On the behavior of IDR(s) algorithms in finite precision

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

Institut für Numerische Simulation
Technische Universität Hamburg-Harburg

Kyushu University, Fukuoka
2010/02/09

Outline

Krylov methods in finite precision

. . . an introduction

IDR and IDR(s)

1976–1980: IDR

2006–2010: IDR(s)

IDR: two close relatives

IDR and Lanczos

IDR and Lanczos-type product methods

Numerical experiments

Many pictures — less mathematics

Outline

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).

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,

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

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,

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

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 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.

The origin of Krylov subspace methods

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

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

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 $\mathbf{K}_m =: \mathbf{Q}_m\mathbf{R}_m$,

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

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 $\mathbf{K}_m =: \mathbf{Q}_m\mathbf{R}_m$,

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

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

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

is unreduced extended Hessenberg.

The connection to Hessenberg decompositions

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

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

is unreduced extended Hessenberg.

We end up with a Hessenberg decomposition

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

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 $\mathbf{C}_m\mathbf{S}_m = \mathbf{S}_m\mathbf{J}_m$.

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.

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 $\mathcal{R}_m(z) := \det\left(\mathbf{I}_m - z\mathbf{C}_m^{-1}\right)$,

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 $\mathcal{R}_m(z) := \det\left(\mathbf{I}_m - z\mathbf{C}_m^{-1}\right)$,
$\mathbf{x}_m = \mathcal{L}_m[z^{-1}](\mathbf{A})\mathbf{r}_0$,

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

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 $\underline{\mathcal{R}}_m(z) := \det(\mathbf{I}_m - z\underline{\mathbf{C}}_m^{\dagger}\underline{\mathbf{I}}_m)$,

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 $\underline{\mathcal{R}}_m(z) := \det(\mathbf{I}_m - z\underline{\mathbf{C}}_m^{\dagger}\underline{\mathbf{I}}_m)$,
$\underline{\mathbf{x}}_m = \underline{\mathcal{L}}_m[z^{-1}](\mathbf{A})\mathbf{r}_0$,

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,

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 $\mathbf{v}_j^{(m)} = \mathcal{A}_m(\theta_j, \mathbf{A})\mathbf{q}_1$,

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 $\mathcal{A}_m(\theta, z) := (\chi_m(\theta) - \chi_m(z))(\theta - z)^{-1}$, $\theta \neq z$,

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

Implementation of Krylov methods

These Hessenberg decompositions are (more or less explicitely) constructed using linear algebra techniques (e.g., orthogonal and oblique projectors).

Implementation of Krylov methods

These Hessenberg decompositions are (more or less explicitely) constructed using linear algebra techniques (e.g., orthogonal and oblique projectors).

In finite precision the recurrence will only approximately be satisfied,

$$\begin{aligned} \mathbf{A}\mathbf{Q}_m + \mathbf{F}_m &= \mathbf{Q}_{m+1}\underline{\mathbf{C}}_m \\ &= \mathbf{Q}_m\mathbf{C}_m + \mathbf{q}_{m+1}\mathbf{c}_{m+1,m}\mathbf{e}_m^T, \end{aligned} \tag{5}$$

where the perturbation term $\mathbf{F}_m$ is in some sense "small" and/or structured.

Implementation of Krylov methods

These Hessenberg decompositions are (more or less explicitely) constructed using linear algebra techniques (e.g., orthogonal and oblique projectors).

In finite precision the recurrence will only approximately be satisfied,

$$\begin{aligned} \mathbf{A}\mathbf{Q}_m + \mathbf{F}_m &= \mathbf{Q}_{m+1}\underline{\mathbf{C}}_m \\ &= \mathbf{Q}_m\mathbf{C}_m + \mathbf{q}_{m+1}\mathbf{c}_{m+1,m}\mathbf{e}_m^T, \end{aligned} \tag{5}$$

where the perturbation term $\mathbf{F}_m$ is in some sense "small" and/or structured.

Equations like Eqn. (5) will be called perturbed Hessenberg decompositions.

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,
- adjugate polynomials $\mathcal{A}_m$,

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,
- adjugate polynomials $\mathcal{A}_m$,
- Lagrange interpolation polynomials $\mathcal{L}_m[z^{-1}]$ and $\underline{\mathcal{L}}_m[z^{-1}]$,

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,
- adjugate polynomials $\mathcal{A}_m$,
- Lagrange interpolation polynomials $\mathcal{L}_m[z^{-1}]$ and $\underline{\mathcal{L}}_m[z^{-1}]$,
- Lagrange interpolation polynomials $\mathcal{L}_m[1-\delta_{z0}]$ and $\underline{\mathcal{L}}_m[1-\delta_{z0}]$,

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,
- adjugate polynomials $\mathcal{A}_m$,
- Lagrange interpolation polynomials $\mathcal{L}_m[z^{-1}]$ and $\underline{\mathcal{L}}_m[z^{-1}]$,
- Lagrange interpolation polynomials $\mathcal{L}_m[1-\delta_{z0}]$ and $\underline{\mathcal{L}}_m[1-\delta_{z0}]$,
- residual polynomials $\mathcal{R}_m$ and $\underline{\mathcal{R}}_m$.

The polynomial point of view

To understand the perturbed case, we generalize and extend the polynomials based on the computed $\mathbf{C}_m$ (or $\underline{\mathbf{C}}_m$) with their useful properties.

The polynomials are named by their property. In (Z., 2007) we considered the following five types of polynomials:

- basis polynomials $\mathcal{B}_m$,
- adjugate polynomials $\mathcal{A}_m$,
- Lagrange interpolation polynomials $\mathcal{L}_m[z^{-1}]$ and $\underline{\mathcal{L}}_m[z^{-1}]$,
- Lagrange interpolation polynomials $\mathcal{L}_m[1-\delta_{z0}]$ and $\underline{\mathcal{L}}_m[1-\delta_{z0}]$,
- residual polynomials $\mathcal{R}_m$ and $\underline{\mathcal{R}}_m$.

In this talk, we restrict ourselves to $\mathcal{L}_m[z^{-1}]$, $\mathcal{L}_m[1-\delta_{z0}]$ and $\mathcal{R}_m$.

Adjugate polynomials

First we consider certain bivariate polynomials — the adjugate polynomials.

Adjugate polynomials

First we consider certain bivariate polynomials — the adjugate polynomials.

- Property:

$$\mathcal{A}_m(z, \mathbf{C}_m) = \mathrm{adj}(z\mathbf{I}_m - \mathbf{C}_m).$$

Adjugate polynomials

First we consider certain bivariate polynomials — the adjugate polynomials.

- Property:

$$\mathcal{A}_m(z, \mathbf{C}_m) = \mathrm{adj}(z\mathbf{I}_m - \mathbf{C}_m).$$

- This implies for all eigenvalues (Ritz values) θ_j of $\mathbf{C}_m$ (Z., 2006)

$$\mathcal{A}_m(\theta_j, \mathbf{C}_m)\mathbf{e}_1 = \mathbf{s}_j, \quad \mathbf{C}_m\mathbf{s}_j = \mathbf{s}_j\theta_j.$$

Adjugate polynomials

First we consider certain bivariate polynomials — the adjugate polynomials.

- Property:

$$\mathcal{A}_m(z, \mathbf{C}_m) = \mathrm{adj}(z\mathbf{I}_m - \mathbf{C}_m).$$

- This implies for all eigenvalues (Ritz values) θ_j of $\mathbf{C}_m$ (Z., 2006)

$$\mathcal{A}_m(\theta_j, \mathbf{C}_m)\mathbf{e}_1 = \mathbf{s}_j, \quad \mathbf{C}_m\mathbf{s}_j = \mathbf{s}_j\theta_j.$$

- Definition:

$$\mathcal{A}_m(\theta, z) := \frac{\chi_m(\theta) - \chi_m(z)}{\theta - z}.$$

Adjugate polynomials

First we consider certain bivariate polynomials — the adjugate polynomials.

- Property:

$$\mathcal{A}_m(z, \mathbf{C}_m) = \mathrm{adj}(z\mathbf{I}_m - \mathbf{C}_m).$$

- This implies for all eigenvalues (Ritz values) θ_j of $\mathbf{C}_m$ (Z., 2006)

$$\mathcal{A}_m(\theta_j, \mathbf{C}_m)\mathbf{e}_1 = \mathbf{s}_j, \quad \mathbf{C}_m\mathbf{s}_j = \mathbf{s}_j\theta_j.$$

- Definition:

$$\mathcal{A}_m(\theta, z) := \frac{\chi_m(\theta) - \chi_m(z)}{\theta - z}.$$

- Generalization:

$$\mathcal{A}_{\ell+1:m}(\theta, z) := \frac{\chi_{\ell+1:m}(\theta) - \chi_{\ell+1:m}(z)}{\theta - z}, \quad \ell = 0, 1, \ldots, m.$$

Lagrange polynomials

Next we consider Lagrange interpolation polynomials interpolating the inverse function and a singularly perturbed identity function.

Lagrange polynomials

Next we consider Lagrange interpolation polynomials interpolating the inverse function and a singularly perturbed identity function.

The Lagrange interpolation of the inverse is denoted by $\mathcal{L}_m[z^{-1}](z)$.

Lagrange polynomials

Next we consider Lagrange interpolation polynomials interpolating the inverse function and a singularly perturbed identity function.

The Lagrange interpolation of the inverse is denoted by $\mathcal{L}_m[z^{-1}](z)$.

- Property:

$$\mathcal{L}_m[z^{-1}](\mathbf{C}_m) = \mathbf{C}_m^{-1}.$$

Lagrange polynomials

Next we consider Lagrange interpolation polynomials interpolating the inverse function and a singularly perturbed identity function.

The Lagrange interpolation of the inverse is denoted by $\mathcal{L}_m[z^{-1}](z)$.

- Property:

$$\mathcal{L}_m[z^{-1}](\mathbf{C}_m) = \mathbf{C}_m^{-1}.$$

- Definition:

$$\mathcal{L}_m[z^{-1}](z) := \frac{\chi_m(0) - \chi_m(z)}{z\chi_m(0)} = -\frac{\mathcal{A}_m(0,z)}{\chi_m(0)}.$$

Lagrange polynomials

Next we consider Lagrange interpolation polynomials interpolating the inverse function and a singularly perturbed identity function.

The Lagrange interpolation of the inverse is denoted by $\mathcal{L}_m[z^{-1}](z)$.

- Property:

$$\mathcal{L}_m[z^{-1}](\mathbf{C}_m) = \mathbf{C}_m^{-1}.$$

- Definition:

$$\mathcal{L}_m[z^{-1}](z) := \frac{\chi_m(0) - \chi_m(z)}{z\chi_m(0)} = -\frac{\mathcal{A}_m(0,z)}{\chi_m(0)}.$$

- Generalization:

$$\mathcal{L}_{\ell+1:m}[z^{-1}](z) := \frac{\chi_{\ell+1:m}(0) - \chi_{\ell+1:m}(z)}{z\chi_{\ell+1:m}(0)} = -\frac{\mathcal{A}_{\ell+1:m}(0,z)}{\chi_{\ell+1:m}(0)}, \quad \ell = 0, 1, \ldots, m.$$

Lagrange polynomials and QOR iterates

Theorem (The finite precision QOR iterates)

Suppose that all $\mathbf{C}_{\ell+1:m}$ *are regular. We define the* m*th QOR solution by*

$$\mathbf{z}_m := \mathbf{C}_m^{-1}\mathbf{e}_1\|\mathbf{r}_0\|_2$$

and the m*th QOR iterate by*

$$\mathbf{x}_m := \mathbf{Q}_m\mathbf{z}_m.$$

Then

$$\mathbf{x}_m = \mathcal{L}_m[z^{-1}](\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\,\mathcal{L}_{\ell+1:m}[z^{-1}](\mathbf{A})\mathbf{f}_\ell. \tag{6}$$

Lagrange polynomials and QOR iterates

Theorem (The finite precision QOR iterates)

Suppose that all $\mathbf{C}_{\ell+1:m}$ *are regular. We define the* m*th QOR solution by*

$$\mathbf{z}_m := \mathbf{C}_m^{-1}\mathbf{e}_1\|\mathbf{r}_0\|_2$$

and the m*th QOR iterate by*

$$\mathbf{x}_m := \mathbf{Q}_m\mathbf{z}_m.$$

Then

$$\mathbf{x}_m = \mathcal{L}_m[z^{-1}](\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\,\mathcal{L}_{\ell+1:m}[z^{-1}](\mathbf{A})\mathbf{f}_\ell. \tag{6}$$

Really sloppily speaking, in case of convergence,

$$\mathbf{x}_\infty = \mathbf{A}^{-1}\mathbf{r}_0 + \mathbf{A}^{-1}\mathbf{F}_\infty\mathbf{z}_\infty = \mathbf{A}^{-1}(\mathbf{r}_0 + \mathbf{F}_\infty\mathbf{z}_\infty).$$

Lagrange polynomials and QOR iterates

Theorem (The finite precision QOR iterates)

Suppose that all $\mathbf{C}_{\ell+1:m}$ *are regular. We define the* m*th QOR solution by*

$$\mathbf{z}_m := \mathbf{C}_m^{-1}\mathbf{e}_1\|\mathbf{r}_0\|_2$$

and the m*th QOR iterate by*

$$\mathbf{x}_m := \mathbf{Q}_m\mathbf{z}_m.$$

Then

$$\mathbf{x}_m = \mathcal{L}_m[z^{-1}](\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\,\mathcal{L}_{\ell+1:m}[z^{-1}](\mathbf{A})\mathbf{f}_\ell. \tag{6}$$

Really sloppily speaking, in case of convergence,

$$\mathbf{x}_\infty = \mathbf{A}^{-1}\mathbf{r}_0 + \mathbf{A}^{-1}\mathbf{F}_\infty\mathbf{z}_\infty = \mathbf{A}^{-1}(\mathbf{r}_0 + \mathbf{F}_\infty\mathbf{z}_\infty).$$

Proving convergence is the hard task.

Lagrange polynomials (continued)

We consider Lagrange interpolation polynomials interpolating the inverse and a singularly perturbed identity.

Lagrange polynomials (continued)

We consider Lagrange interpolation polynomials interpolating the inverse and a singularly perturbed identity. The Lagrange interpolation polynomial of the singularly perturbed identity is denoted by $\mathcal{L}_m^0[1-\delta_{z0}](z)$.

Lagrange polynomials (continued)

We consider Lagrange interpolation polynomials interpolating the inverse and a singularly perturbed identity. The Lagrange interpolation polynomial of the singularly perturbed identity is denoted by $\mathcal{L}_m^0[1-\delta_{z0}](z)$.

- Properties:

$$\mathcal{L}_m^0[1-\delta_{z0}](\mathbf{C}_m) = \mathbf{I}_m, \quad \mathcal{L}_m^0[1-\delta_{z0}](0) = 0.$$

Lagrange polynomials (continued)

We consider Lagrange interpolation polynomials interpolating the inverse and a singularly perturbed identity. The Lagrange interpolation polynomial of the singularly perturbed identity is denoted by $\mathcal{L}_m^0[1-\delta_{z0}](z)$.

- Properties:

$$\mathcal{L}_m^0[1-\delta_{z0}](\mathbf{C}_m) = \mathbf{I}_m, \quad \mathcal{L}_m^0[1-\delta_{z0}](0) = 0.$$

- Definition:

$$\mathcal{L}_m^0[1-\delta_{z0}](z) := \frac{\chi_m(0)-\chi_m(z)}{\chi_m(0)} = \mathcal{L}_m[z^{-1}](z)z.$$

Lagrange polynomials (continued)

We consider Lagrange interpolation polynomials interpolating the inverse and a singularly perturbed identity. The Lagrange interpolation polynomial of the singularly perturbed identity is denoted by $\mathcal{L}_m^0[1-\delta_{z0}](z)$.

- Properties:

$$\mathcal{L}_m^0[1-\delta_{z0}](\mathbf{C}_m) = \mathbf{I}_m, \quad \mathcal{L}_m^0[1-\delta_{z0}](0) = 0.$$

- Definition:

$$\mathcal{L}_m^0[1-\delta_{z0}](z) := \frac{\chi_m(0)-\chi_m(z)}{\chi_m(0)} = \mathcal{L}_m[z^{-1}](z)z.$$

- Generalization:

$$\mathcal{L}_{\ell+1:m}^0[1-\delta_{z0}](z) := \frac{\chi_{\ell+1:m}(0)-\chi_{\ell+1:m}(z)}{\chi_{\ell+1:m}(0)} = \mathcal{L}_{\ell+1:m}[z^{-1}](z)z, \quad \ell = 0,1,\ldots,m.$$

Residual polynomials

Last but not least we consider the well-known residual polynomials (Stiefel, 1955) denoted by $\mathcal{R}_m(z)$.

Residual polynomials

Last but not least we consider the well-known residual polynomials (Stiefel, 1955) denoted by $\mathcal{R}_m(z)$.

- Properties:

$$\mathcal{R}_m(\mathbf{C}_m) = \mathbf{O}_m, \quad \mathcal{R}_m(0) = 1.$$

Residual polynomials

Last but not least we consider the well-known residual polynomials (Stiefel, 1955) denoted by $\mathcal{R}_m(z)$.

- Properties:

$$\mathcal{R}_m(\mathbf{C}_m) = \mathbf{O}_m, \quad \mathcal{R}_m(0) = 1.$$

- Definition:

$$\mathcal{R}_m(z) := \frac{\chi_m(z)}{\chi_m(0)} = 1 - \mathcal{L}_m^0[1 - \delta_{z0}](z) = \det(\mathbf{I}_m - z\mathbf{C}_m^{-1}).$$

Residual polynomials

Last but not least we consider the well-known residual polynomials (Stiefel, 1955) denoted by $\mathcal{R}_m(z)$.

- Properties:

$$\mathcal{R}_m(\mathbf{C}_m) = \mathbf{O}_m, \quad \mathcal{R}_m(0) = 1.$$

- Definition:

$$\mathcal{R}_m(z) := \frac{\chi_m(z)}{\chi_m(0)} = 1 - \mathcal{L}_m^0[1 - \delta_{z0}](z) = \det(\mathbf{I}_m - z\mathbf{C}_m^{-1}).$$

- Generalization:

$$\mathcal{R}_{\ell+1:m}(z) := \frac{\chi_{\ell+1:m}(z)}{\chi_{\ell+1:m}(0)} = 1 - \mathcal{L}_{\ell+1:m}^0[1 - \delta_{z0}](z), \quad \ell = 0, 1, \ldots, m.$$

Residual polynomials

Last but not least we consider the well-known residual polynomials (Stiefel, 1955) denoted by $\mathcal{R}_m(z)$.

- Properties:

$$\mathcal{R}_m(\mathbf{C}_m) = \mathbf{O}_m, \quad \mathcal{R}_m(0) = 1.$$

- Definition:

$$\mathcal{R}_m(z) := \frac{\chi_m(z)}{\chi_m(0)} = 1 - \mathcal{L}_m^0[1 - \delta_{z0}](z) = \det(\mathbf{I}_m - z\mathbf{C}_m^{-1}).$$

- Generalization:

$$\mathcal{R}_{\ell+1:m}(z) := \frac{\chi_{\ell+1:m}(z)}{\chi_{\ell+1:m}(0)} = 1 - \mathcal{L}_{\ell+1:m}^0[1 - \delta_{z0}](z), \quad \ell = 0, 1, \ldots, m.$$

Two types of polynomials $\Rightarrow$ two expressions for the QOR residuals.

Residual polynomials and QOR residuals

Theorem (The finite precision QOR residuals)

Suppose that $\mathbf{q}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$ *and that all* $\mathbf{C}_{\ell+1:m}$ *are regular. Let* $\mathbf{x}_m$ *denote the* m*th QOR iterate and* $\mathbf{r}_m := \mathbf{r}_0 - \mathbf{A}\mathbf{x}_m$ *the corresponding residual.*

Then

$$\begin{aligned}\mathbf{r}_m &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 + \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{L}^0_{\ell+1:m}[1-\delta_{z0}](\mathbf{A})\mathbf{f}_\ell \\ &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{R}_{\ell+1:m}(\mathbf{A})\mathbf{f}_\ell + \mathbf{F}_m\mathbf{z}_m.\end{aligned} \tag{7}$$

Residual polynomials and QOR residuals

Theorem (The finite precision QOR residuals)

Suppose that $\mathbf{q}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$ *and that all* $\mathbf{C}_{\ell+1:m}$ *are regular. Let* $\mathbf{x}_m$ *denote the* m*th QOR iterate and* $\mathbf{r}_m := \mathbf{r}_0 - \mathbf{A}\mathbf{x}_m$ *the corresponding residual.*

Then

$$\begin{aligned}\mathbf{r}_m &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 + \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{L}^0_{\ell+1:m}[1-\delta_{z0}](\mathbf{A})\mathbf{f}_\ell \\ &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{R}_{\ell+1:m}(\mathbf{A})\mathbf{f}_\ell + \mathbf{F}_m\mathbf{z}_m.\end{aligned} \tag{7}$$

The first equation is related to the perturbation amplification.

Residual polynomials and QOR residuals

Theorem (The finite precision QOR residuals)

Suppose that $\mathbf{q}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$ *and that all* $\mathbf{C}_{\ell+1:m}$ *are regular. Let* $\mathbf{x}_m$ *denote the* m*th QOR iterate and* $\mathbf{r}_m := \mathbf{r}_0 - \mathbf{A}\mathbf{x}_m$ *the corresponding residual.*

Then

$$\begin{aligned}\mathbf{r}_m &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 + \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{L}^0_{\ell+1:m}[1-\delta_{z0}](\mathbf{A})\mathbf{f}_\ell \\ &= \mathcal{R}_m(\mathbf{A})\mathbf{r}_0 - \sum_{\ell=1}^{m} \mathbf{z}_{\ell m}\mathcal{R}_{\ell+1:m}(\mathbf{A})\mathbf{f}_\ell + \mathbf{F}_m\mathbf{z}_m.\end{aligned} \tag{7}$$

The first equation is related to the perturbation amplification.

The second equation is related to the attainable accuracy.

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

- the computed Ritz values θ_j, $1 \leqslant j \leqslant m$,

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

- the computed Ritz values θ_j, $1 \leqslant j \leqslant m$,
- the components of the computed QOR solutions $\mathbf{z}_m$, and

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

- the computed Ritz values θ_j, $1 \leqslant j \leqslant m$,
- the components of the computed QOR solutions $\mathbf{z}_m$, and
- the perturbation terms $\mathbf{f}_\ell$, $1 \leqslant \ell \leqslant m$.

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

- the computed Ritz values θ_j, $1 \leqslant j \leqslant m$,
- the components of the computed QOR solutions $\mathbf{z}_m$, and
- the perturbation terms $\mathbf{f}_\ell$, $1 \leqslant \ell \leqslant m$.

The components of the QOR solutions are at hand in the course of the computation. The perturbation terms can be described using bounds on the errors introduced by the execution of the algorithm in finite precision, followed by some simple algebraic manipulations.

Implications for the analysis of IDR/IDR(s)

We have shown that the behavior of a perturbed QOR Krylov subspace method can completely be described in terms of

- the computed Ritz values θ_j, $1 \leqslant j \leqslant m$,
- the components of the computed QOR solutions $\mathbf{z}_m$, and
- the perturbation terms $\mathbf{f}_\ell$, $1 \leqslant \ell \leqslant m$.

The components of the QOR solutions are at hand in the course of the computation. The perturbation terms can be described using bounds on the errors introduced by the execution of the algorithm in finite precision, followed by some simple algebraic manipulations.

Thus, we need to understand how the Ritz values behave in finite precision.

Outline

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).

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”.

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 $\mathcal{S} := \mathbf{p}^{\perp}$.

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 $k \geqslant 1$

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

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 $k \geqslant 1$

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

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

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 $k \geqslant 1$

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

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. 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

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

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".

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 iteration form,

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

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 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:

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

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

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

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

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

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

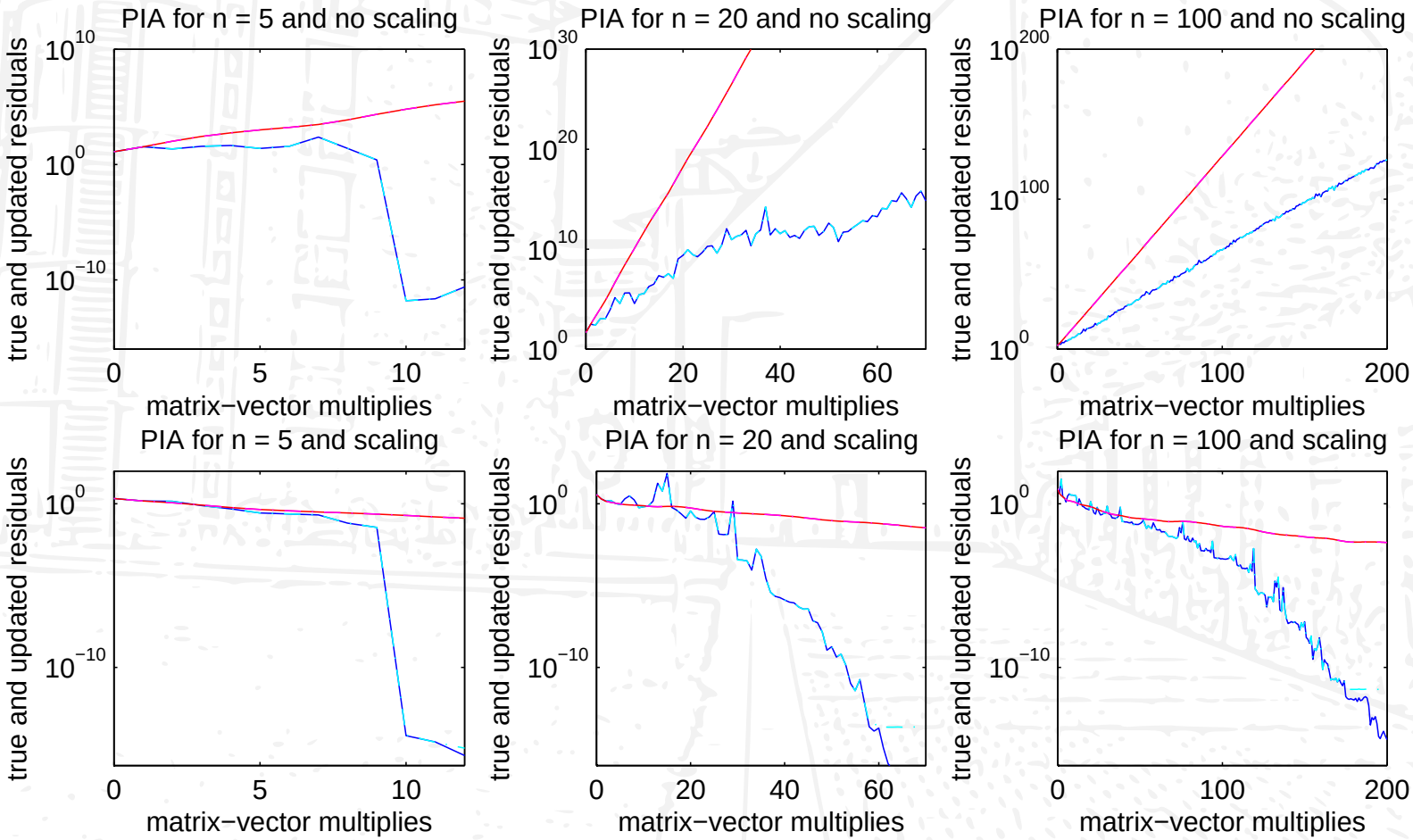
$$\gamma = \mathbf{p}^{\mathsf{T}}\mathbf{r}_{\text{new}} / \mathbf{p}^{\mathsf{T}}(\mathbf{r}_{\text{old}} - \mathbf{r}_{\text{new}})$$
$$\mathbf{s} = \mathbf{r}_{\text{new}} + \gamma(\mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}})$$
$$\mathbf{x}_{\text{tmp}} = \mathbf{x}_{\text{new}} + \gamma(\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}}) + \mathbf{s}$$
$$\mathbf{r}_{\text{tmp}} = \mathbf{s} - \mathbf{A}\mathbf{s}$$
$$\mathbf{x}_{\text{old}} = \mathbf{x}_{\text{new}},\ \mathbf{x}_{\text{new}} = \mathbf{x}_{\text{tmp}}$$
$$\mathbf{r}_{\text{old}} = \mathbf{r}_{\text{new}},\ \mathbf{r}_{\text{new}} = \mathbf{r}_{\text{tmp}}$$

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.

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.

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.

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 $k = 1, \ldots$ 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

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 $k = 1, \ldots$ 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).

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 $k = 1, \ldots$ 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.

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 $k = 1, \ldots$ 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

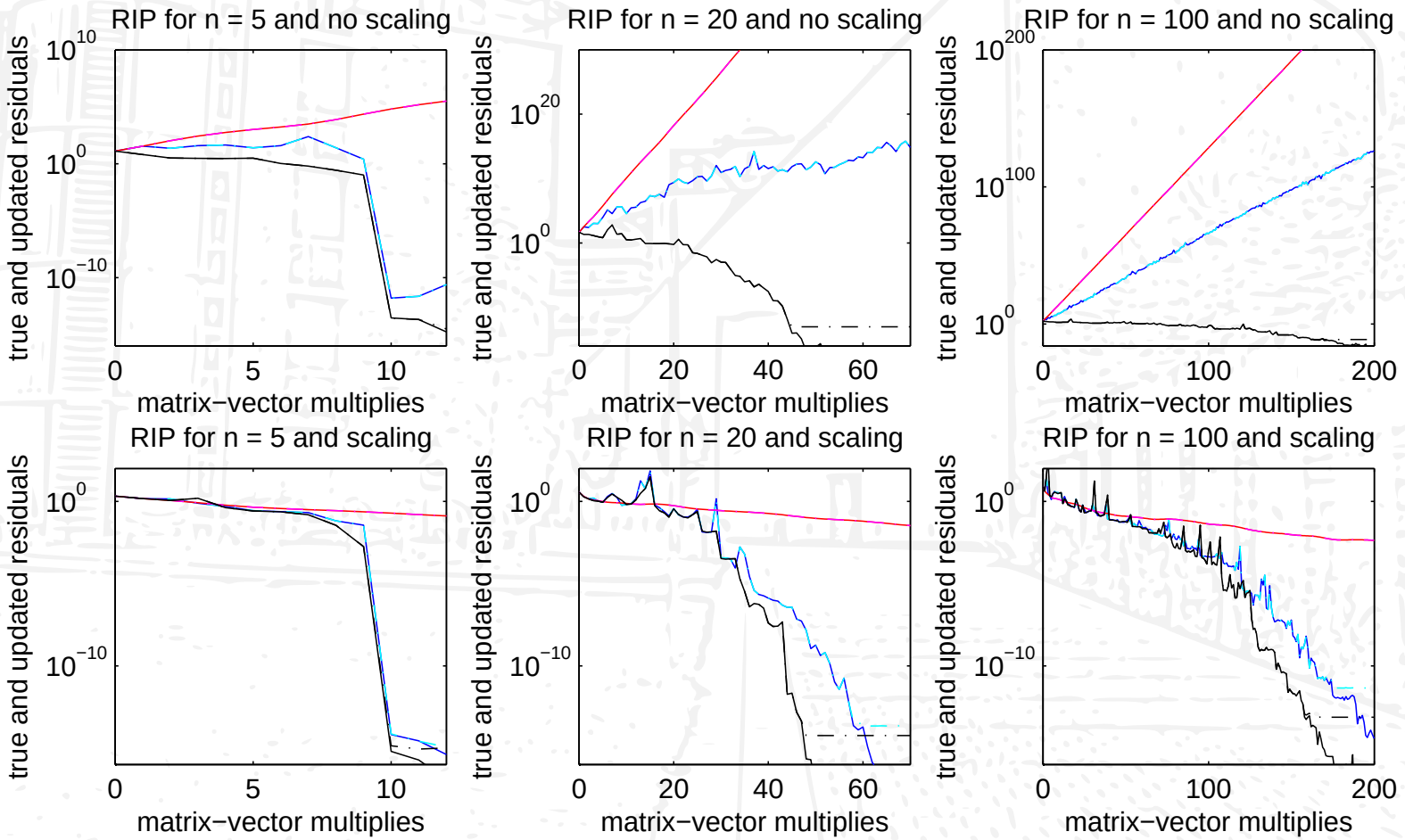
This is the original IDR Algorithm from page 551 of (Wesseling and Sonneveld, 1980).

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 philisophy 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.

Brothers of classical and primitive IDR

In 1976 Sonneveld considered the acceleration of Gauß-Seidel (AGS). Similarly, the IDR philisophy 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.

Brothers of classical and primitive IDR

In 1976 Sonneveld considered the acceleration of Gauß-Seidel (AGS). Similarly, the IDR philisophy 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.

Brothers of classical and primitive IDR

In 1976 Sonneveld considered the acceleration of Gauß-Seidel (AGS). Similarly, the IDR philisophy 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 . . .

Outline

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.

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 $\mathbf{P} \in \mathbb{C}^{n \times s}$, $1 \leqslant s \leqslant n$.

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.

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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

A few remarks:

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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

A few remarks:

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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

A few remarks:

We can start with any (simple) Krylov subspace method.

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

A few remarks:

We can start with any (simple) Krylov subspace method.

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

A few remarks:

We can start with any (simple) Krylov subspace method.

The 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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

A few remarks:

We can start with any (simple) Krylov subspace method.

The 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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  end for
  $j \leftarrow j+1$
end while

A few remarks:

We can start with any (simple) Krylov subspace method.

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

A few remarks:

We can start with any (simple) Krylov subspace method.

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

A few remarks:

We can start with any (simple) Krylov subspace method.

The 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 $\mathbf{R}_{s+1} = \mathbf{R}_{0:s} = (\mathbf{r}_0, \ldots, \mathbf{r}_s)$ using, e.g., ORTHORES
$\nabla\mathbf{R}_{1:s} = (\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_s) = (\mathbf{r}_1 - \mathbf{r}_0, \ldots, \mathbf{r}_s - \mathbf{r}_{s-1})$
$n \leftarrow s+1, j \leftarrow 1$
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 ω_j
  $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
  $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
  $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  for $k = 1, \ldots, s$
    $\mathbf{c}_n = (\mathbf{P}^{\mathsf{H}}\nabla\mathbf{R}_{n-s:n-1})^{-1}\mathbf{P}^{\mathsf{H}}\mathbf{r}_{n-1}$
    $\mathbf{v}_{n-1} = \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n$
    $\nabla\mathbf{r}_n = -\nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n - \omega_j\mathbf{A}\mathbf{v}_{n-1}$
    $\mathbf{r}_n = \mathbf{r}_{n-1} + \nabla\mathbf{r}_n, n \leftarrow n+1$
    $\nabla\mathbf{R}_{n-s:n-1} = (\nabla\mathbf{r}_{n-s}, \ldots, \nabla\mathbf{r}_{n-1})$
  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.

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

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

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.

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

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

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

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

Here, $\mathbf{H}_n$ denotes an unreduced Hessenberg matrix.

Understanding IDR: Hessenberg decompositions

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

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

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

$$\mathbf{A}\mathbf{Q}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T. \tag{9}$$

Understanding IDR: Hessenberg decompositions

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

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

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

$$\mathbf{A}\mathbf{Q}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T. \tag{9}$$

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

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T \tag{10}$$

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

IDR: Generalized Hessenberg decompositions

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

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T \tag{10}$$

and perturbed generalized Hessenberg decompositions

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T \tag{11}$$

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

IDR: Generalized Hessenberg decompositions

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

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T \tag{10}$$

and perturbed generalized Hessenberg decompositions

$$\mathbf{A}\mathbf{Q}_n\mathbf{U}_n + \mathbf{F}_n = \mathbf{Q}_{n+1}\underline{\mathbf{H}}_n = \mathbf{Q}_n\mathbf{H}_n + \mathbf{q}_{n+1}h_{n+1,n}\mathbf{e}_n^T \tag{11}$$

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,

$$\begin{aligned}\widehat{\mathbf{Q}}_n^H(\mathbf{A}, \mathbf{I})\mathbf{Q}_n\mathbf{U}_n &= \widehat{\mathbf{Q}}_n^H(\mathbf{A}\mathbf{Q}_n\mathbf{U}_n, \mathbf{Q}_n\mathbf{U}_n) \\ &= \widehat{\mathbf{Q}}_n^H(\mathbf{Q}_{n+1}\underline{\mathbf{H}}_n, \mathbf{Q}_n\mathbf{U}_n) = (\underline{\mathbf{I}}_n^T\underline{\mathbf{H}}_n, \mathbf{U}_n) = (\mathbf{H}_n, \mathbf{U}_n),\end{aligned} \tag{12}$$

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

Understanding IDR: OrthoRes-type methods

The entries of the Hessenberg matrices of these Hessenberg decompositions are defined in different variations.

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.

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.

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 Hessenberg decomposition

$$\mathbf{A}\mathbf{R}_n = \mathbf{R}_{n+1}\underline{\mathbf{H}}_n^{\circ} = \mathbf{R}_n\mathbf{H}_n^{\circ} + \mathbf{r}_{n+1}h_{n+1,n}^{\circ}\mathbf{e}_n^{\mathsf{T}}, \tag{13}$$

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

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 Hessenberg decomposition

$$\mathbf{A}\mathbf{R}_n = \mathbf{R}_{n+1}\underline{\mathbf{H}}_n^{\circ} = \mathbf{R}_n\mathbf{H}_n^{\circ} + \mathbf{r}_{n+1}h_{n+1,n}^{\circ}\mathbf{e}_n^{\mathsf{T}}, \tag{13}$$

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

$$\mathbf{R}_{n+1} = (\mathbf{r}_0,\ldots,\mathbf{r}_n) = \mathbf{Q}_{n+1}\,\mathrm{diag}\left(\frac{\|\mathbf{r}_0\|_2}{\|\mathbf{q}_1\|_2},\ldots,\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{q}_{n+1}\|_2}\right) \tag{14}$$

is diagonally scaled to be the matrix of residual vectors.

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

$$\mathbf{A}\mathbf{R}_n\mathbf{U}_n = \mathbf{R}_{n+1}\underline{\mathbf{H}}_n^{\circ} = \mathbf{R}_n\mathbf{H}_n^{\circ} + \mathbf{r}_{n+1}h_{n+1,n}^{\circ}\mathbf{e}_n^{\mathsf{T}}, \tag{13}$$

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

$$\mathbf{R}_{n+1} = (\mathbf{r}_0,\ldots,\mathbf{r}_n) = \mathbf{Q}_{n+1}\,\mathrm{diag}\left(\frac{\|\mathbf{r}_0\|_2}{\|\mathbf{q}_1\|_2},\ldots,\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{q}_{n+1}\|_2}\right) \tag{14}$$

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

$$\begin{aligned}
\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\
&= \left(1-\gamma_s^{(n)}\right)\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}\left(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)}\right)\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\
\mathbf{r}_n &:= \left(\mathbf{I}-\omega_j\mathbf{A}\right)\mathbf{v}_{n-1}\,.
\end{aligned} \tag{15}$$

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= (1-\gamma_s^{(n)})\,\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)})\,\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ \mathbf{r}_n &:= (\mathbf{I}-\omega_j\mathbf{A})\,\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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).

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= (1-\gamma_s^{(n)})\,\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)})\,\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ \mathbf{r}_n &:= (\mathbf{I}-\omega_j\mathbf{A})\,\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{16}$$

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= \left(1-\gamma_s^{(n)}\right)\mathbf{r}_{n-1} + \sum_{\ell=1}^{s-1}\left(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)}\right)\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ \mathbf{r}_n &:= \left(\mathbf{I}-\omega_j\mathbf{A}\right)\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{16}$$

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= (1-\gamma_s^{(n)})\,\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)})\,\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ \mathbf{r}_n &:= (\mathbf{I} - \omega_j\mathbf{A})\,\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{16}$$

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= (1-\gamma_s^{(n)})\,\mathbf{r}_{n-1} + \textstyle\sum_{\ell=1}^{s-1}(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)})\,\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ 1\cdot\mathbf{r}_n &:= (\mathbf{I}-\omega_j\mathbf{A})\,\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{16}$$

IDR: The underlying Hessenberg decomposition

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

$$\begin{aligned}\mathbf{v}_{n-1} &:= \mathbf{r}_{n-1} - \nabla\mathbf{R}_{n-s:n-1}\mathbf{c}_n = \mathbf{R}_{n-s-1:n-1}\mathbf{y}_n \\ &= \left(1 - \gamma_s^{(n)}\right)\mathbf{r}_{n-1} + \sum_{\ell=1}^{s-1}\left(\gamma_{s-\ell+1}^{(n)} - \gamma_{s-\ell}^{(n)}\right)\mathbf{r}_{n-\ell-1} + \gamma_1^{(n)}\,\mathbf{r}_{n-s-1}\,, \\ 1\cdot\mathbf{r}_n &:= \left(\mathbf{I} - \omega_j\mathbf{A}\right)\mathbf{v}_{n-1}\,.\end{aligned} \tag{15}$$

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

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

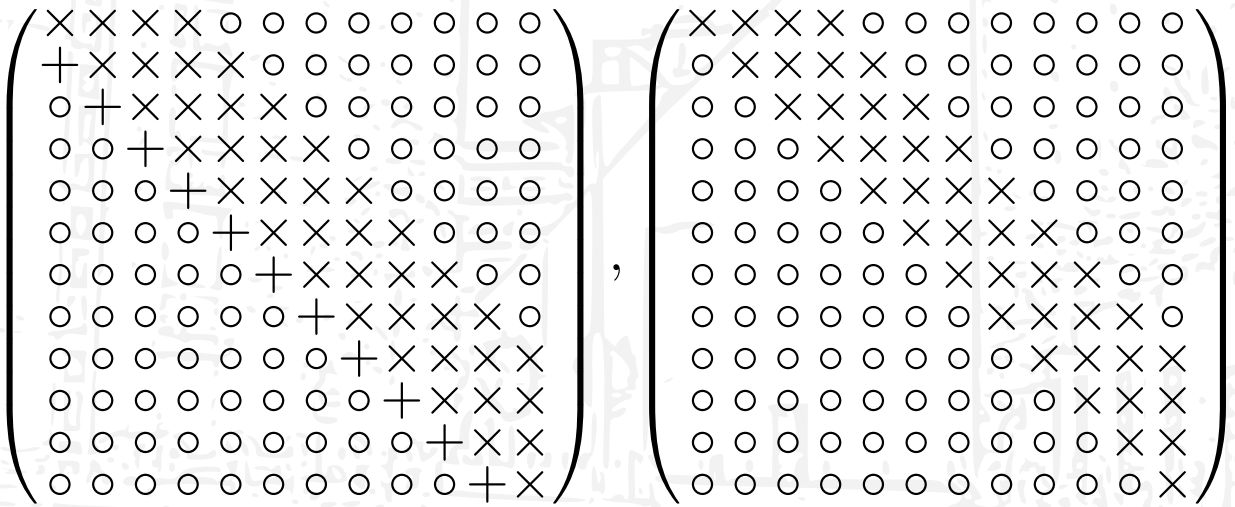
compare with the so-called "index functions" (Yeung/Boley, 2005).

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

$$\mathbf{A}\mathbf{R}_n\mathbf{Y}_n\mathbf{D}_\omega = \mathbf{R}_{n+1}\underline{\mathbf{Y}}_n^\circ. \tag{16}$$

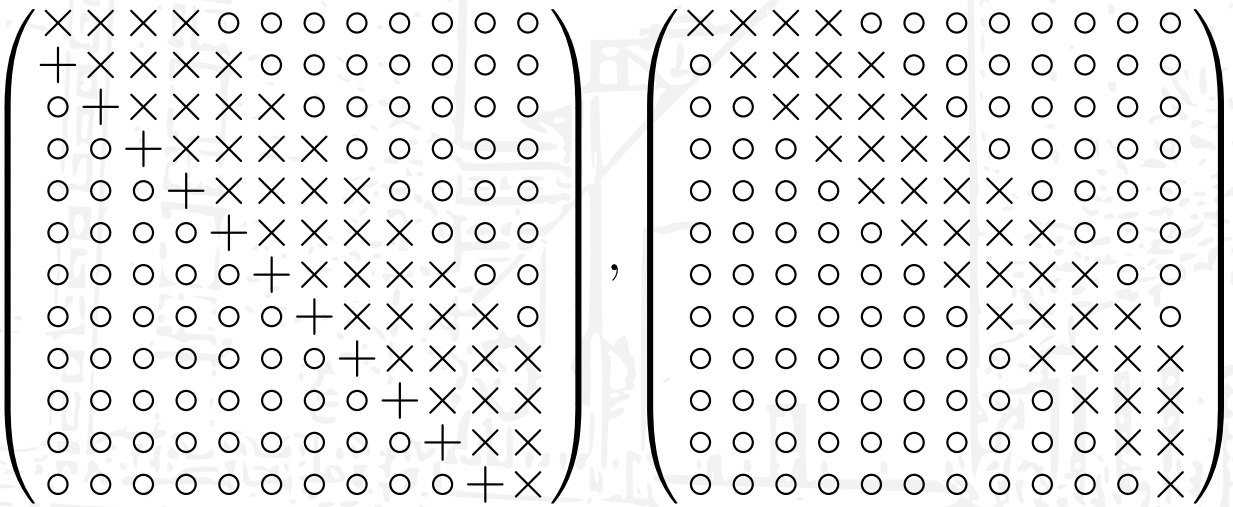
IDR: Sonneveld pencil and Sonneveld matrix

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



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



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.

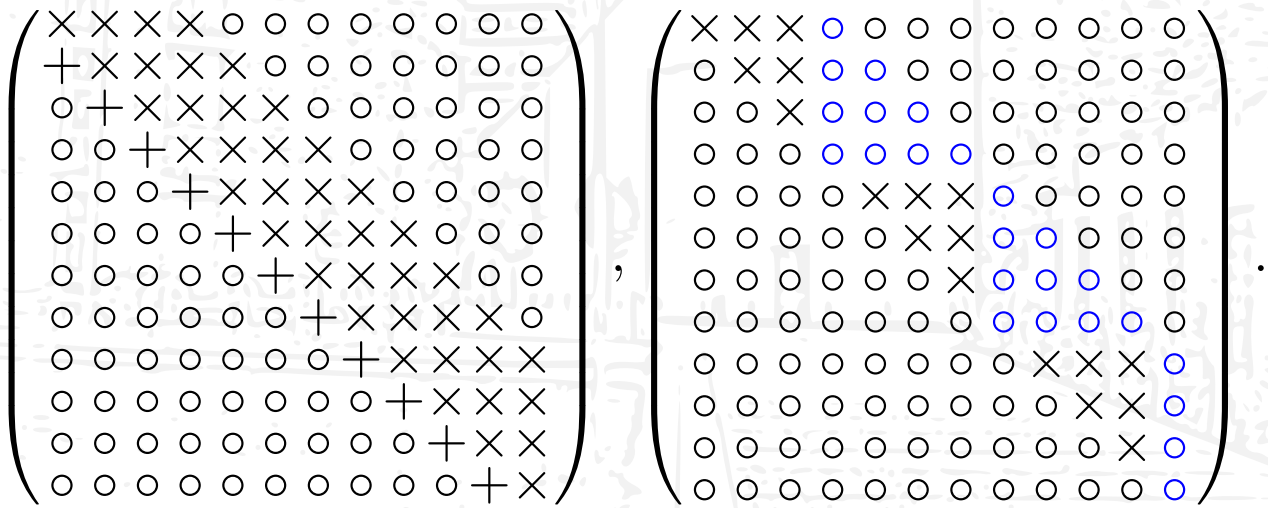
Understanding IDR: Purification

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

Understanding IDR: Purification

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

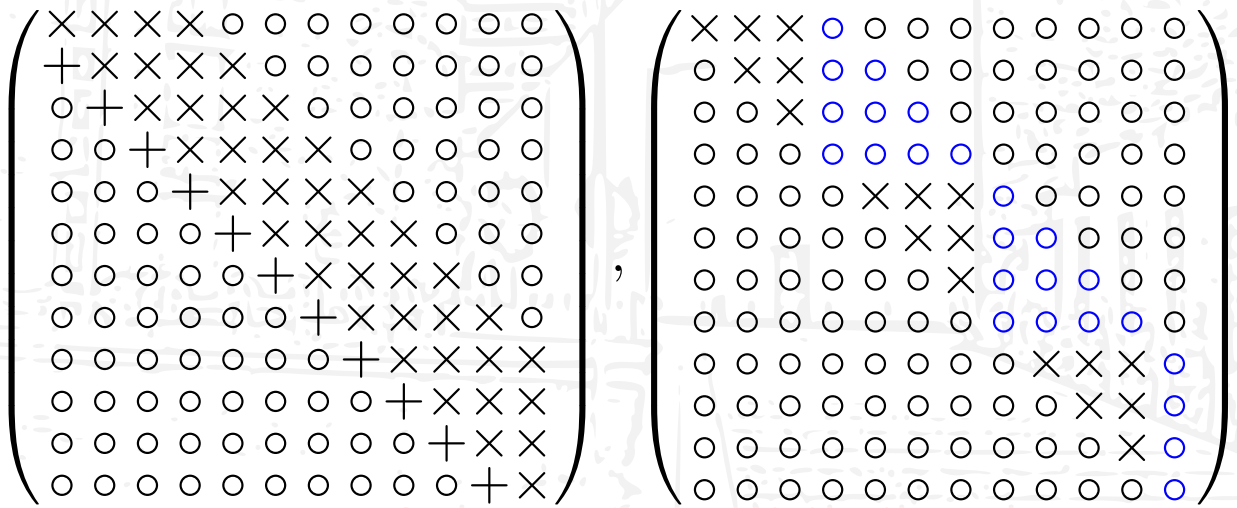
The purified IDR(s)ORes pencil $(\mathbf{Y}_n^\circ, \mathbf{U}_n\mathbf{D}_\omega^{(n)})$, that has only the remaining eigenvalues and some infinite ones as eigenvalues, can be depicted by



Understanding IDR: Purification

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

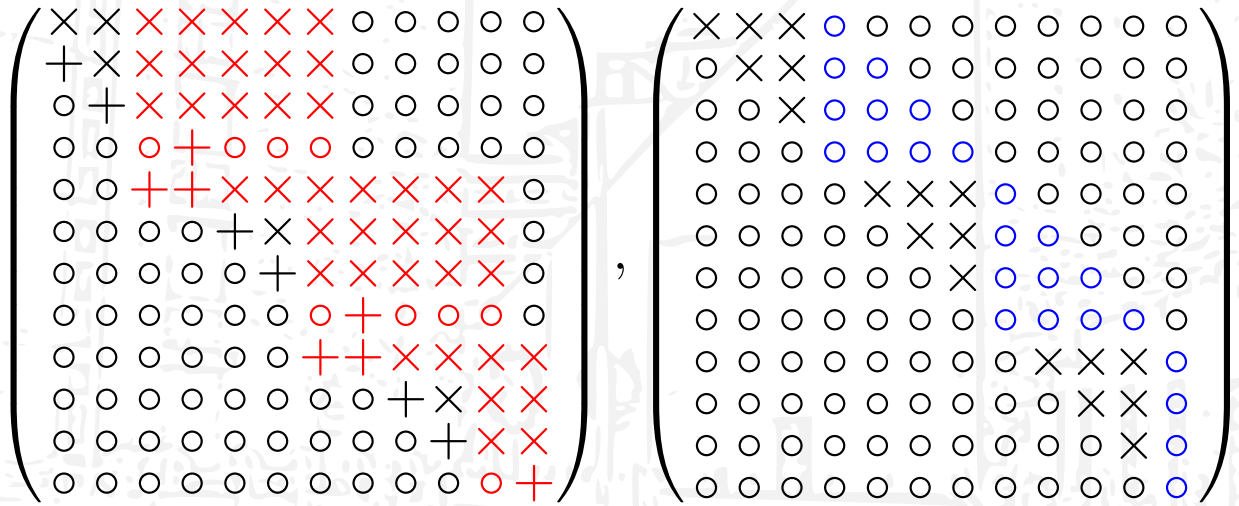
The purified IDR(s)ORes pencil $(\mathbf{Y}_n^\circ, \mathbf{U}_n\mathbf{D}_\omega^{(n)})$, that has only the remaining eigenvalues and some infinite ones as eigenvalues, can be depicted by



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

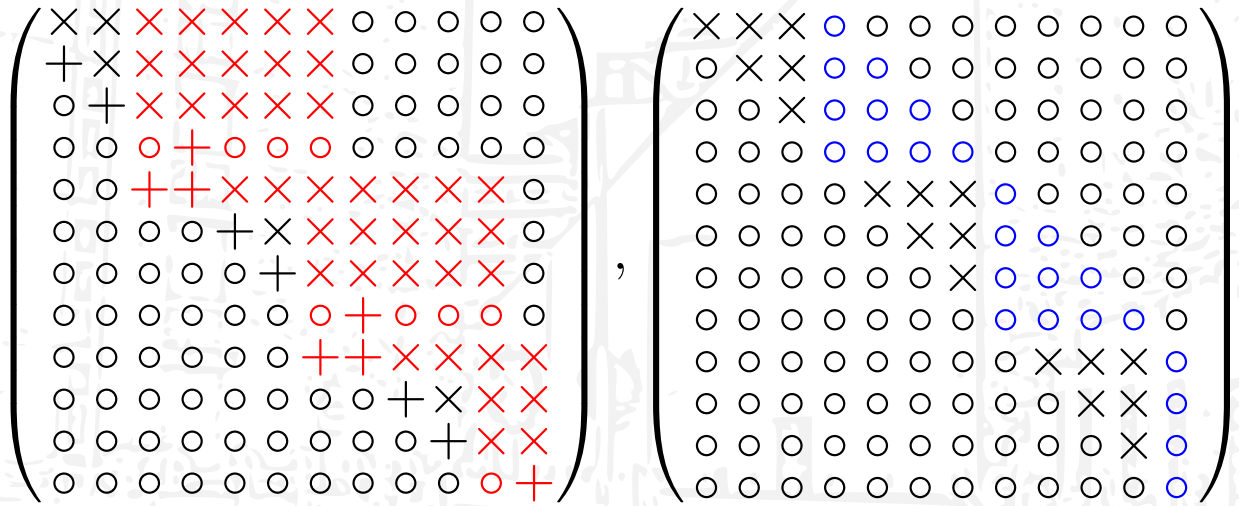
Understanding IDR: Gaussian elimination

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



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



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.

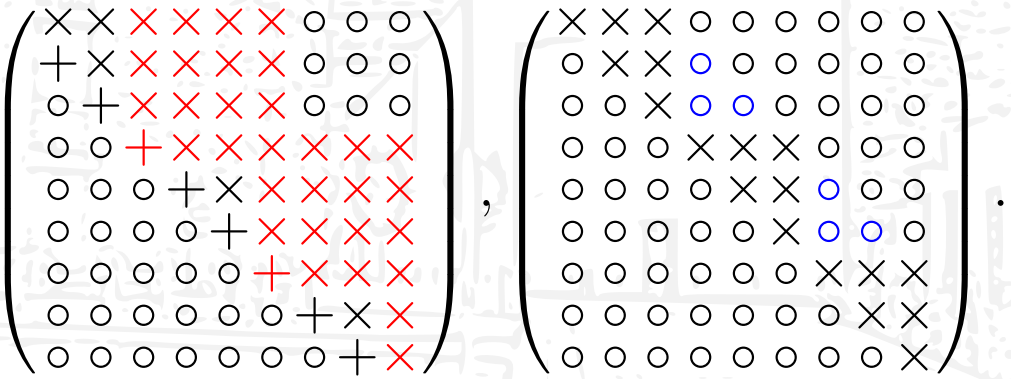
Understanding IDR: Deflation

Let D denote an deflation operator that removes every $s+1$th column and row from the matrix the operator is applied to.

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.

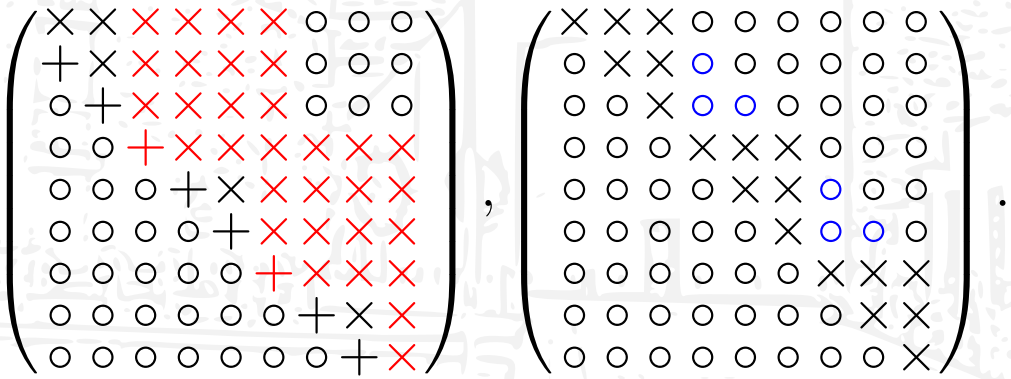
The deflated purified IDR(s)ORes pencil, after the deflation step $(D(\mathbf{Y}_n^{\circ}\mathbf{G}_n), D(\mathbf{U}_n\mathbf{D}_{\omega}^{(n)}))$, can be depicted by



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



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

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

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.

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 $\mathbf{P}$.

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

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

Using induction (Sleijpen et al., 2008) 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.

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 ($\mathcal{S} := \mathbf{P}^{\perp}$)

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

$$\mathcal{G}_j = \bigcap_{k=0}^{j-1} \Omega_k(\mathbf{A})^{-1}\Omega_j(\mathbf{A})(\mathcal{S}) = \Big(\mathop{+}_{k=0}^{j-1} \Omega_j(\mathbf{A})^{-\mathsf{H}}\Omega_k(\mathbf{A})^{\mathsf{H}}\{\mathbf{P}\}\Big)^{\perp}$$

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

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).

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

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

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

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

IDR: a Lanczos process with multiple left-hand sides

The second equality

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

is based on

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

and

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

IDR: a Lanczos process with multiple left-hand sides

The second equality

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

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

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

$$= \Omega_j(\mathbf{A}) \left(\mathcal{K}_j(\mathbf{A}^{\mathsf{H}}, \mathbf{P}) \right)^{\perp}$$

IDR: a Lanczos process with multiple left-hand sides

The third and fourth equality

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

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.

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.

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.

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.

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).

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).

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).

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 $\mathbf{A}$.

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.

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.

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.

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.

Outline

Similarities between Lanczos and IDR(1)

We have seen that IDR(s) is a Lanczos-type product method (LTPM), where the underlying Lanczos process uses s left and one right starting vectors.

Similarities between Lanczos and IDR(1)

We have seen that IDR(s) is a Lanczos-type product method (LTPM), where the underlying Lanczos process uses s left and one right starting vectors.

Furthermore, IDR(1) is slightly different from the classical IDR, but they produce at every second step the same residuals, and so does BiCGStab. We come back to this point in a minute.

Similarities between Lanczos and IDR(1)

We have seen that IDR(s) is a Lanczos-type product method (LTPM), where the underlying Lanczos process uses s left and one right starting vectors.

Furthermore, IDR(1) is slightly different from the classical IDR, but they produce at every second step the same residuals, and so does BiCGStab. We come back to this point in a minute.

In case of $s = 1$ we have to deal with a perturbed simple Lanczos process, i.e., the Lanczos method. The Lanczos method is known to produce multiple copies of outliers, i.e., many Ritz values converge to just one single prominent eigenvalue of $\mathbf{A}$.

Similarities between Lanczos and IDR(1)

We have seen that IDR(s) is a Lanczos-type product method (LTPM), where the underlying Lanczos process uses s left and one right starting vectors.

Furthermore, IDR(1) is slightly different from the classical IDR, but they produce at every second step the same residuals, and so does BiCGStab. We come back to this point in a minute.

In case of $s = 1$ we have to deal with a perturbed simple Lanczos process, i.e., the Lanczos method. The Lanczos method is known to produce multiple copies of outliers, i.e., many Ritz values converge to just one single prominent eigenvalue of $\mathbf{A}$.

The only difference between the Lanczos process underlying IDR(1) and the finite precision Lanczos method is the perturbation term. Yet we will see that the structural constraints result in totally different deviations from the theoretical behavior of the Lanczos method.

Chris Paige, Anne Greenbaum, the Lanczos process

Following his seminal PhD thesis (Paige, 1971), Chris Paige published a sequence of papers (Paige, 1972; Paige, 1976; Paige, 1980) on the error analysis of the finite-precision behavior of the symmetric Lanczos process.

Chris Paige, Anne Greenbaum, the Lanczos process

Following his seminal PhD thesis (Paige, 1971), Chris Paige published a sequence of papers (Paige, 1972; Paige, 1976; Paige, 1980) on the error analysis of the finite-precision behavior of the symmetric Lanczos process.

His results form the basis of Anne Greenbaum's celebrated "backward error analysis" (Greenbaum, 1989) of the finite-precision symmetric Lanczos and CG methods, compare with (Greenbaum and Strakoš, 1992).

Chris Paige, Anne Greenbaum, the Lanczos process

Following his seminal PhD thesis (Paige, 1971), Chris Paige published a sequence of papers (Paige, 1972; Paige, 1976; Paige, 1980) on the error analysis of the finite-precision behavior of the symmetric Lanczos process.

His results form the basis of Anne Greenbaum's celebrated "backward error analysis" (Greenbaum, 1989) of the finite-precision symmetric Lanczos and CG methods, compare with (Greenbaum and Strakoš, 1992).

For an introduction and a general exposition especially on the finite-precision symmetric Lanczos and CG methods see also (Meurant, 2006; Meurant and Strakoš, 2006).

Chris Paige, Anne Greenbaum, the Lanczos process

Following his seminal PhD thesis (Paige, 1971), Chris Paige published a sequence of papers (Paige, 1972; Paige, 1976; Paige, 1980) on the error analysis of the finite-precision behavior of the symmetric Lanczos process.

His results form the basis of Anne Greenbaum's celebrated "backward error analysis" (Greenbaum, 1989) of the finite-precision symmetric Lanczos and CG methods, compare with (Greenbaum and Strakoš, 1992).

For an introduction and a general exposition especially on the finite-precision symmetric Lanczos and CG methods see also (Meurant, 2006; Meurant and Strakoš, 2006).

Thus far, this is maybe the only successful error analysis ever carried out for a perturbed short-term Krylov subspace method.

Lanczos' method in finite precision

We used the diagonal matrix

$$\mathbf{A} = \texttt{diag([linspace(0,1,50),3])}$$

and the starting vector

$$\mathbf{e} = \texttt{ones(51,1)}$$

in an implementation of Lanczos' method in MATLAB on a PC conforming to ANSI/IEEE 754 with machine precision $\texttt{eps(1)} = 2^{-52} \approx 2.2204 \cdot 10^{-16}$.

Lanczos' method in finite precision

We used the diagonal matrix

$$\mathbf{A} = \texttt{diag([linspace(0,1,50),3])}$$

and the starting vector

$$\mathbf{e} = \texttt{ones(51,1)}$$

in an implementation of Lanczos' method in MATLAB on a PC conforming to ANSI/IEEE 754 with machine precision $\texttt{eps(1)} = 2^{-52} \approx 2.2204 \cdot 10^{-16}$.

At step 10 the first Ritz value has converged (up to machine precision) to the eigenvalue 3, at step 27 the second one has converged. Detoriation reaches a maximum at step $19 = \lceil (10 + 27)/2 \rceil$.

Lanczos' method in finite precision

We used the diagonal matrix

$$\mathbf{A} = \texttt{diag([linspace(0,1,50),3])}$$

and the starting vector

$$\mathbf{e} = \texttt{ones(51,1)}$$

in an implementation of Lanczos' method in MATLAB on a PC conforming to ANSI/IEEE 754 with machine precision $\texttt{eps(1)} = 2^{-52} \approx 2.2204 \cdot 10^{-16}$.

At step 10 the first Ritz value has converged (up to machine precision) to the eigenvalue 3, at step 27 the second one has converged. Detoriation reaches a maximum at step $19 = \lceil (10 + 27)/2 \rceil$.

Eigenvalues and eigenvectors are computed using MRRR, i.e., LAPACK's routine `DSTEGR`, since MATLAB's `eig` (using LAPACK's `DSYEV`, i.e., the QR algorithm implemented as `DSTEQR`) fails in delivering accurate eigenvectors.

Lanczos' method in finite precision

We used the diagonal matrix

$$\mathbf{A} = \texttt{diag([linspace(0,1,50),3])}$$

and the starting vector

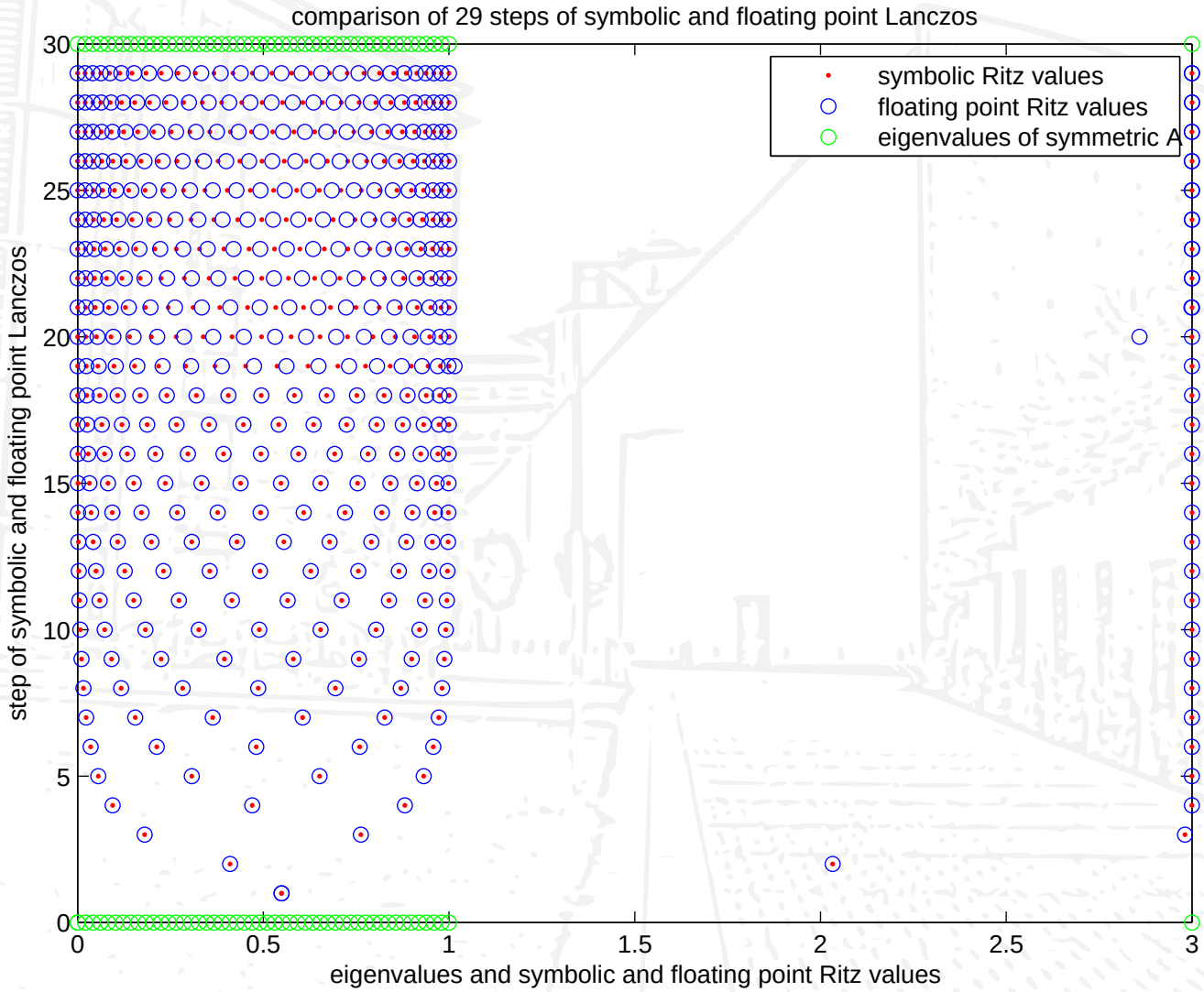
$$\mathbf{e} = \texttt{ones(51,1)}$$

in an implementation of Lanczos' method in MATLAB on a PC conforming to ANSI/IEEE 754 with machine precision $\texttt{eps(1)} = 2^{-52} \approx 2.2204 \cdot 10^{-16}$.

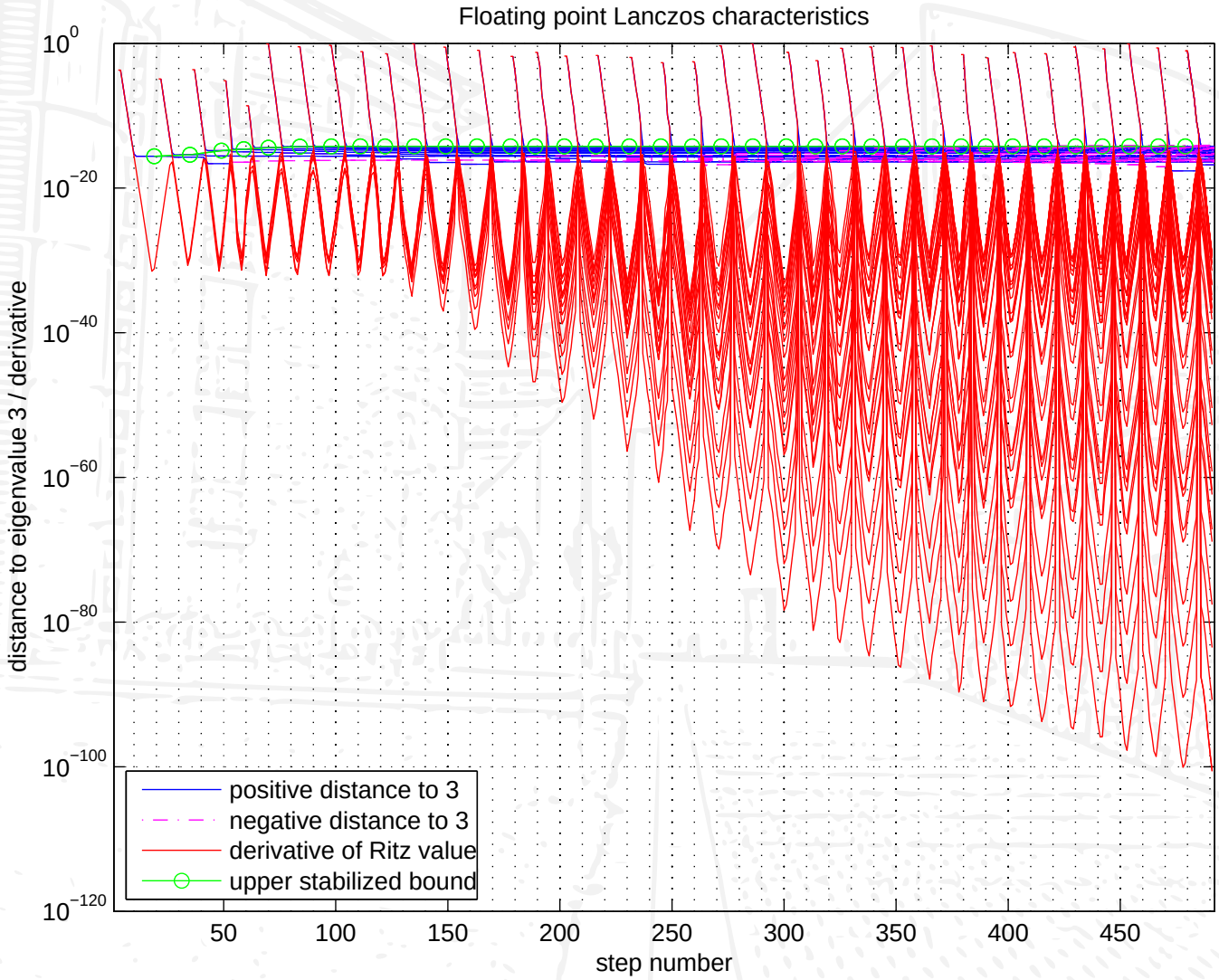
At step 10 the first Ritz value has converged (up to machine precision) to the eigenvalue 3, at step 27 the second one has converged. Detoriation reaches a maximum at step $19 = \lceil (10 + 27)/2 \rceil$.

Eigenvalues and eigenvectors are computed using MRRR, i.e., LAPACK's routine `DSTEGR`, since MATLAB's `eig` (using LAPACK's `DSYEV`, i.e., the QR algorithm implemented as `DSTEQR`) fails in delivering accurate eigenvectors. Additionally, we heavily used the symbolic toolbox, i.e., MAPLE.

Lanczos' method in finite precision



Lanczos' method in finite precision



Lanczos' method in finite precision

The theory of the Lanczos method in case of non-selfadjoint matrices is still less satisfactory. Some of the conclusions carry over, and the behavior in finite precision shows some similarities.

.

Lanczos' method in finite precision

The theory of the Lanczos method in case of non-selfadjoint matrices is still less satisfactory. Some of the conclusions carry over, and the behavior in finite precision shows some similarities.

The next example uses the matrix `pores_2` of size 1224×1224 from Matrix Market. The left and right starting vectors have been chosen such that all components are equal.

.

Lanczos' method in finite precision

The theory of the Lanczos method in case of non-selfadjoint matrices is still less satisfactory. Some of the conclusions carry over, and the behavior in finite precision shows some similarities.

The next example uses the matrix `pores_2` of size 1224×1224 from Matrix Market. The left and right starting vectors have been chosen such that all components are equal.

As there does not exist the best Lanczos method, we have chosen one of the more stable ones, namely the variant described in (Bai, 1994).

Lanczos' method in finite precision

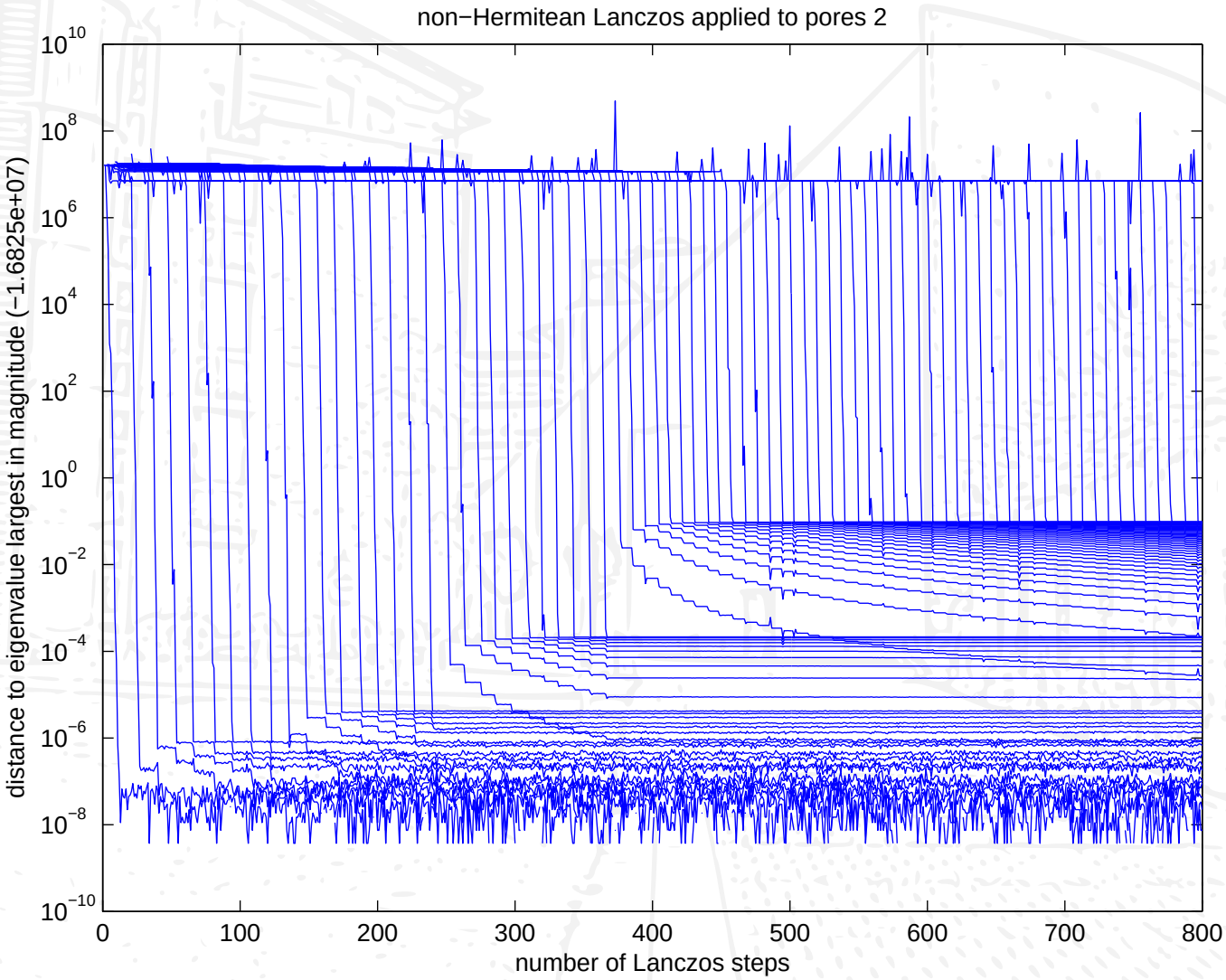
The theory of the Lanczos method in case of non-selfadjoint matrices is still less satisfactory. Some of the conclusions carry over, and the behavior in finite precision shows some similarities.

The next example uses the matrix `pores_2` of size 1224×1224 from Matrix Market. The left and right starting vectors have been chosen such that all components are equal.

As there does not exist the best Lanczos method, we have chosen one of the more stable ones, namely the variant described in (Bai, 1994).

We note that we can observe multiple copies, but this time the approximation quality is reduced after a couple of steps, all Ritz values computed after certain steps show worse behavior than before.

Lanczos' method in finite precision



Outline

IDR, IDR(1), and BiCGStab

We already noted that IDR, IDR(1), and BiCGStab are in some sense equivalent: They produce every second step the same residual. These steps are related to the Lanczos method and the minimizers ω_j.

IDR, IDR(1), and BiCGStab

We already noted that IDR, IDR(1), and BiCGStab are in some sense equivalent: They produce every second step the same residual. These steps are related to the Lanczos method and the minimizers ω_j.

If all sets of ω_j, those of IDR, IDR(1), and those of BiCGStab, are chosen by line minimization, the methods behave the same in exact arithmetic.

IDR, IDR(1), and BiCGStab

We already noted that IDR, IDR(1), and BiCGStab are in some sense equivalent: They produce every second step the same residual. These steps are related to the Lanczos method and the minimizers ω_j.

If all sets of ω_j, those of IDR, IDR(1), and those of BiCGStab, are chosen by line minimization, the methods behave the same in exact arithmetic.

We show numerically that the Lanczos-part of finite precision BiCGStab behaves by no means similar to the Lanczos method in finite precision.

IDR, IDR(1), and BiCGStab

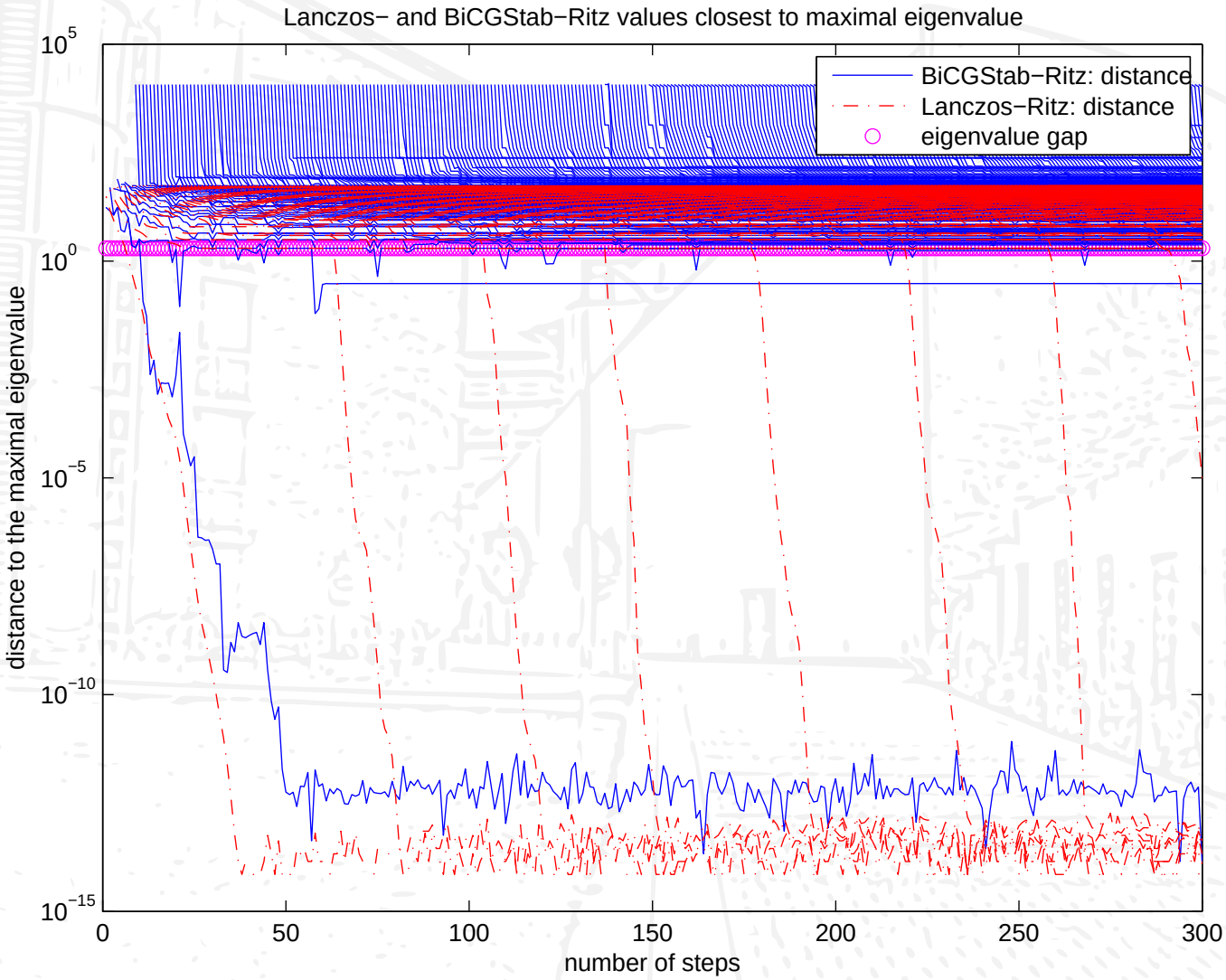
We already noted that IDR, IDR(1), and BiCGStab are in some sense equivalent: They produce every second step the same residual. These steps are related to the Lanczos method and the minimizers ω_j.

If all sets of ω_j, those of IDR, IDR(1), and those of BiCGStab, are chosen by line minimization, the methods behave the same in exact arithmetic.

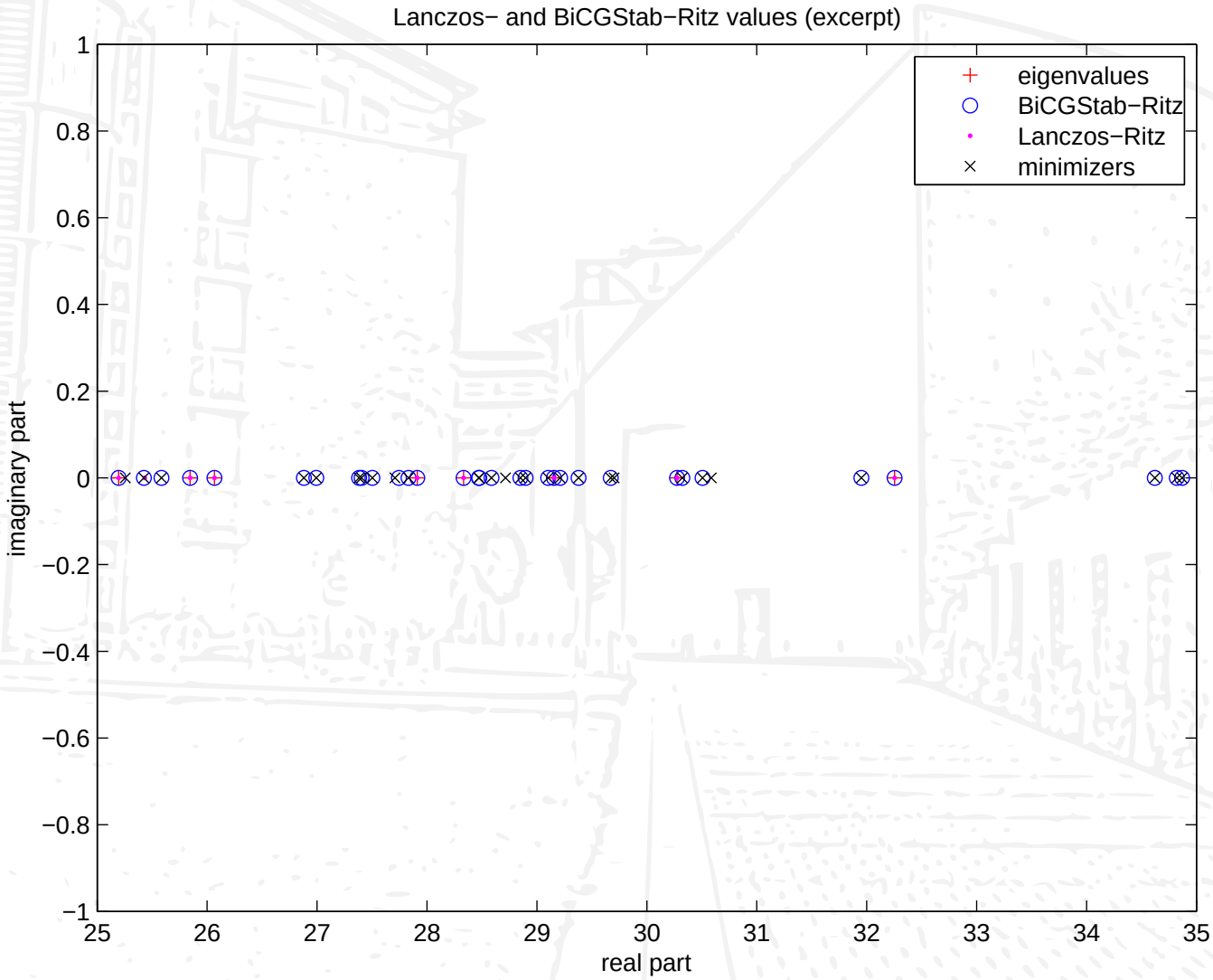
We show numerically that the Lanczos-part of finite precision BiCGStab behaves by no means similar to the Lanczos method in finite precision.

The eigenvalue approximations corresponding to BiCGStab are easier to analyze than those corresponding to IDR or IDR(1), since in BiCGStab the coefficients of the tridiagonal matrix of the underlying Lanczos method are explicitely computed.

IDR, IDR(1), and BiCGStab



IDR, IDR(1), and BiCGStab



Outline

General setting

We used the stablest variant of IDR(s)Eig to compute the Ritz values of IDR(s)ORes. In all experiments presented, $s = 2$.

General setting

We used the stablest variant of IDR(s)Eig to compute the Ritz values of IDR(s)ORes. In all experiments presented, $s = 2$.

The matrices have been constructed to have a small condition number and such that all eigenvalues are well-conditioned, but these real matrices have imaginary eigenvalues. The matrices are shifted to be positive real or, at least, such that zero is in the left part of the field of values.

General setting

We used the stablest variant of IDR(s)Eig to compute the Ritz values of IDR(s)ORes. In all experiments presented, $s = 2$.

The matrices have been constructed to have a small condition number and such that all eigenvalues are well-conditioned, but these real matrices have imaginary eigenvalues. The matrices are shifted to be positive real or, at least, such that zero is in the left part of the field of values.

We try to find multiple copies. From the analysis of IDR(s)ORes we know that after the ultimately attainable accuracy has been reached, the eigenvalue approximations could deteriorate. So we investigate far beyond this point.

General setting

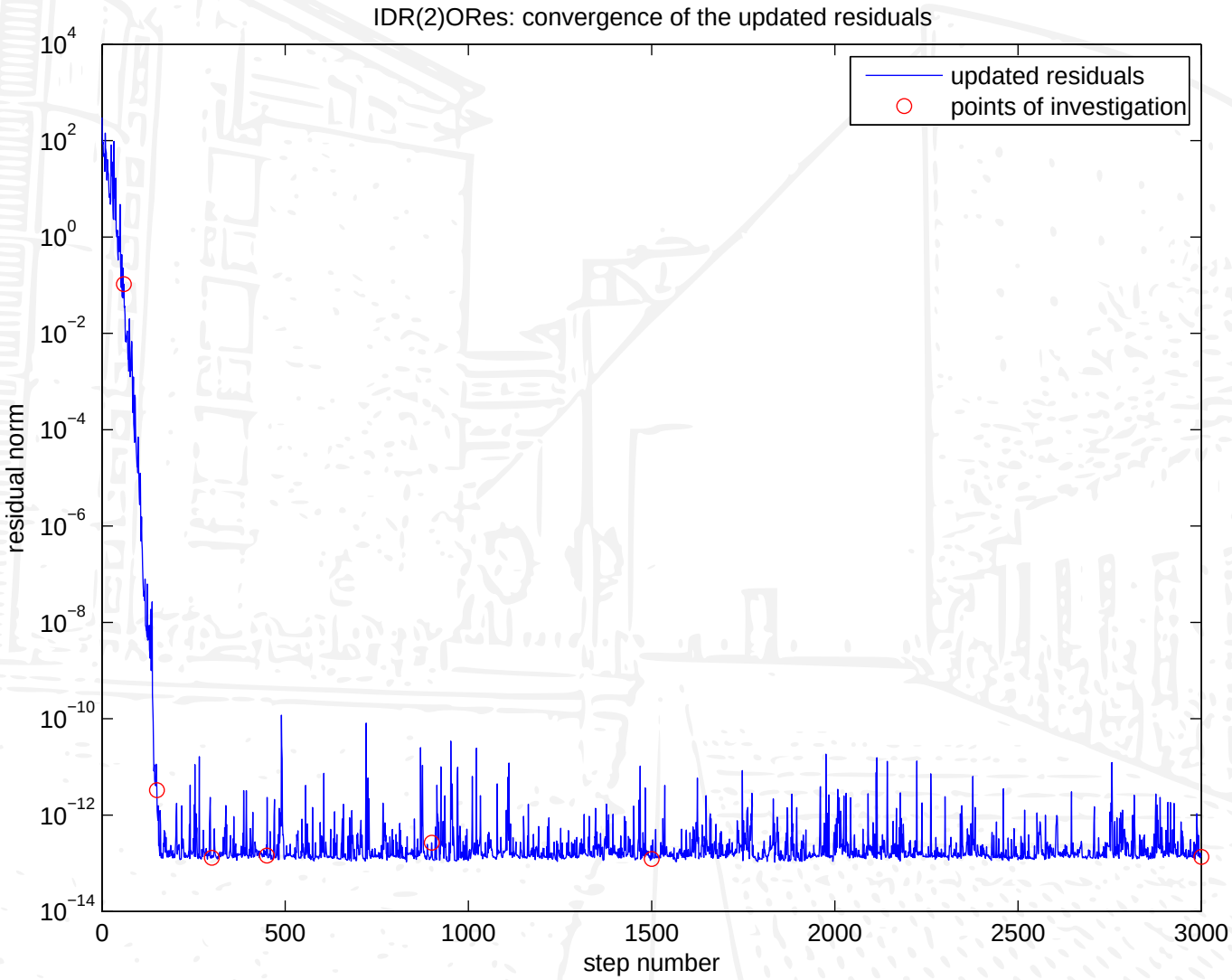
We used the stablest variant of IDR(s)Eig to compute the Ritz values of IDR(s)ORes. In all experiments presented, $s = 2$.

The matrices have been constructed to have a small condition number and such that all eigenvalues are well-conditioned, but these real matrices have imaginary eigenvalues. The matrices are shifted to be positive real or, at least, such that zero is in the left part of the field of values.

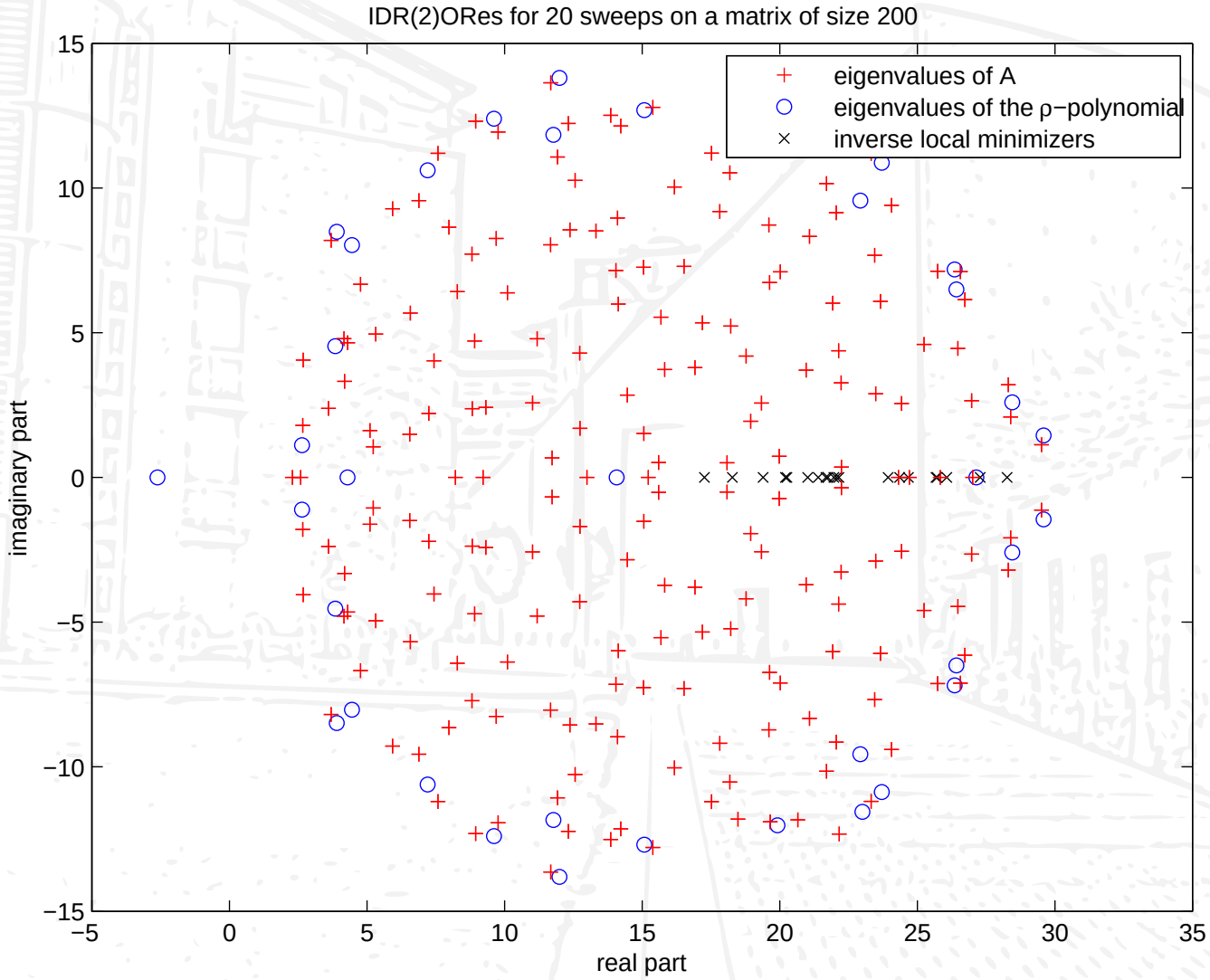
We try to find multiple copies. From the analysis of IDR(s)ORes we know that after the ultimately attainable accuracy has been reached, the eigenvalue approximations could deteriorate. So we investigate far beyond this point.

Since for real data all minimizers are real, we have run two sets of experiments with the same matrix and starting residual: The first experiment is based on an orthonormal random real matrix $\mathbf{P} \in \mathbb{R}^{n\times 2}$, the second experiment is based on an orthonormal random complex matrix $\mathbf{P} \in \mathbb{C}^{n\times 2}$.

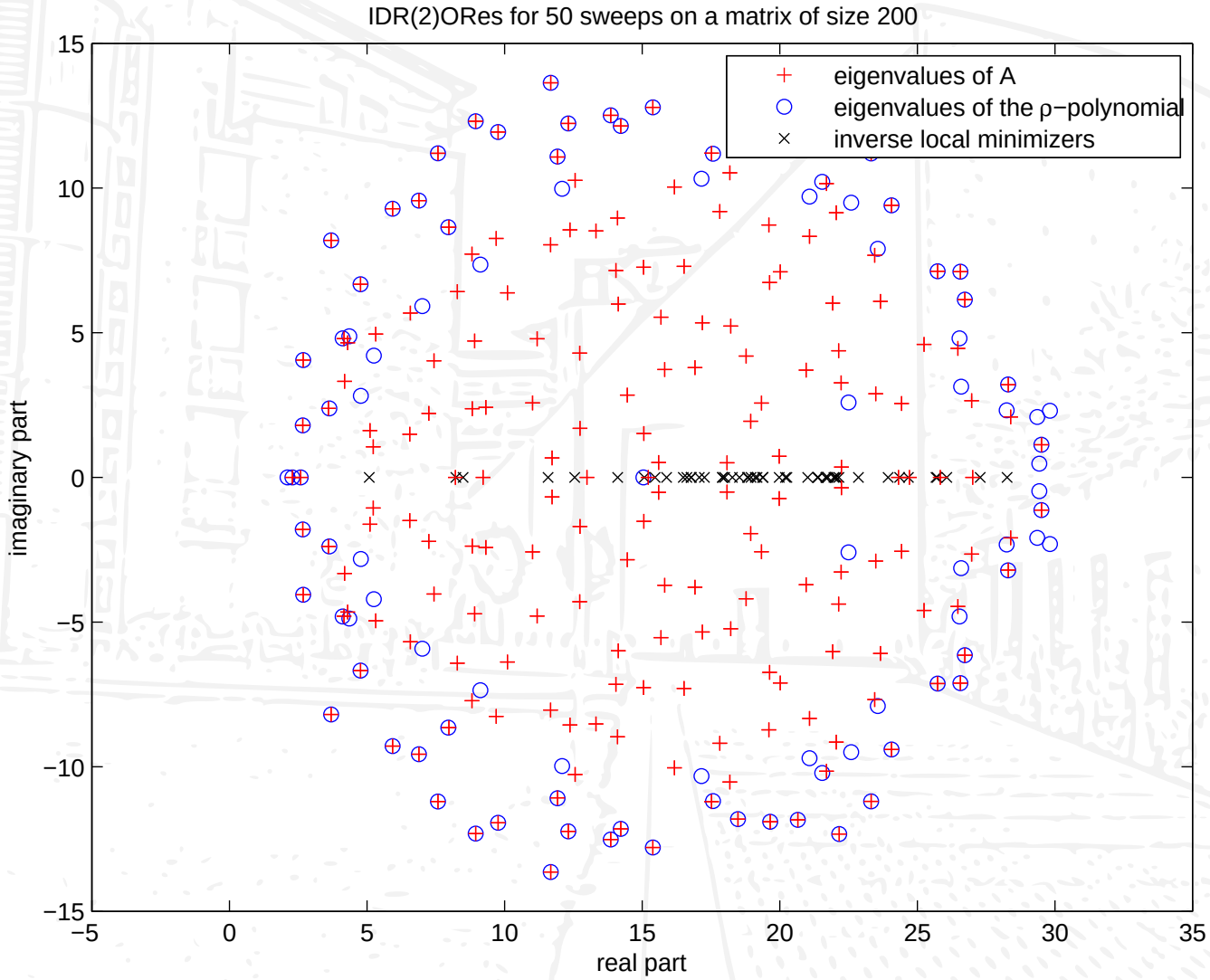
Understanding IDR: Convergence for $s = 2$; $\mathbf{P}$ real



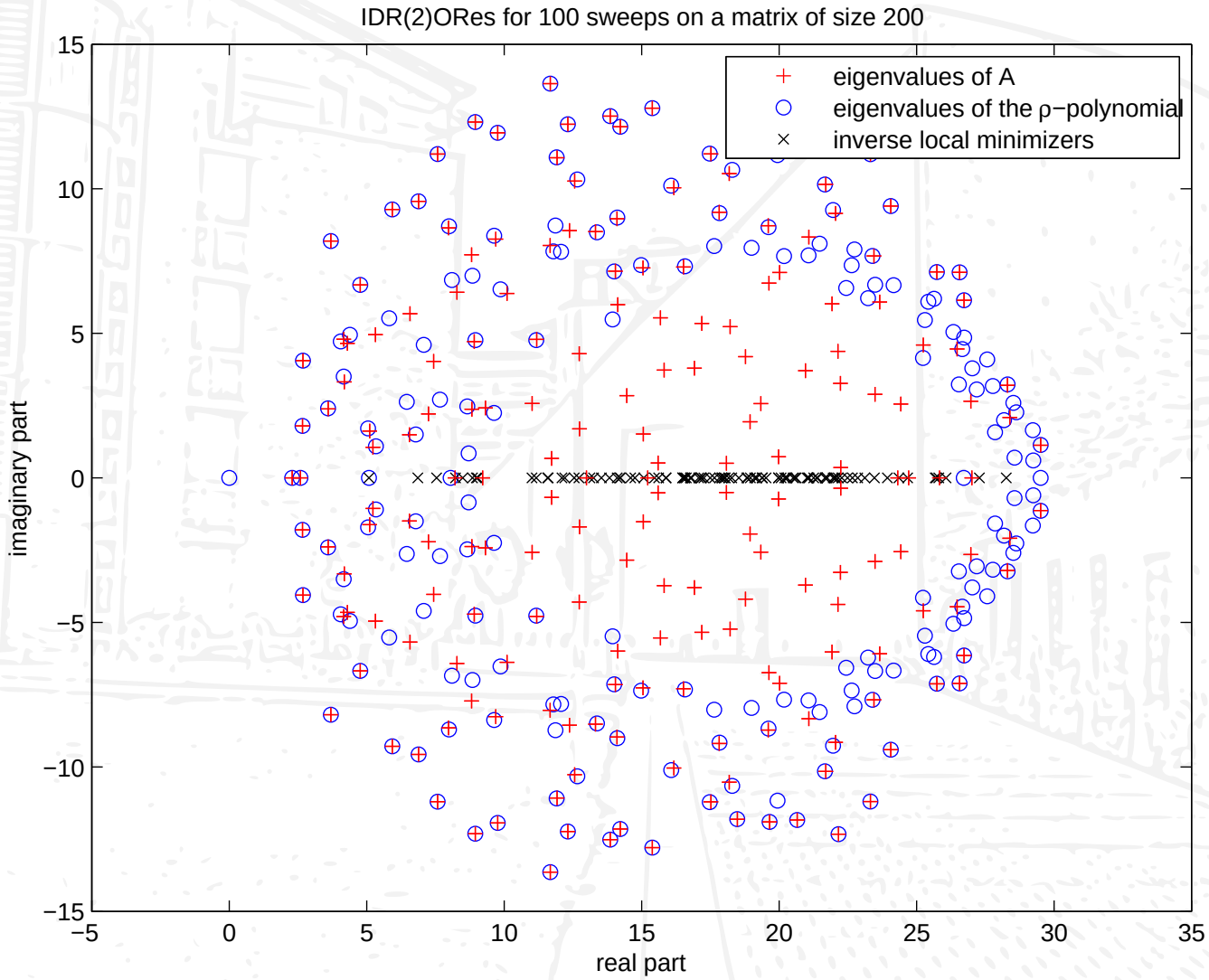
Understanding IDR: 20 steps for $s = 2$; $\mathbf{P}$ real



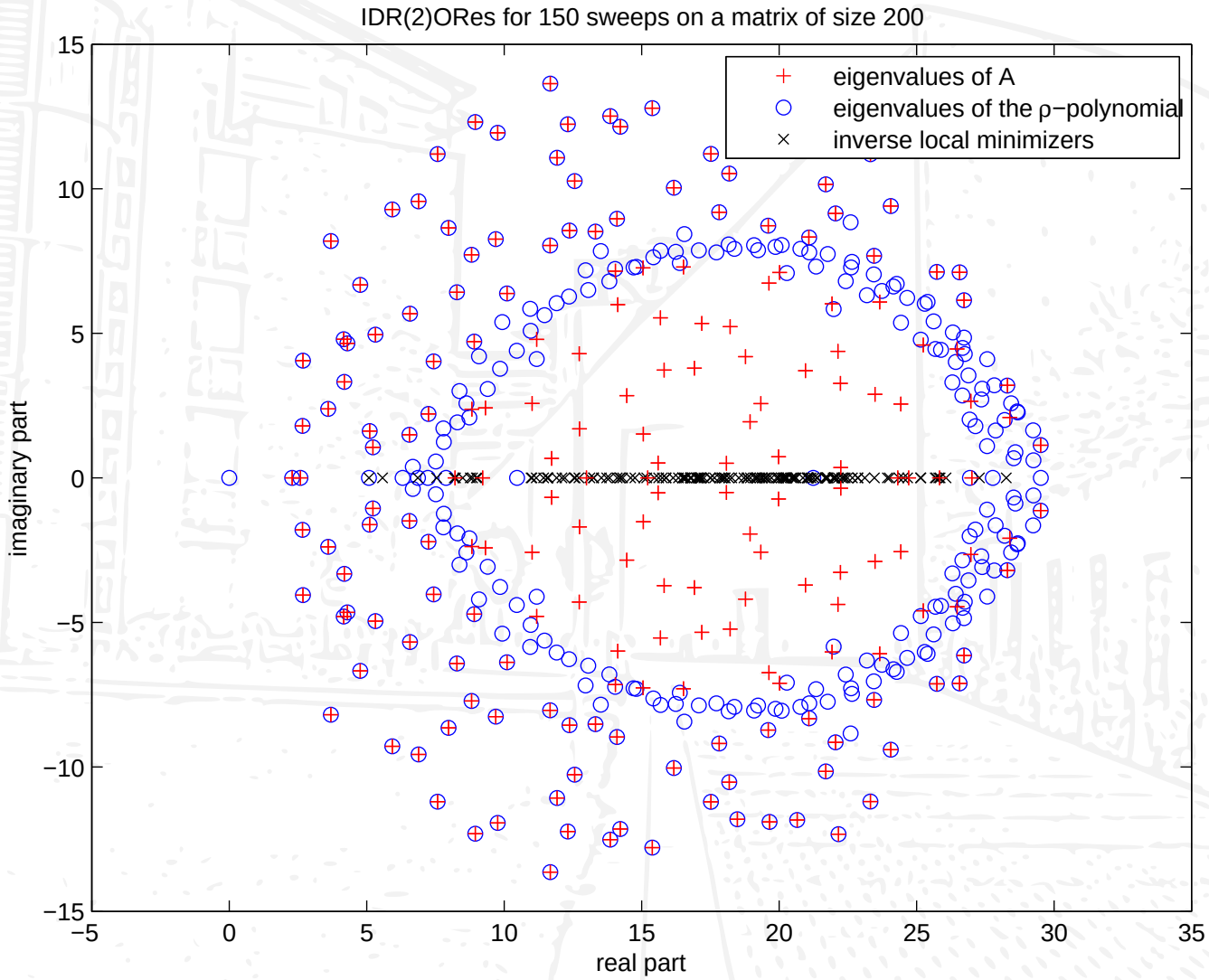
Understanding IDR: 50 steps for $s = 2$; $\mathbf{P}$ real



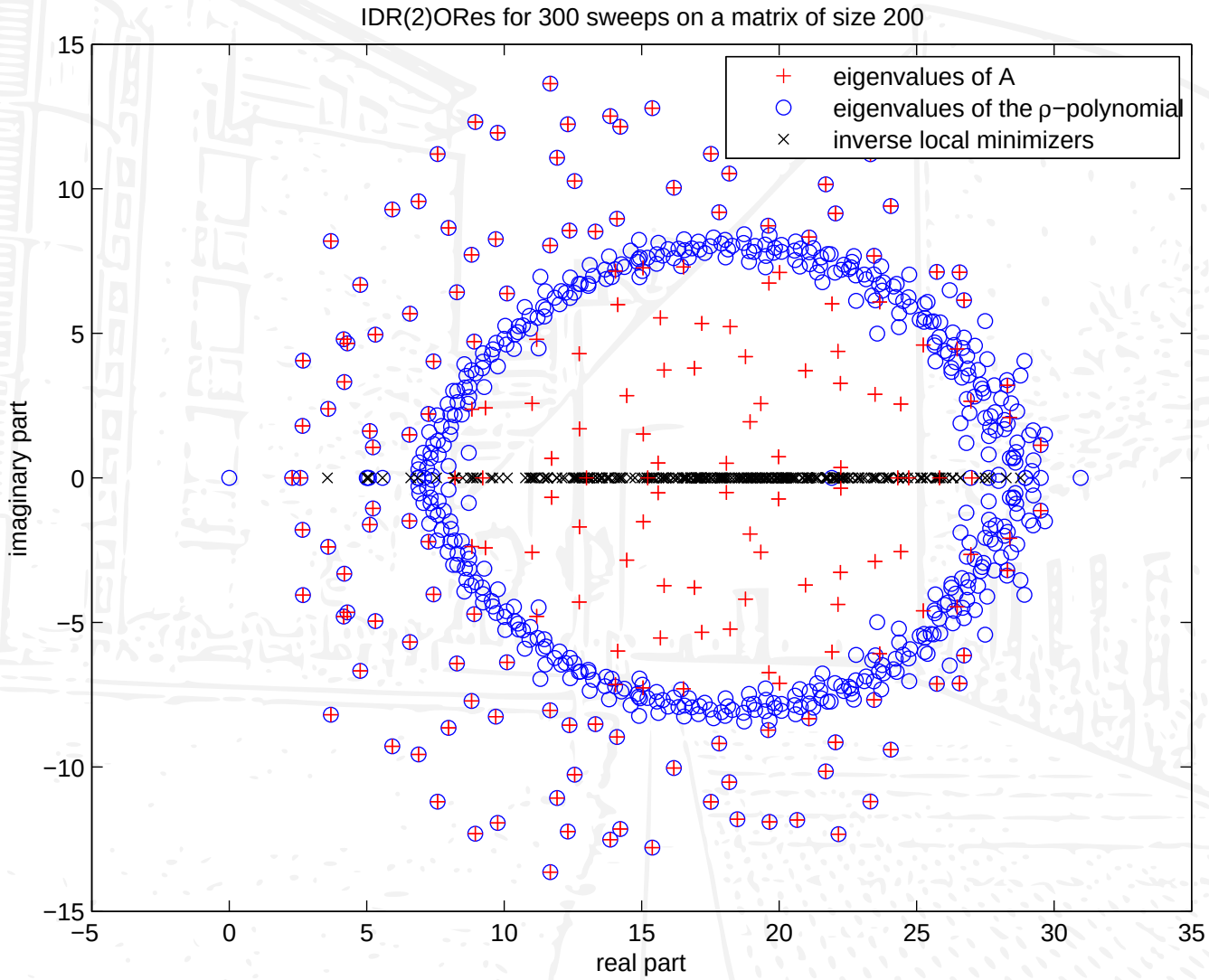
Understanding IDR: 100 steps for $s = 2$; $\mathbf{P}$ real



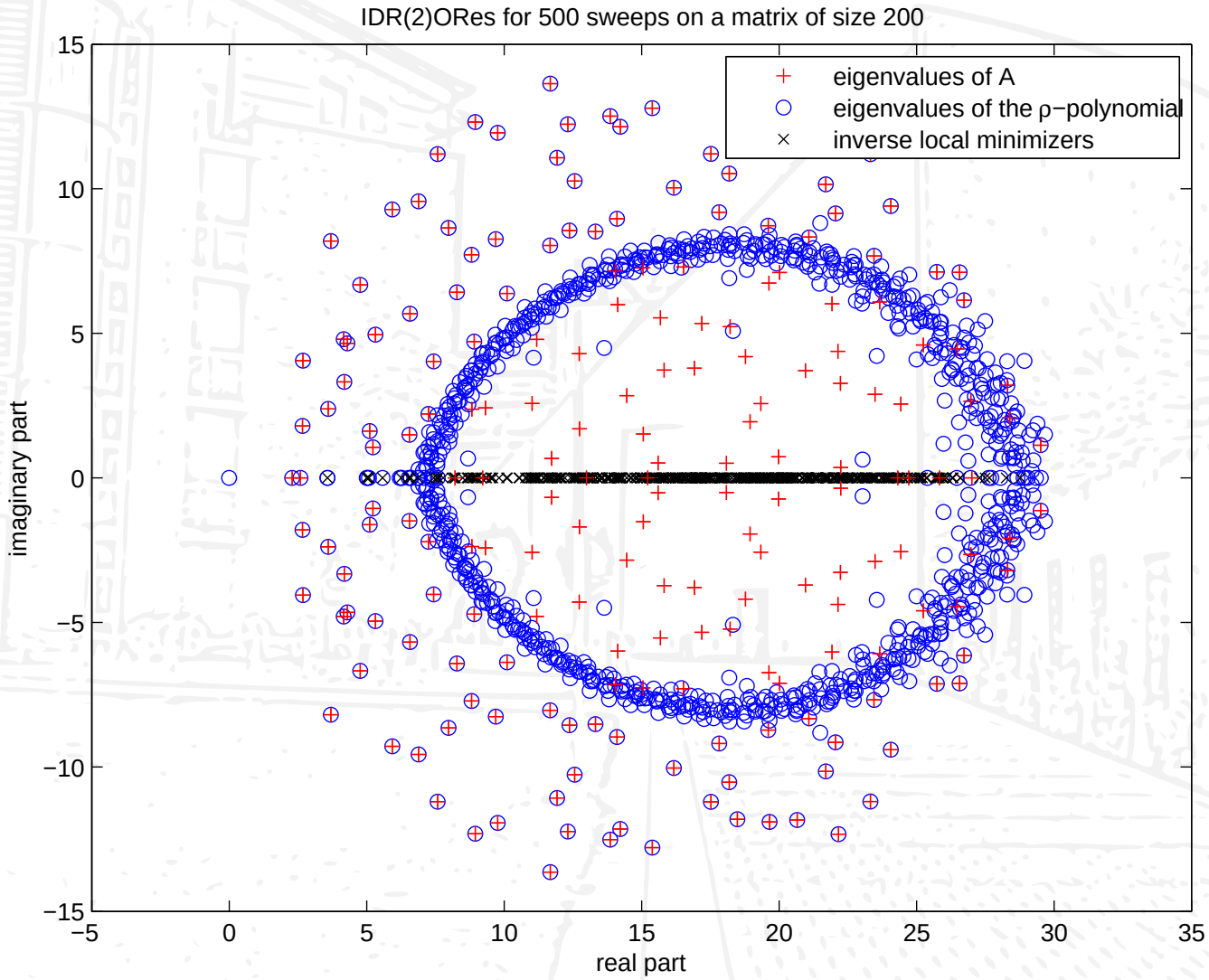
Understanding IDR: 150 steps for $s = 2$; $\mathbf{P}$ real



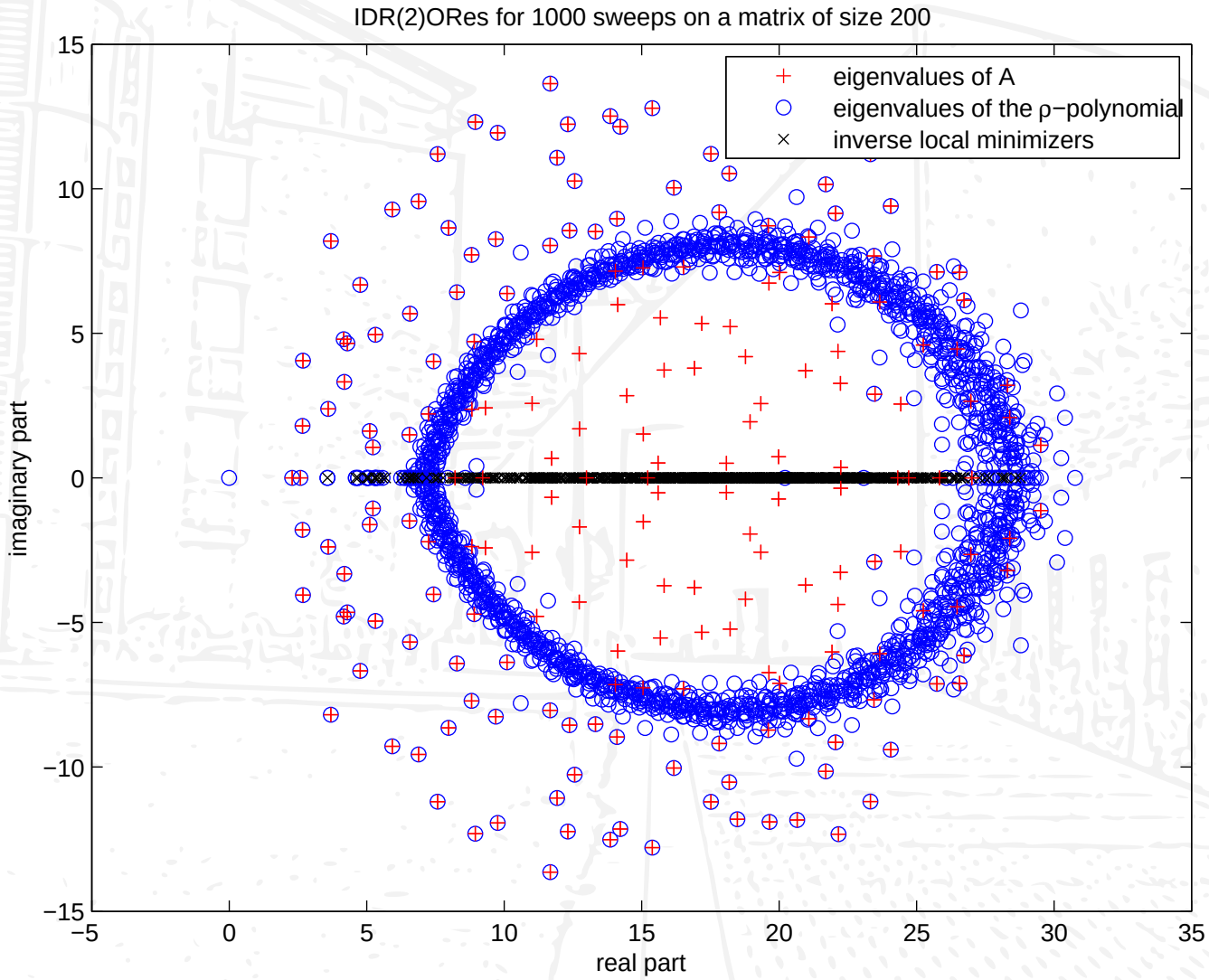
Understanding IDR: 300 steps for $s = 2$; $\mathbf{P}$ real



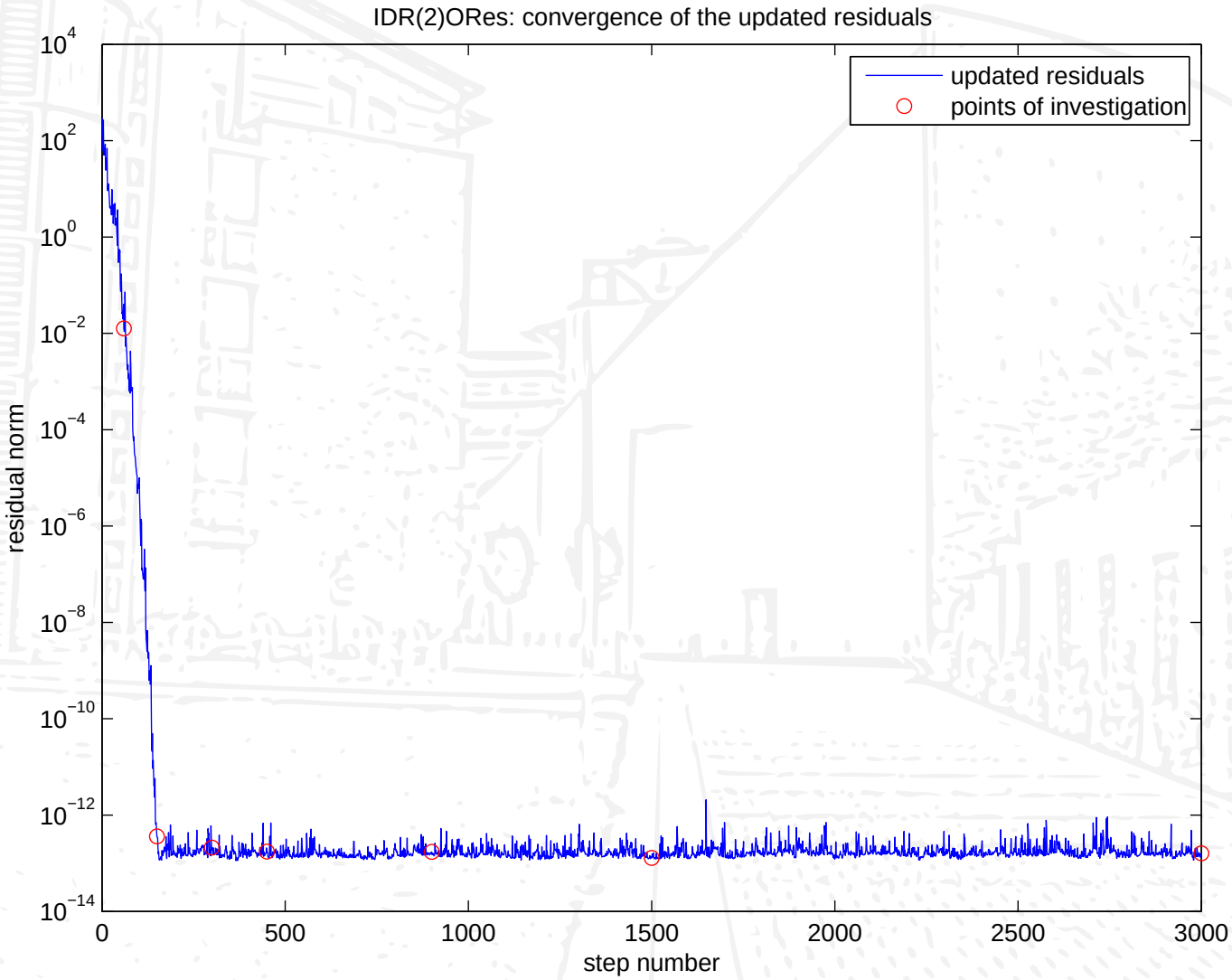
Understanding IDR: 500 steps for $s = 2$; $\mathbf{P}$ real



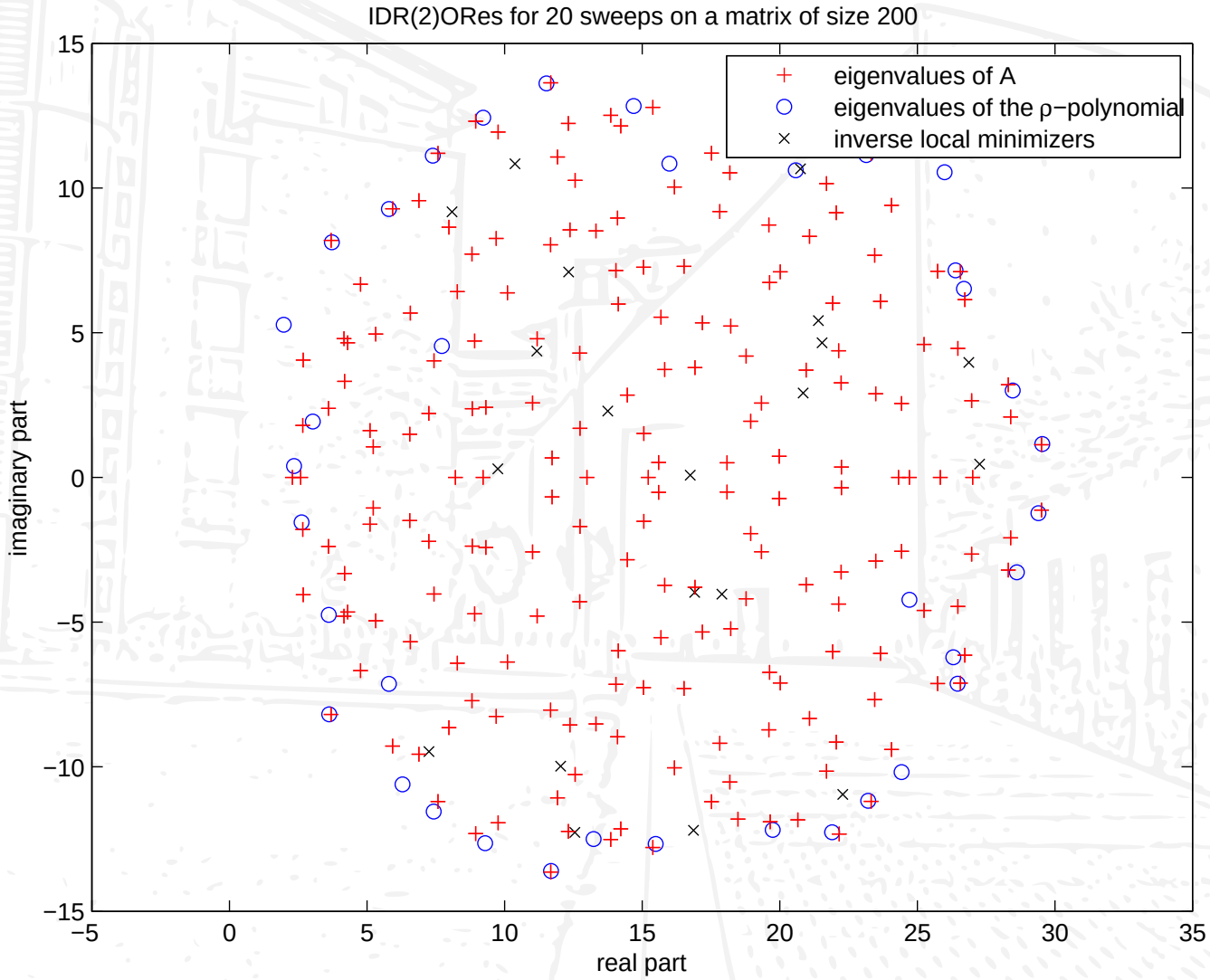
Understanding IDR: 1000 steps for $s = 2$; $\mathbf{P}$ real



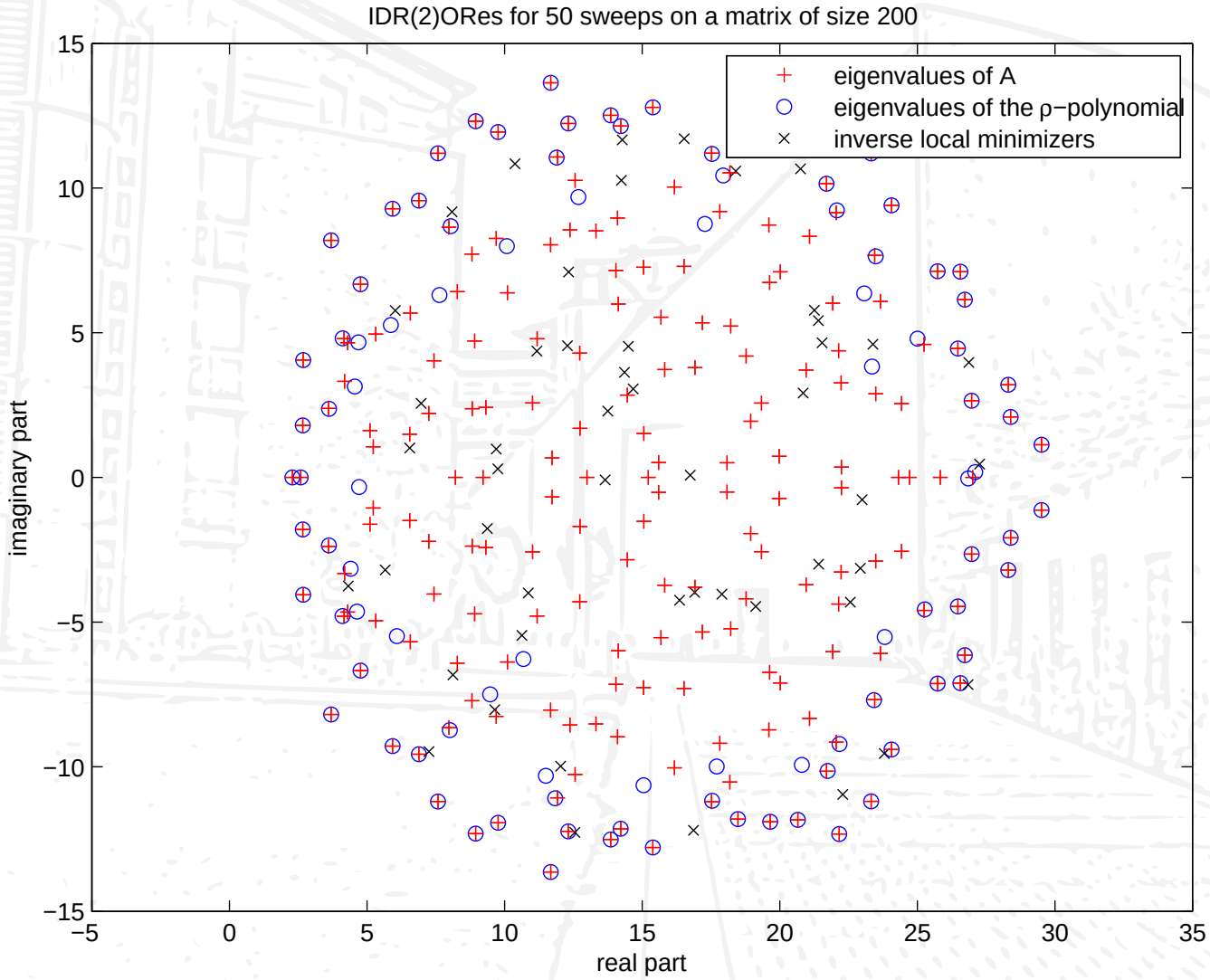
Understanding IDR: Convergence for $s = 2$; $\mathbf{P}$ complex



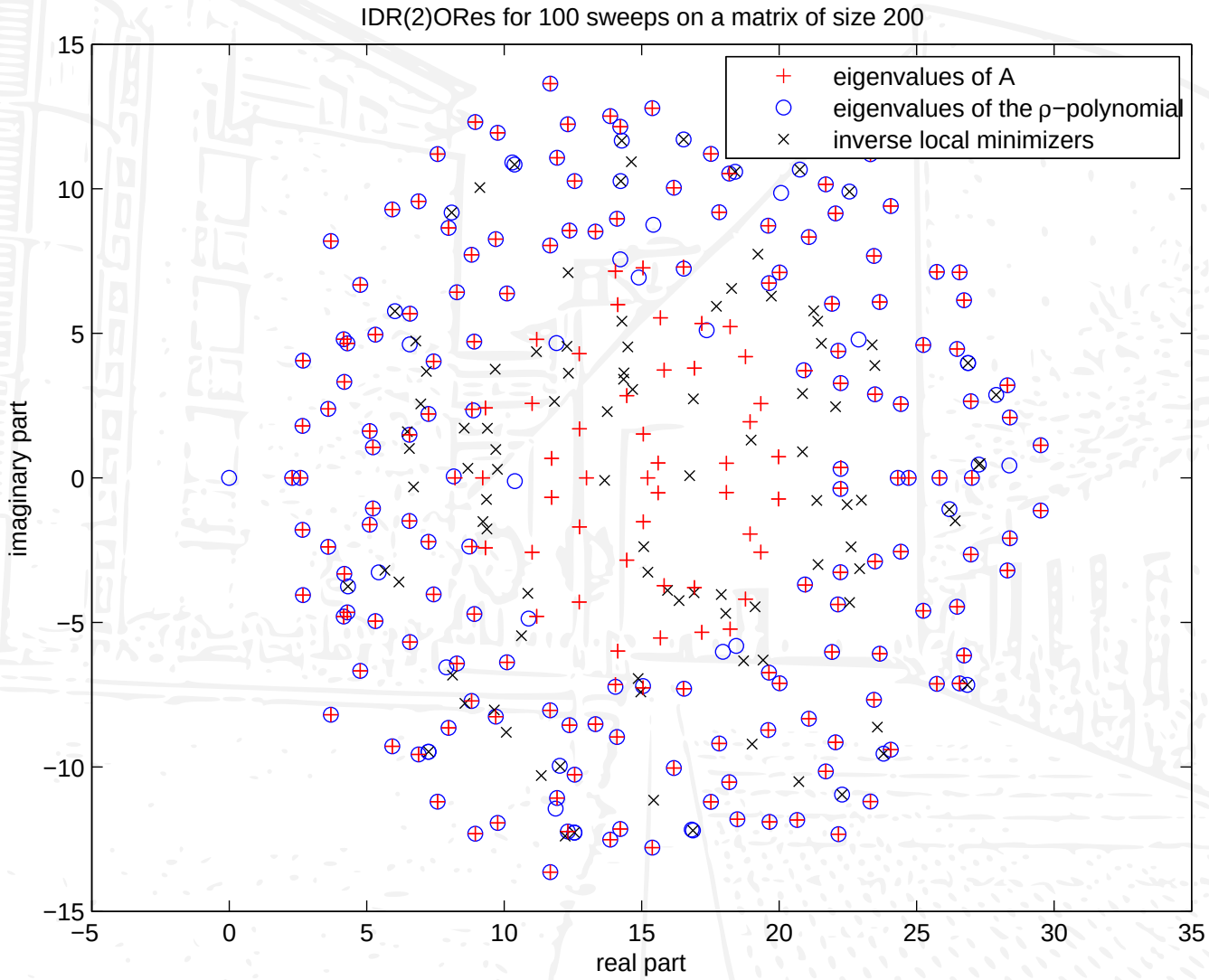
Understanding IDR: 20 steps for $s = 2$; $\mathbf{P}$ complex



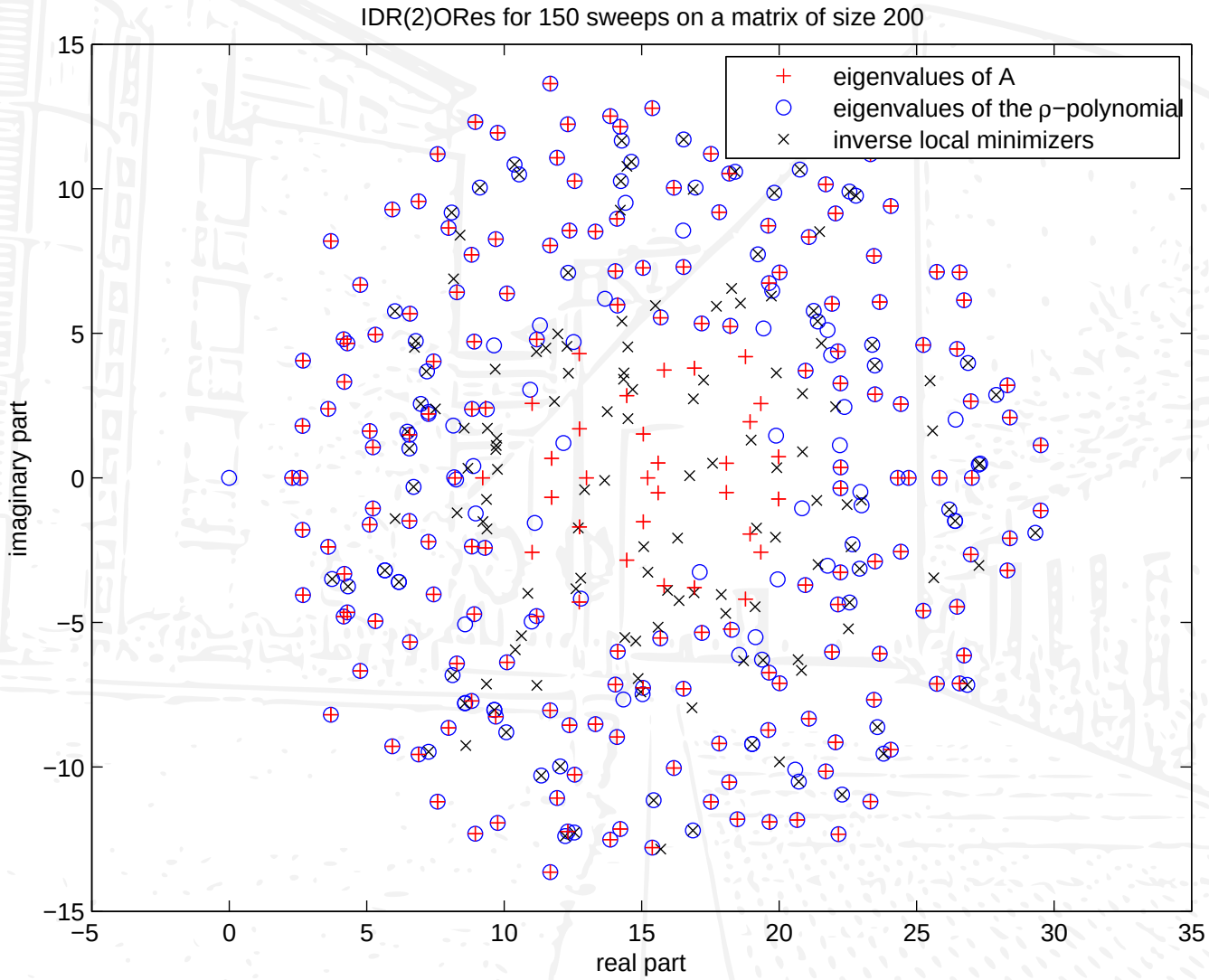
Understanding IDR: 50 steps for $s = 2$; $\mathbf{P}$ complex



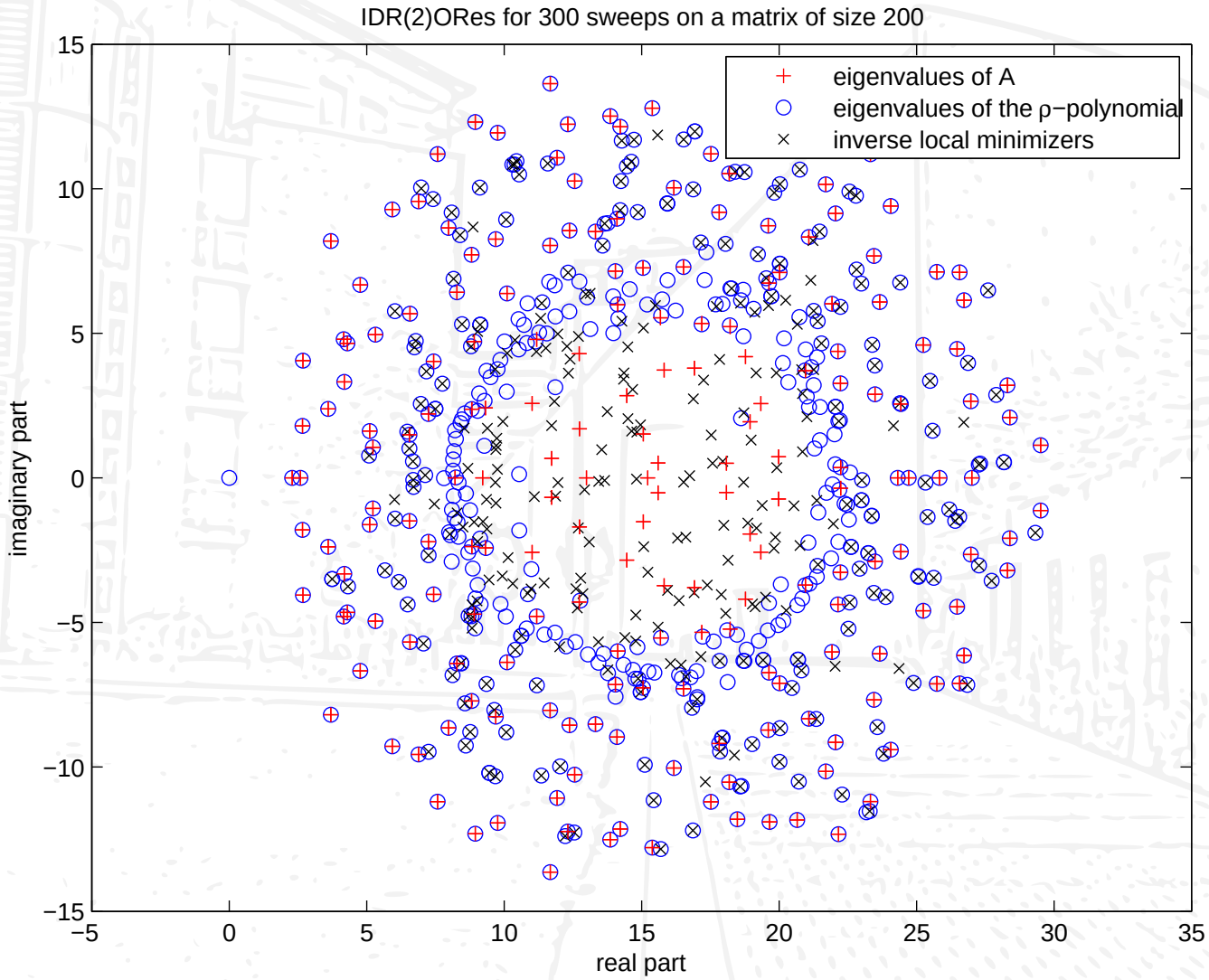
Understanding IDR: 100 steps for $s = 2$; $\mathbf{P}$ complex



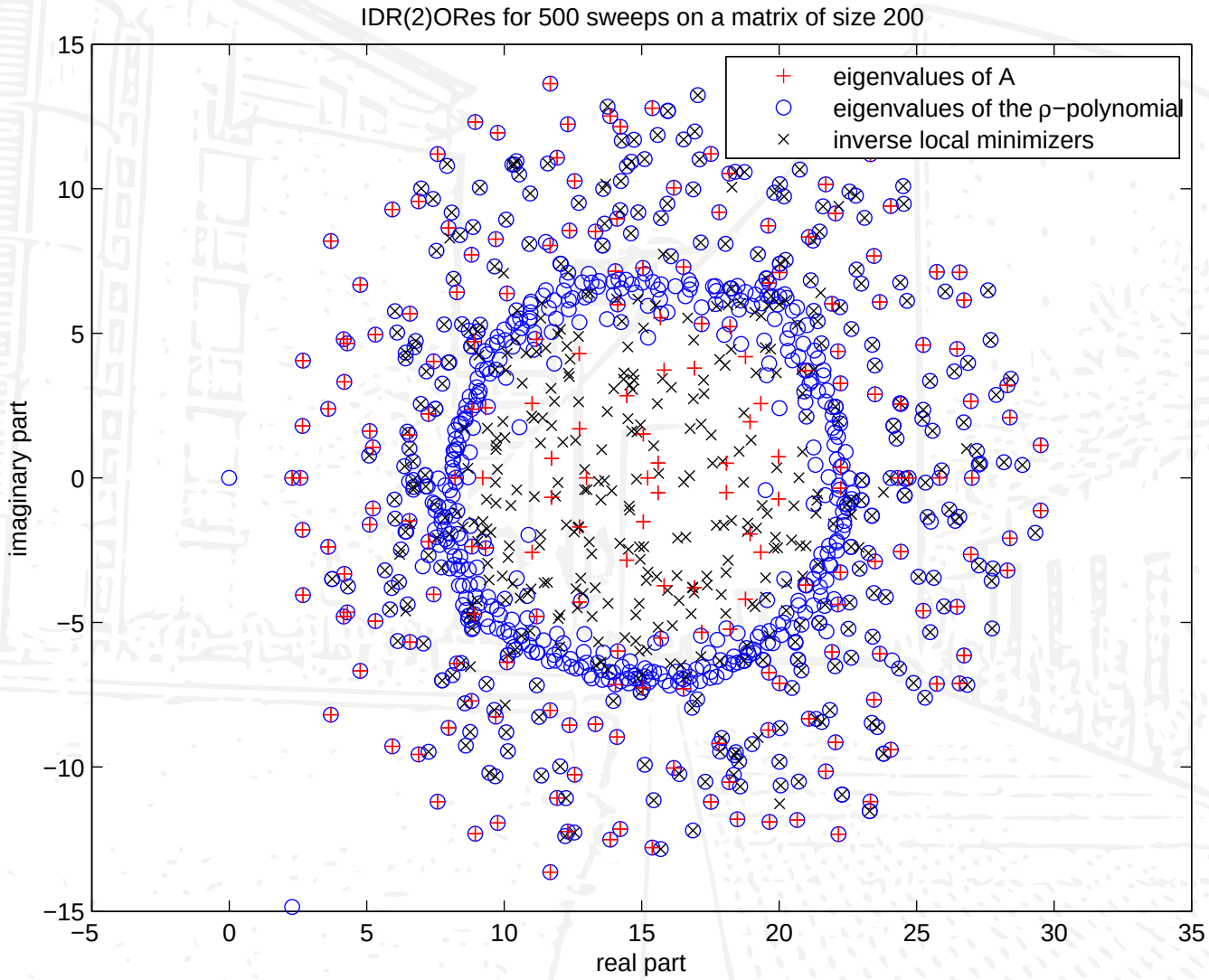
Understanding IDR: 150 steps for $s = 2$; $\mathbf{P}$ complex



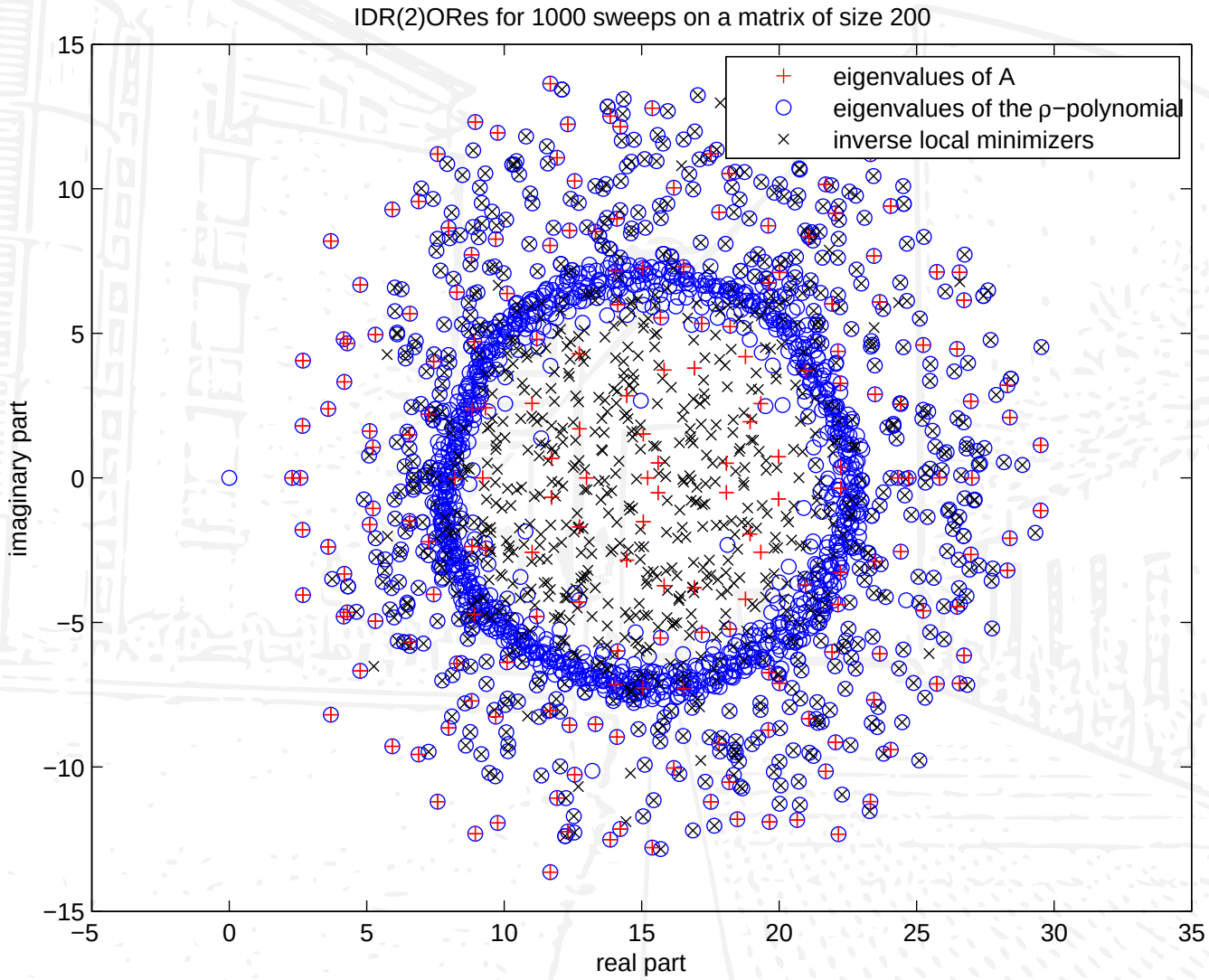
Understanding IDR: 300 steps for $s = 2$; $\mathbf{P}$ complex



Understanding IDR: 500 steps for $s = 2$; $\mathbf{P}$ complex



Understanding IDR: 1000 steps for $s = 2$; $\mathbf{P}$ complex



Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).
- IDR, and, more general, IDR(s), is affected by finite precision: In contrast to the Lanczos method the delay in convergence is due to Ritz values of the Lanczos-part converging to the minimizers $1/\omega_j$.

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).
- IDR, and, more general, IDR(s), is affected by finite precision: In contrast to the Lanczos method the delay in convergence is due to Ritz values of the Lanczos-part converging to the minimizers $1/\omega_j$.
- In IDR, BiCGStab and IDR(s) we did not observe multiple Ritz approximations to simple eigenvalues.

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).
- IDR, and, more general, IDR(s), is affected by finite precision: In contrast to the Lanczos method the delay in convergence is due to Ritz values of the Lanczos-part converging to the minimizers $1/\omega_j$.
- In IDR, BiCGStab and IDR(s) we did not observe multiple Ritz approximations to simple eigenvalues.
- The Ritz values build up a "barrier" and cluster there, once IDR(s)ORes has reached the ultimately attainable accuracy.

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).
- IDR, and, more general, IDR(s), is affected by finite precision: In contrast to the Lanczos method the delay in convergence is due to Ritz values of the Lanczos-part converging to the minimizers $1/\omega_j$.
- In IDR, BiCGStab and IDR(s) we did not observe multiple Ritz approximations to simple eigenvalues.
- The Ritz values build up a "barrier" and cluster there, once IDR(s)ORes has reached the ultimately attainable accuracy.
- At the same time a Ritz value close to zero shows up.

Conclusions and Outlook

- We related the convergence of finite precision Krylov methods (including IDR and IDR(s)) to the behaviour of the Ritz values.
- We presented the transition from IDR(s)ORes to BiORes(s,1). The results obtained in transition enable us to compute the Ritz values. This is the IDR(s)Eig approach sketched in (Gutknecht and Z., 2010).
- We gave selected numerical examples for (the symmetric and non-symmetric variant of) the Lanczos method, BiCGStab and IDR(s).
- IDR, and, more general, IDR(s), is affected by finite precision: In contrast to the Lanczos method the delay in convergence is due to Ritz values of the Lanczos-part converging to the minimizers $1/\omega_j$.
- In IDR, BiCGStab and IDR(s) we did not observe multiple Ritz approximations to simple eigenvalues.
- The Ritz values build up a "barrier" and cluster there, once IDR(s)ORes has reached the ultimately attainable accuracy.
- At the same time a Ritz value close to zero shows up.
- We still have to carefully look at the perturbation terms of the underlying perturbed Lanczos process.

Thank you very much for your attention!

どうも有難う御座います

藤野先生。

Thank you very much for your attention!

どうも有難う御座います

藤野先生。

どうも有難う御座いました。

Bai, Z. (1994).
Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem.
Mathematics of Computation, 62(205):209–226.

Greenbaum, A. (1989).
Behaviour of slightly perturbed Lanczos and conjugate-gradient recurrences.
Linear Algebra and its Applications, 113:7–63.

Greenbaum, A. and Strakoš, Z. (1992).
Predicting the behavior of finite precision Lanczos and conjugate gradient computations.
SIAM J. Matrix Anal. Appl., 13(1):121–137.

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).

Meurant, G. (2006).
The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computation.
SIAM, Philadelphia.

Meurant, G. and Strakoš, Z. (2006).
The Lanczos and conjugate gradient algorithms in finite precision arithmetic.
Acta Numerica, 15:471–542.

Paige, C. C. (1971).
The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.
PhD thesis, London University Institute of Computer Science.

Paige, C. C. (1972).
Computational variants of the Lanczos method for the eigenproblem.
J. Inst. Maths Applics, 10:373–381.

Paige, C. C. (1976).
Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix.
J. Inst. Math. Appl., 18(341–349).

Paige, C. C. (1980).
Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem.
Linear Algebra and its Applications, 34:235–258.

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.

Stiefel, E. (1955).
Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme.
Comment. Math. Helv., 29:157–179.

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.

Zemke, J.-P. M. (2006).
Hessenberg eigenvalue-eigenmatrix relations.
Linear Algebra and its Applications, 414(2–3):589–606.

Zemke, J.-P. M. (2007).
Abstract perturbed Krylov methods.
Linear Algebra and its Applications, 424(2–3):405–434.

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.