Applied Mathematic*, pages 411–412, Chiba, Japan, September 2008. (In Japanese).
  - [20] Masaaki Tanio and Masaaki Sugihara. GBi-CGSTAB( $s, L$ ): IDR( $s$ ) with higher-order stabilization polynomials. Technical Report METR 2009-16, Department of Mathematical Informatics, Graduate School of information Science and Technology, University of Tokio, April 2009.
  - [21] 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.*, 13:631–644, 1992. Received May 21, 1990.
  - [22] H. A. van der Vorst and P. Sonneveld. CGSTAB, a more smoothly converging variant of CG-S. Report 90-50, Department of Mathematics and Informatics, Delft University of Technology, 1990.
  - [23] Martin B. van Gijzen and Peter Sonneveld. An elegant IDR( $s$ ) variant that efficiently exploits bi-orthogonality properties. Reports of the Department of Applied Mathematical Analysis Report 08-21, Delft University of Technology, 2008. ISSN 1389-6520.
  - [24] P. Wesseling and P. Sonneveld. Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In *Approximation Methods for Navier-Stokes Problems*, volume 771 of *Lecture Notes in Mathematics*, pages 543–562. Springer, 1980.
  - [25] Man-Chung Yeung and Daniel Boley. Transpose-free multiple Lanczos and its application in Padé approximation. *J. Comput. Appl. Math.*, 177(1):101–127, 2005.
  - [26] Man-Chung Yeung and Tony F. Chan. ML( $k$ )BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors. *SIAM J. Sci. Comput.*, 21(4):1263–1290, 1999. Received May 16, 1997, electr. publ. Dec. 15, 1999.
  - [27] Jens-Peter M. Zemke. Hessenberg eigenvalue-eigenmatrix relations. *Linear Algebra and its Applications*, 414(2–3):589–606, 2006.
  - [28] Jens-Peter M. Zemke. Abstract perturbed Krylov methods. *Linear Algebra and its Applications*, 424(2–3):405–434, 2007.
  - [29] Shao-Liang Zhang. GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 18(2):537–551, 1997.

```

input : A, b, x0, s, P
output: Rm+1, cs+1, cs+2, ..., ω1, ω2, ...
1 x = x0
2 r = b - Ax
3 ω = 1
4 PT = PT
5 G = zeros(n, s)
6 U = zeros(n, s)
7 M = eye(s)
8 while not converged do
9   PTr = PT · r
10  for k = 1, ..., s do
11    % Solve small system and make v orthogonal to P:
12    c = M(k : s, k : s)-1PTr(k : s)
13    v = r - G(:, k : s)c
14    U(:, k) = U(:, k : s)c + ωv
15    % Compute G(:, k) = AU(:, k)
16    G(:, k) = AU(:, k)
17    % Bi-Orthogonalize the new basis vectors:
18    for j = 1, ..., k - 1 do
19      | α = (PT(j, :)G(:, k))/M(j, j)
20      | G(:, k) = G(:, k) - αG(:, j)
21      | U(:, k) = U(:, k) - αU(:, j)
22    end
23    % New column of M = P'*G (first k-1 entries are zero)
24    M(k : s, k) = PT(k : s, :)G(:, k)
25    % Make r orthogonal to pj, j = 1, ..., k
26    β = PTr(k)/M(k, k)
27    r = r - βG(:, k)
28    x = x + βU(:, k)
29    % New PTr = P'*r (first k components are zero)
30    if k < s then
31      | PTr(k + 1 : s) = PTr(k + 1 : s) - βM(k + 1 : s, k)
32    end
33  end
34  % Note: r is already perpendicular to P so v = r
35  v = r
36  t = Av
37  select or compute ω
38  r = r - ωt
39  x = x + ωv
40 end

```

Algorithm 5: IDR(*s*)BiO