IDR(s) and IDR(s)Eig in Parallel Computing

Jens-Peter M. Zemke
zemke@tu-harburg.de

Institut für Numerische Simulation
Technische Universität Hamburg-Harburg

The University of Tokyo, Tokyo
2010/02/12

Outline

IDR and IDR(s)

Krylov subspace methods

1976–1980: IDR

2006–2010: IDR(s)

IDR(s)Eig

Sonneveld pencil

Purified pencil

Deflated pencil

BiORes(s,1)

Generalizations of IDR(s)

Parallelization of IDR(s) and IDR(s)Eig

. . . an introduction to IDR(s) parallelization

Krylov subspace methods

In this talk we consider the IDR methods by Peter Sonneveld (Sonneveld, 2006; Sonneveld, 2008; Wesseling and Sonneveld, 1980) and their generalizations, the IDR(s) methods, starting with the first IDR(s) algorithm (Sonneveld and van Gijzen, 2008).

IDR and IDR(s) are Krylov subspace methods. The mth Krylov subspace $\mathcal{K}_m$ is defined for a given square matrix $\mathbf{A}$ and a starting vector $\mathbf{q}$ as follows,

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

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 $\nu \in \mathcal{P}_{m-1}$ (as long as the Krylov subspace $\mathcal{K}_m$ has full dimension $\dim(\mathcal{K}_m) = m$).

The origin of Krylov subspace methods

The Krylov matrices $\mathbf{K}_m := \left(\mathbf{q}, \mathbf{A}\mathbf{q}, \mathbf{A}^2\mathbf{q}, \ldots, \mathbf{A}^{m-1}\mathbf{q}\right)$ satisfy the matrix recurrence

$$\left(\mathbf{q}, \mathbf{A}\mathbf{K}_m\right) = \mathbf{K}_{m+1}. \tag{1}$$

The mth Krylov matrix spans a basis of the mth Krylov space $\mathcal{K}_m$ iff m is less or equal to the grade of $\mathbf{q}$. We assume here that this is always the case.

Suppose we choose upper triangular basis transformations $\mathbf{K}_m =: \mathbf{Q}_m\mathbf{R}_m$,

$$\left(\mathbf{q}, \mathbf{A}\mathbf{Q}_m\mathbf{R}_m\right) = \mathbf{Q}_{m+1}\mathbf{R}_{m+1} \quad \Rightarrow \quad \left(\mathbf{q}, \mathbf{A}\mathbf{Q}_m\right) = \mathbf{Q}_{m+1}\mathbf{R}_{m+1}\begin{pmatrix} 1 & \mathbf{o}^{\mathsf{T}} \\ \mathbf{o} & \mathbf{R}_m \end{pmatrix}^{-1}. \tag{2}$$

Next we strip off the first column on both sides.

The connection to Hessenberg decompositions

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

$$\begin{pmatrix} \star & \underline{\mathbf{C}}_m \\ \mathbf{o} & \end{pmatrix} := \mathbf{R}_{m+1} \begin{pmatrix} 1 & \mathbf{o}^\mathsf{T} \\ \mathbf{o} & \mathbf{R}_m \end{pmatrix}^{-1} \qquad (3)$$

is unreduced extended Hessenberg.

We end up with a Hessenberg decomposition

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_{m+1}\underline{\mathbf{C}}_m =: \mathbf{Q}_m\mathbf{C}_m + \mathbf{q}_{m+1}\mathbf{c}_{m+1,m}\mathbf{e}_m^\mathsf{T}, \qquad (4)$$

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

Classification of Krylov methods: Matrix based

There are three well-known approaches based on such Hessenberg decompositions (with $\|\mathbf{r}_0\|_2\mathbf{q}_1 = \mathbf{r}_0$), namely,

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

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

Ritz-Galërkin: approximate part of $\mathbf{J} = \mathbf{V}^{-1}\mathbf{A}\mathbf{V}$ by $\mathbf{J}_m := \mathbf{S}_m^{-1}\mathbf{C}_m\mathbf{S}_m$
and part of $\mathbf{V}$ by $\mathbf{V}_m := \mathbf{Q}_m\mathbf{S}_m,$ 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 (Gutknecht and Z., 2010) to compute eigenvalues using IDR.

It turns out to be helpful to look at the corresponding polynomial description: Krylov subspace methods compute elements from the polynomial Krylov subspace $\mathcal{K}_m$.

Classification of Krylov methods: Polynomial based

The three classes of methods can be described using certain polynomials and polynomial interpolation:

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-Galërkin: 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 $\chi_m(z) := \det(z\mathbf{I}_m - \mathbf{C}_m)$.

Origin

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 three-term recurrence

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

almost always stops after $2n$ steps with the zero vector $\mathbf{f}_{2n} = \mathbf{o}_n$ (Sonneveld, 2006; Sonneveld, 2008).

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, nowadays known as “Sonneveld spaces”.

He thus called this property “Induced Dimension Reduction” (IDR), and algorithms like the given three-term recurrence “IDR Algorithms”.

IDR Theorem

Sonneveld first made experiments and then gave a rigorous proof. 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 $k \geqslant 1$

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

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. We remark that generically $\dim(\mathcal{K}_n(\mathbf{B}^{-\mathsf{H}}, \mathbf{B}^{-\mathsf{H}}\mathbf{p})) = n$.

Using the Krylov subspace point of view and the explicit orthogonalization against $\mathbf{p}$ before multiplication with $\mathbf{B}$, we see that indeed $\mathbf{f}_{2n} = \mathbf{B}\mathbf{o}_n = \mathbf{o}_n$.

IDR Algorithms

The three-term recurrence

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

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 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 iteration with a starting guess $\mathbf{x}_0$ is then given by

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

Primitive IDR

With $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, the Richardson 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

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

The update of the iterates has to be modified accordingly,

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

Primitive IDR

Sonneveld terms the outcome the Primitive IDR Algorithm (Sonneveld, 2006):

```
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{r}_0$
$\mathbf{r}_1 = \mathbf{r}_0 - \mathbf{A}\mathbf{r}_0$

For $k = 1, 2, \ldots$ do
    $\gamma_k = \mathbf{p}^\mathsf{T}\mathbf{r}_k / \mathbf{p}^\mathsf{T}(\mathbf{r}_{k-1} - \mathbf{r}_k)$
    $\mathbf{s}_k = \mathbf{r}_k + \gamma_k(\mathbf{r}_k - \mathbf{r}_{k-1})$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k(\mathbf{x}_k - \mathbf{x}_{k-1}) + \mathbf{s}_k$
    $\mathbf{r}_{k+1} = \mathbf{s}_k - \mathbf{A}\mathbf{s}_k$
done
```

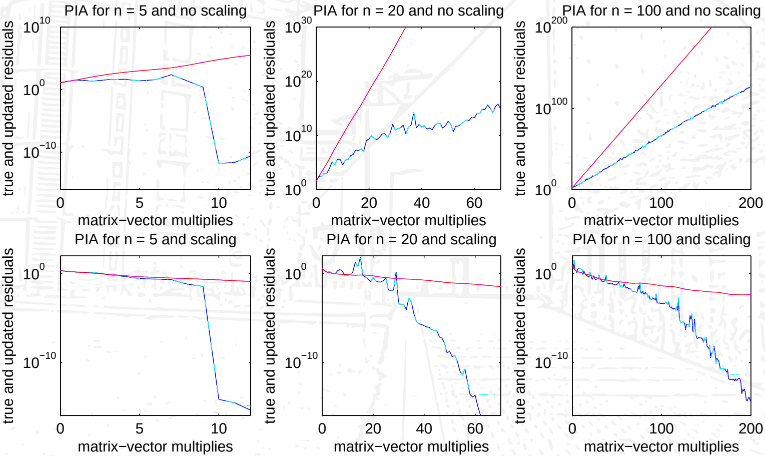
```
$\mathbf{x}_\text{old} = \mathbf{x}_0$
$\mathbf{r}_\text{old} = \mathbf{b} - \mathbf{A}\mathbf{x}_\text{old}$
$\mathbf{x}_\text{new} = \mathbf{x}_\text{old} + \mathbf{r}_\text{old}$
$\mathbf{r}_\text{new} = \mathbf{r}_\text{old} - \mathbf{A}\mathbf{r}_\text{old}$

While "not converged" do
    $\gamma = \mathbf{p}^\mathsf{T}\mathbf{r}_\text{new} / \mathbf{p}^\mathsf{T}(\mathbf{r}_\text{old} - \mathbf{r}_\text{new})$
    $\mathbf{s} = \mathbf{r}_\text{new} + \gamma(\mathbf{r}_\text{new} - \mathbf{r}_\text{old})$
    $\mathbf{x}_\text{tmp} = \mathbf{x}_\text{new} + \gamma(\mathbf{x}_\text{new} - \mathbf{x}_\text{old}) + \mathbf{s}$
    $\mathbf{r}_\text{tmp} = \mathbf{s} - \mathbf{A}\mathbf{s}$
    $\mathbf{x}_\text{old} = \mathbf{x}_\text{new}$, $\mathbf{x}_\text{new} = \mathbf{x}_\text{tmp}$
    $\mathbf{r}_\text{old} = \mathbf{r}_\text{new}$, $\mathbf{r}_\text{new} = \mathbf{r}_\text{tmp}$
done
```

On the next slide we compare Richardson iteration (red) and PIA (blue).

Primitive IDR

Impressions of “finite termination” and acceleration in finite precision:



Primitive IDR

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 (Sonneveld, 2008) 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 ω_j is fixed for two steps and otherwise could be chosen freely, but non-zero.

This algorithm with minimization of every second residual is included in the proceedings from 1980 (Wesseling and Sonneveld, 1980). The connection to Krylov methods, e.g., BiCG/Lanczos, is also given there.

Classical IDR

```
$\gamma_0 = 0$, $\mathbf{f}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$, $\Delta\mathbf{g}_0 = \mathbf{o}_n$, $\Delta\mathbf{y}_0 = \mathbf{o}_n$
For $k = 1, \dots$ do
  $\mathbf{s}_k = \mathbf{f}_{k-1} + \gamma_{k-1}\Delta\mathbf{g}_{k-1}$
  $\mathbf{t}_k = \mathbf{A}\mathbf{s}_k$
  if $k = 1$ or $k$ is even
    $\omega_k = (\mathbf{t}_k^{\mathsf{H}}\mathbf{s}_k)/(\mathbf{t}_k^{\mathsf{H}}\mathbf{t}_k)$
  else
    $\omega_k = \omega_{k-1}$
  end
  $\Delta\mathbf{x}_k = \gamma_{k-1}\Delta\mathbf{y}_{k-1} - \omega_k\mathbf{s}_k$
  $\Delta\,\mathbf{f}_k = \gamma_{k-1}\Delta\mathbf{g}_{k-1} - \omega_k\mathbf{t}_k$
  $\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta\mathbf{x}_k$
  $\mathbf{f}_k = \mathbf{f}_{k-1} + \Delta\mathbf{f}_k$
  if $k$ is even
    $\Delta\mathbf{y}_k = \Delta\mathbf{y}_{k-1}$
    $\Delta\mathbf{g}_k = \Delta\mathbf{g}_{k-1}$
  else
    $\Delta\mathbf{y}_k = \Delta\mathbf{x}_k$
    $\Delta\mathbf{g}_k = \Delta\mathbf{f}_k$
  end
  $\gamma_k = -(\mathbf{p}^{\mathsf{H}}\mathbf{f}_k)/(\mathbf{p}^{\mathsf{H}}\Delta\mathbf{g}_k)$
done
```

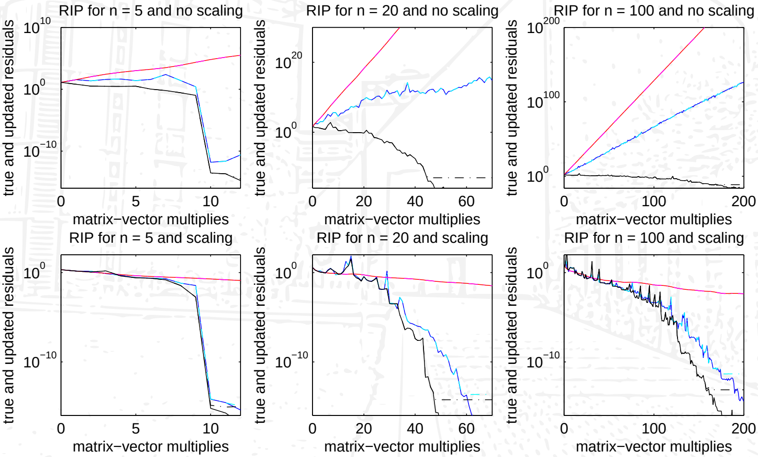
This is the original IDR Algorithm from page 551 of (Wesseling and Sonneveld, 1980).

It uses OrthoRes(1) in the first step and a residual (these are the $-\mathbf{f}_{2j}$) minimization every second step.

The finite termination property follows from a generalization of the IDR Theorem based on commutativity of the linear polynomials $\mathbf{I} - \omega_j\mathbf{A}$.

Classical IDR

A numerical comparison of Richardson iteration, original IDR, and PIA.



Brothers of classical and primitive IDR

In 1976 Sonneveld 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.

Some of these 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 . . .

Rebirth of IDR: IDR(s)

In 2006, 30 years after inventing IDR, Sonneveld together with van Gijzen reconsidered IDR and 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 $\mathbf{P} \in \mathbb{C}^{n \times s}$, $1 \leqslant s \leqslant n$.

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-Galërkin.

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) (without the recurrences for $\mathbf{x}_n$ and thus already slightly rewritten)

```
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
compute $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \dots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \dots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \dots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
while not converged
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    compute $\omega_j$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, \ n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \dots, \nabla\mathbf{r}_{n-1})$
    for $k = 1, \dots, s$
        $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
        $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
        $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
        $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, \ n \leftarrow n+1$
        $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \dots, \nabla\mathbf{r}_{n-1})$
    end for
    $j \leftarrow j+1$
end while
```

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 ω_j is computed.

IDR(s)ORes is based on oblique projections. and $s+1$ consecutive multiplications with the same linear factor $\mathbf{I} - \omega_j\mathbf{A}$.

Understanding IDR: Hessenberg decompositions

We already noted that essential features of Krylov subspace methods can be described by a Hessenberg decomposition

$$\mathbf{A}\mathbf{Q}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^{\mathsf{T}}. \tag{5}$$

Here, $\mathbf{H}_n$ 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

$$\mathbf{A}\mathbf{Q}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^{\mathsf{T}}. \tag{6}$$

The matrix $\mathbf{H}_n$ of the perturbed variant will, in general, still be unreduced.

IDR: Generalized Hessenberg decompositions

In case of IDR, we have to consider generalized Hessenberg decompositions

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^{\mathsf{T}} \tag{7}$$

and perturbed generalized Hessenberg decompositions

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^{\mathsf{T}} \tag{8}$$

with upper triangular (possibly even singular) $\mathbf{U}_n$.

Generalized Hessenberg decompositions correspond to an oblique projection of the pencil $(\mathbf{A}, \mathbf{I})$ to the pencil $(\mathbf{H}_n, \mathbf{U}_n)$ as long as $\mathbf{Q}_{n+1}$ has full rank,

$$\begin{aligned}\widehat{\mathbf{Q}}_n^{\mathsf{H}}(\mathbf{A}, \mathbf{I})\mathbf{Q}_n\mathbf{U}_n &= \widehat{\mathbf{Q}}_n^{\mathsf{H}}(\mathbf{A}\mathbf{Q}_n\mathbf{U}_n, \mathbf{Q}_n\mathbf{U}_n) \\ &= \widehat{\mathbf{Q}}_n^{\mathsf{H}}(\mathbf{Q}_{n+1}\underline{\mathbf{H}}_n, \mathbf{Q}_n\mathbf{U}_n) = (\underline{\mathbf{I}}_n^{\mathsf{T}}\underline{\mathbf{H}}_n, \mathbf{U}_n) = (\mathbf{H}_n, \mathbf{U}_n),\end{aligned} \tag{9}$$

where $\widehat{\mathbf{Q}}_n^{\mathsf{H}} := \underline{\mathbf{I}}_n^{\mathsf{T}}\mathbf{Q}_{n+1}^{\dagger}$.

Understanding IDR: OrthoRes-type methods

The entries of the Hessenberg matrices of these Hessenberg decompositions are defined in different variations. Three well-known ways for implementing the QOR/QMR approach are commonly denoted as OrthoRes, OrthoMin, 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

$$\mathbf{A}\mathbf{R}_n\mathbf{U}_n = \mathbf{R}_{n+1}\underline{\mathbf{H}}_n^\circ = \mathbf{R}_n\mathbf{H}_n^\circ + \mathbf{r}_{n+1}h_{n+1,n}^\circ\mathbf{e}_n^\mathsf{T}, \tag{10}$$

where $\mathbf{e}^\mathsf{T}\underline{\mathbf{H}}_n^\circ = \mathbf{o}_n^\mathsf{T}$, $\mathbf{e}^\mathsf{T} = (1, \ldots, 1)$., and the matrix

$$\mathbf{R}_{n+1} = (\mathbf{r}_0, \ldots, \mathbf{r}_n) = \mathbf{Q}_{n+1}\,\mathrm{diag}\left(\frac{\|\mathbf{r}_0\|_2}{\|\mathbf{q}_1\|_2}, \ldots, \frac{\|\mathbf{r}_n\|_2}{\|\mathbf{q}_{n+1}\|_2}\right) \tag{11}$$

is diagonally scaled to be the matrix of residual vectors.

IDR: The underlying Hessenberg decomposition

The IDR recurrences of IDR(s)ORes can be summarized by

$$\boxed{\begin{aligned} \mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= \left(1 - \gamma_s^{(n)}\right)\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}\left(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)}\right)\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ 1 \cdot \mathbf{r}_n &:= \left(\mathbf{I} - \omega_j\mathbf{A}\right)\mathbf{v}_{n-1}\,. \end{aligned}} \tag{12}$$

Here, $n > s$, and the index of the scalar ω_j is defined by

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

compare with the so-called “index functions” (Yeung/Boley, 2005).

Removing $\mathbf{v}_{n-1}$ from the recurrence we obtain the generalized Hessenberg decomposition

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{13}$$

IDR: Sonneveld pencil and Sonneveld matrix

The IDR(s)ORes pencil, the so-called Sonneveld pencil $(\mathbf{Y}_n^\circ, \mathbf{Y}_n\mathbf{D}_\omega^{(n)})$, can be depicted by

$$\left(\begin{array}{cccccccccccc}
\times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
+ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times
\end{array}\right), \quad
\left(\begin{array}{cccccccccccc}
\times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times
\end{array}\right).$$

The upper triangular matrix $\mathbf{Y}_n\mathbf{D}_\omega^{(n)}$ could be inverted, which results in the Sonneveld matrix, a full unreduced Hessenberg matrix.

Understanding IDR: Purification

We know the eigenvalues ≈ roots of kernel polynomials $1/\omega_j$. We are only interested in the other eigenvalues.

The purified IDR(s)ORes pencil $(\mathbf{Y}_n^\circ, \mathbf{U}_n\mathbf{D}_\omega^{(n)})$, 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 \\
+ & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & \times \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times
\end{pmatrix},
\begin{pmatrix}
\times & \times & \times & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \times & \times & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \times & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \times & \times & \times & {\color{blue}\circ} & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \times & \times & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \times & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & {\color{blue}\circ}
\end{pmatrix}.$$

We get rid of the infinite eigenvalues using a change of basis (Gauß/Schur).

Understanding IDR: Gaussian elimination

The deflated purified IDR(s)ORes pencil, after the elimination step $(\mathbf{Y}_n^\circ \mathbf{G}_n, \mathbf{U}_n \mathbf{D}_\omega^{(n)})$, can be depicted by

$$\begin{pmatrix}
\times & \times & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ & \circ & \circ & \circ & \circ \\
+ & \times & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ & \circ & \circ & \circ & \circ \\
\circ & + & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & {\color{red}\circ} & {\color{red}+} & {\color{red}\circ} & {\color{red}\circ} & {\color{red}\circ} & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & {\color{red}+} & {\color{red}+} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ \\
\circ & \circ & \circ & \circ & + & \times & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ \\
\circ & \circ & \circ & \circ & \circ & + & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & {\color{red}\circ} & {\color{red}+} & {\color{red}\circ} & {\color{red}\circ} & {\color{red}\circ} & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & {\color{red}+} & {\color{red}+} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & \times & {\color{red}\times} & {\color{red}\times} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & + & {\color{red}\times} & {\color{red}\times} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & {\color{red}\circ} & {\color{red}+}
\end{pmatrix}, \quad
\begin{pmatrix}
\times & \times & \times & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \times & \times & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \times & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \times & \times & \times & {\color{blue}\circ} & \circ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \times & \times & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \times & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & {\color{blue}\circ} & \circ \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \times & {\color{blue}\circ} \\
\circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & {\color{blue}\circ}
\end{pmatrix}.$$

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

Understanding IDR: Deflation

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 $(D(\mathbf{Y}_n^\circ \mathbf{G}_n), D(\mathbf{U}_n \mathbf{D}_\omega^{(n)}))$, can be depicted by

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

The block-diagonal matrix $D(\mathbf{U}_n \mathbf{D}_\omega^{(n)})$ has invertible upper triangular blocks and can be inverted to expose the underlying Lanczos process.

IDR: a Lanczos process with multiple left-hand sides

Inverting the block-diagonal matrix $D(\mathbf{U}_n\mathbf{D}_\omega^{(n)})$ gives an algebraic eigenvalue problem with a block-tridiagonal unreduced upper Hessenberg matrix

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

This is the matrix of the underlying BiORes$(s, 1)$ process.

The extended matrix version $\underline{\mathbf{L}}_n$ satisfies

$$\mathbf{A}\mathbf{Q}_n = \mathbf{Q}_{n+1}\underline{\mathbf{L}}_n,$$

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

IDR: a Lanczos process with multiple left-hand sides

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, $\mathbf{q}_{js+k} \perp \Omega_{j-1}(\mathbf{A}^{\mathsf{H}})\mathbf{P}$.

Using induction (Sleijpen et al., 2008) one can prove that $\mathbf{q}_{js+k} \perp \mathcal{K}_j(\mathbf{A}^{\mathsf{H}}, \mathbf{P})$; 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 ($\mathcal{S} := \mathbf{P}^\perp$)

$$\mathcal{G}_0 = \mathcal{K}(\mathbf{A}, \mathbf{r}_0), \quad \text{where } \mathcal{K}(\mathbf{A}, \mathbf{r}_0) \text{ denotes the } \textit{full} \text{ Krylov subspace,}$$

$$\mathcal{G}_j = \bigcap_{k=0}^{j-1} \Omega_k(\mathbf{A})^{-1}\Omega_j(\mathbf{A})(\mathcal{S}) = \Bigl(\sum_{k=0}^{j-1}{}^{+}\; \Omega_j(\mathbf{A})^{-\mathsf{H}}\Omega_k(\mathbf{A})^{\mathsf{H}}\{\mathbf{P}\}\Bigr)^\perp$$

$$= \Bigl(\Omega_j(\mathbf{A})^{-\mathsf{H}}\mathcal{K}_j(\mathbf{A}^{\mathsf{H}}, \mathbf{P})\Bigr)^\perp = \Omega_j(\mathbf{A})\Bigl(\mathcal{K}_j(\mathbf{A}^{\mathsf{H}}, \mathbf{P})\Bigr)^\perp$$

of the Sonneveld spaces.

IDR: a Lanczos process with multiple left-hand sides

This has to be compared with Theorem 4.2 in (Sleijpen et al., 2008) and with Theorem 4.1 in (Simoncini and Szyld, 2009) (similar result; 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 $\mathcal{S}=\mathbf{P}^\perp$,
- the last residuals are $\mathbf{I}-\omega_j\mathbf{A}$ images of linear combinations of previously obtained images $(\mathbf{I}-\omega_{j-1}\mathbf{A})\cdots(\mathbf{I}-\omega_k\mathbf{A})$ of $\mathcal{S}=\mathbf{P}^\perp$.

IDR: a Lanczos process with multiple left-hand sides

The second equality

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

is based on

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

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

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

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

$$\mathbf{y} \in \mathbf{B}\mathbf{P}^{\perp} \;\Leftrightarrow\; \left\{\mathbf{y} = \mathbf{B}\mathbf{v} \wedge \mathbf{P}^{\mathsf{H}}\mathbf{v} = \mathbf{o}_n\right\} \;\Leftrightarrow\; \mathbf{P}^{\mathsf{H}}\mathbf{v} = \mathbf{P}^{\mathsf{H}}\mathbf{B}^{-1}\mathbf{y} = (\mathbf{B}^{-\mathsf{H}}\mathbf{P})^{\mathsf{H}}\mathbf{y} = \mathbf{o}_n.$$

IDR: a Lanczos process with multiple left-hand sides

The third and fourth equality

$$\left(\overset{j-1}{\underset{k=0}{+}}\,\Omega_j(\mathbf{A})^{-\mathsf{H}}\,\Omega_k(\mathbf{A})^{\mathsf{H}}\{\mathbf{P}\}\right)^{\perp} = \left(\Omega_j(\mathbf{A})^{-\mathsf{H}}\,\mathcal{K}_j(\mathbf{A}^{\mathsf{H}},\mathbf{P})\right)^{\perp}$$
$$= \Omega_j(\mathbf{A})\left(\mathcal{K}_j(\mathbf{A}^{\mathsf{H}},\mathbf{P})\right)^{\perp}$$

are satisfied

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

Generalizations of IDR(s)

In the analysis of the similarities of and differences between IDR and ML(k)BiCGStab, Sonneveld and van Gijzen realized that the residuals computed last in a complete cycle are of importance.

In their new implementation IDR(s)BiO (van Gijzen and Sonneveld, 2008) of the IDR Theorem, they use basis vectors $\mathbf{g}_{-1}, \ldots, \mathbf{g}_{-s} \in \mathcal{G}_j$, which are not simply residual differences but linear combinations.

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

$$\mathbf{v}_n = \mathbf{r}_n - \sum_{i=1}^{s} \mathbf{g}_{n-i}\gamma_i =: \mathbf{r}_n - \mathbf{G}_n\mathbf{c}_n, \quad \text{and thus,}$$

$$\mathbf{r}_{n+1} = (\mathbf{I} - \omega\mathbf{A})\mathbf{v}_n = \mathbf{r}_n - \omega\mathbf{A}\mathbf{v}_n - \sum_{i=1}^{s} \mathbf{g}_{n-i}\gamma_i,$$

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

Generalizations of IDR(s)

Recently, the relations between IDR(s) and BiCGStab(ℓ) and combinations of both methods have been investigated.

- In (Sleijpen et al., 2008) the authors derive different implementations of ML(k)BiCGStab-like algorithms.
- In (Sleijpen and van Gijzen, 2009) the authors combine the IDR philosophy with higher degree stabilization polynomials. The resulting method is named IDR(s)Stab(ℓ). The approach is comparable to the one resulting in BiCGStab(ℓ).
- In (Tanio and Sugihara, 2009) the authors derive the algorithm GBiCGStab(s,L), which is similar to IDR(s)Stab(ℓ). 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 (Tanio and Sugihara, 2008).
- In (Sleijpen and Abe, 2010) the ideas behind BiCGStab2 (Gutknecht, 1993) and GPBiCG (Zhang, 1997) are considered.

Generalizations of IDR(s)

The relation of IDR to Petrov-Galërkin with a rational Krylov space motived the method IDR-Ritz (Simoncini and Szyld, 2009).

Another, simpler motivation is that the residual polynomials should be designed to dampen the spectrum. Using the residual polynomial representation of IDR 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 ω_j which are in the field of values of $\mathbf{A}^{-\mathsf{H}}$, 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 nonsymmetric matrices this typically results in an algorithm based on complex arithmetic in place of real arithmetic.

Generalizations of IDR(s)

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 ω_j are fixed by the splitting chosen.

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,
- `_dot`s (line search minimization),
- `_axpy`s or `_gemv`s (updates of vectors).

On the next slide we sketch the IDR(s) variant IDR(s)BiO described in the Technical Report (van Gijzen and Sonneveld, 2008). Its Matlab source code can be downloaded from `http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html`.

This version of IDR(s) is contained in release 3.0 of the IFISS package.

Structure of IDR(*s*) – unpreconditioned IDR(*s*)BiO

```
x = x0; r = b - A*x; ω = 1; PT = P';
G = zeros(n,s); U = zeros(n,s); M = eye(s);
while "not converged"
  PTr = PT*r;
  for k = 1:s
% Solve small system and make v orthogonal to P:
    c = M(k:s,k:s)\PTr(k:s);
    v = r - G(:,k:s)*c;
    U(:,k) = U(:,k:s)*c + ω*v;
% Compute G(:,k) = A U(:,k)
    G(:,k) = A*U(:,k);
% Bi-Orthogonalize the new basis vectors:
    for j = 1:k-1
       α = (PT(j,:)*G(:,k))/M(j,j);
       G(:,k) = G(:,k) - α*G(:,j);
       U(:,k) = U(:,k) - α*U(:,j);
    end
% New column of M = P'*G (first k-1 entries are zero)
    M(k:s,k) = PT(k:s,:)*G(:,k);
% Make r orthogonal to p_j, j = 1,...,k
    β = PTr(k)/M(k,k);
    r = r - β*G(:,k); x = x + β*U(:,k);
% New PTr = P'*r (first k components are zero)
    if k < s
       PTr(k+1:s) = PTr(k+1:s) - β*M(k+1:s,k);
    end
  end
% Note: r is already perpendicular to P so v = r
  v = r; t = A*v;
  select or compute ω;
  r = r - ω*t; x = x + ω*v;
end
```

Parallelization possible, e.g.: {Sca,P}LAPACK, CUBLAS/CUDA, cloud computing, . . .

A naïve CUBLAS implementation would be based on

- parallel evaluation of small sized problems,
- several calls to `cublasSaxpy()`,
- several calls to `cublasSgemv()`,
- several calls to `cublasSdot()`,
- several calls to `cublasScopy()`.

Structure of IDR(s) – unpreconditioned IDR(s)BiO

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 `_dot`s from orthogonalization and (possibly) the computation of ω_j and (possibly) the computation of the norm of the residual or backward error.

Structure of IDR(s) – unpreconditioned IDR(s)BiO

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 reorthogonalization (Hernández et al., 2006) 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 places elsewhere.

Similarly to the approaches used in parallel implementations of CG (Meurant, 1987; D'Azevedo et al., 1993) 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 ω by utilization of algebraic rewritings.

Structure of IDR(s)

We can quite easily get rid of the synchronization points caused by the computation of the non-zero scalars ω_j. One idea is to precompute a certain amount of Ritz values like in (Simoncini and Szyld, 2009) or to use the CPU (or some nodes of the GPU) to compute rough approximations to eigenvalues based on the Sonneveld pencil.

If we use a method like 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* IDR(s) method. This should be optimized using code adopted to the architecture we are working on.

Structure of $\mathbf{P}$ and 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 $\mathbf{p}_j$ discretizations 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 ± 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 careful balance the benefits and the risks of resulting instabilities.

Structure of IDR(s)Eig

As IDR(s)Eig is based on (a given variant) of 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 ω_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 IDR(s)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 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.

Conclusions and Outlook

- We gave a short introduction to Krylov subspace methods.
- We sketched the IDR and IDR(s) families.
- We indicated how to compute eigenvalues using one particular instance of IDR(s), namely, IDR(s)ORes.
- We sketched the relation between IDR(s)ORes, the eigenvalue routine IDR(s)Eig based on it, and a two-sided Lanczos process BiORes(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 remains to be done . . .

Thank you very much for your attention!

どうも有難う御座いました。

D'Azevedo, E. F., Eijkhout, V., and Romine, C. H. (1993).
A matrix framework for conjugate gradient methods and some variants of cg with less synchronization overhead.
In *PPSC*, pages 644–646.

Gutknecht, M. H. (1993).
Variants of BICGSTAB for matrices with complex spectrum.
SIAM J. Sci. Comput., 14(5):1020–1033.

Gutknecht, M. H. and Zemke, J.-P. M. (2010).
Eigenvalue computations based on IDR.
Technical Report (to appear 2010).

Hernández, V., Román, J. E., and Tomás, A. (2006).
A parallel variant of the Gram-Schmidt process with reorthogonalization.
In Joubert, G. R., Nagel, W. E., Peters, F. J., Plata, O. G., Tirado, P., and Zapata, E. L., editors, *Proceedings of the International Conference on Parallel Computing (ParCo 2005)*, volume 33, pages 221–228. Central Institute for Applied Mathematics, Jülich, Germany.

Meurant, G. (1987).
Multitasking the conjugate gradient method on the CRAY X-MP/48.
Parallel Comput., 5(3):267–280.

Simoncini, V. and Szyld, D. (2009).
Interpreting IDR as a Petrov-Galerkin method.
Report 09-10-22, Dipartimento di Matematica, Università di Bologna and Department of Mathematics, Temple University, Philadelphia.

Sleijpen, G. L. G. and Abe, K. (2010).
Publication in preparation (January 2010).

Sleijpen, G. L. G., Sonneveld, P., and van Gijzen, M. B. (2008).
Bi-CGSTAB as an induced dimension reduction method.
Reports of the Department of Applied Mathematical Analysis Report 08-07, Delft University of Technology.
ISSN 1389-6520.

Sleijpen, G. L. G. and van Gijzen, M. B. (2009).
Exploiting BiCGstab(ℓ) strategies to induce dimension reduction.
Reports of the Department of Applied Mathematical Analysis Report 09-02, Delft University of Technology.
ISSN 1389-6520.

Sonneveld, P. (2006).
History of IDR: an example of serendipity.
PDF file sent by Peter Sonneveld on Monday, 24th of July 2006.
8 pages; evolved into (Sonneveld, 2008).

Sonneveld, P. (2008).
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.

Sonneveld, P. and van Gijzen, M. B. (2008).
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.
Received Mar. 20, 2007.

Tanio, M. and Sugihara, M. (2008).
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.
(In Japanese).

Tanio, M. and Sugihara, M. (2009).
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.

van Gijzen, M. B. and Sonneveld, P. (2008).
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.
ISSN 1389-6520.

Wesseling, P. and Sonneveld, P. (1980).
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.

Zhang, S.-L. (1997).
GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems.
SIAM Journal on Scientific Computing, 18(2):537–551.