The IDR approach for solving large nonsymmetric linear systems and eigenvalue problems

ICCAM conference, Leuven

Martin van Gijzen

July 8, 2010

Institute of Applied Mathematics

TU Delft

Delft University of Technology

Outline

- Introduction
- The IDR approach for solving linear systems
- Polynomial analysis:
  - The Bi-CG method
  - IDR(1) and Bi-CGSTAB
  - IDR(s)
- Algorithmic variants
- IDR for eigenvalues

Introduction

The iterative solution of large linear systems

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

is still an active area of research.

In the eighties and nineties many iterative solvers have been proposed.

The focus of the research has since then been mainly on the development of preconditioners.

Still, the search for faster and more robust iterative solvers remains important, in particular for nonsymmetric problems.

The most popular iterative methods

$\boldsymbol{A}$ is symmetric positive definite:

- CG: minimizes the $\boldsymbol{A}$-norm of the error over the Krylov subspace

$$\mathcal{K}^n(\boldsymbol{A}, \boldsymbol{r}_0) = \boldsymbol{r}_0 \oplus \boldsymbol{A}\boldsymbol{r}_0 \oplus \boldsymbol{A}^2\boldsymbol{r}_0 \oplus \cdots \oplus \boldsymbol{A}^n\boldsymbol{r}_0 \ ,$$

  and uses short recursions.

$\boldsymbol{A}$ nonsymmetric:

- GMRES: minimizes the residual norm over the Krylov subspace and uses long recurrences
- Bi-CGSTAB: does not minimize an error norm, but uses short recurrences

IDR(s)

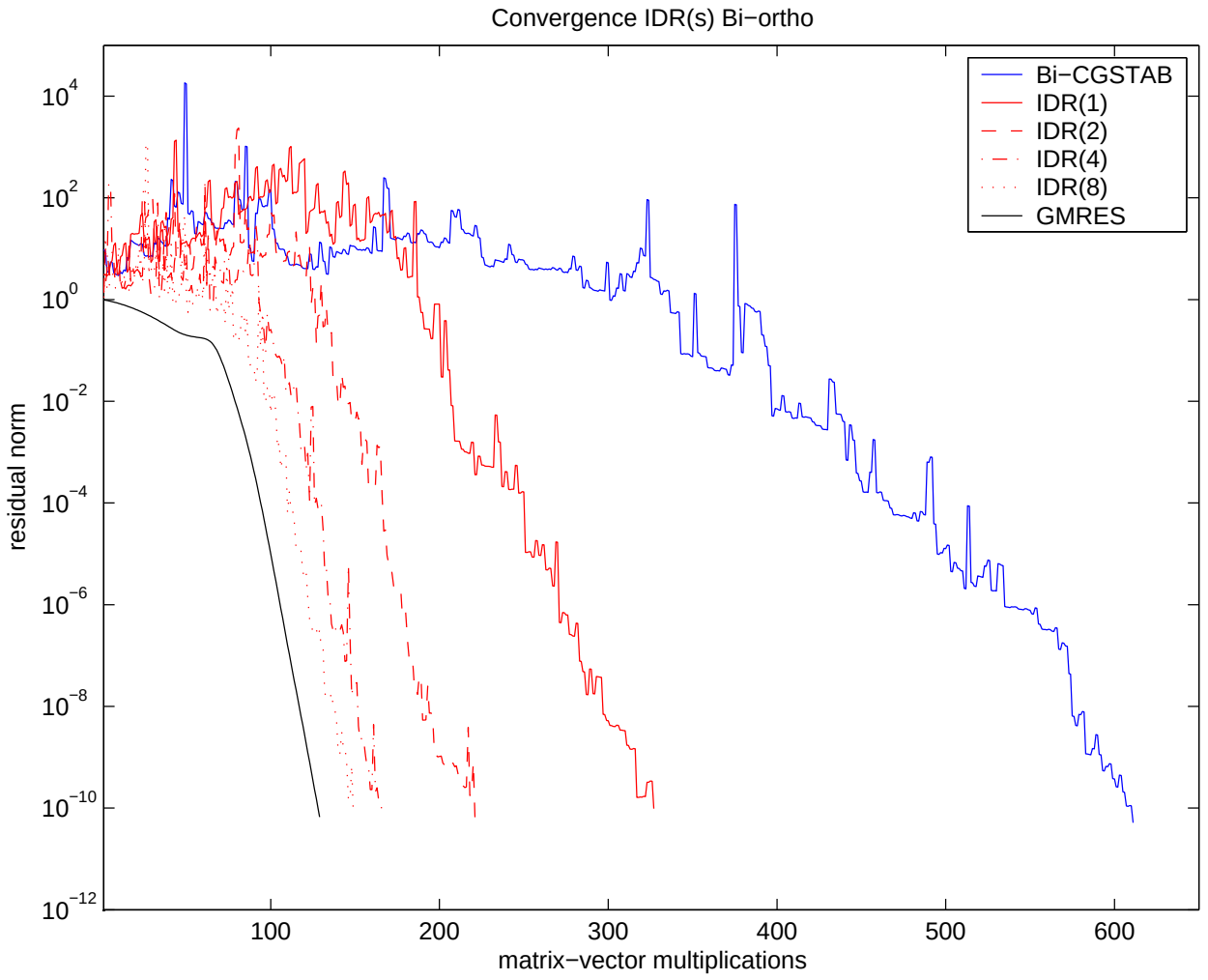
IDR(s) is a new family of Krylov subspace methods.

All IDR-methods are based on the IDR-theorem that provides a way to generate a sequence of nested subspaces of shrinking dimension.

Like Bi-CGSTAB, IDR(s) uses short recurrences, and does not minimize an error over the Krylov subspace.

The IDR(s) paper, joint with Peter Sonneveld, appeared in SISC in 2008 and has generated active new research in Krylov subspace methods.

IDR(s) applied to a difficult problem



The IDR approach for solving $Ax = b$

Generate residuals $\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$ that are in subspaces $\mathcal{G}_j$ of decreasing dimension.

These nested subspaces are related by

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A})(\mathcal{G}_{j-1} \cap \boldsymbol{P}^{\perp}) \quad \boldsymbol{P} = [\boldsymbol{p}_1 \cdots \boldsymbol{p}_s]$$

where $\omega_j \in \mathbb{C}$'s are non-zero scalars.

Then the IDR theorem states that:

i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ for all $j > 0$.

ii) $\mathcal{G}_j = \{\mathbf{0}\}$ for some $j \leq N$.

Making an IDR algorithm

Every IDR-based algorithm consists of two different steps:

- The dimension reduction steps:
  Given sufficient vectors in $\mathcal{G}_j$, compute first residual in $\mathcal{G}_{j+1}$;
- Intermediate steps:
  Compute sufficient vectors in $\mathcal{G}_j$ to make a dimension reduction step.

The dimension reduction step

Given $\boldsymbol{g}_1^j \cdots \boldsymbol{g}_s^j, \boldsymbol{r}_s^j \in \mathcal{G}_j$, a vector $\boldsymbol{v} \in \mathcal{G}_j \cap \boldsymbol{P}^{\perp}$ is computed by

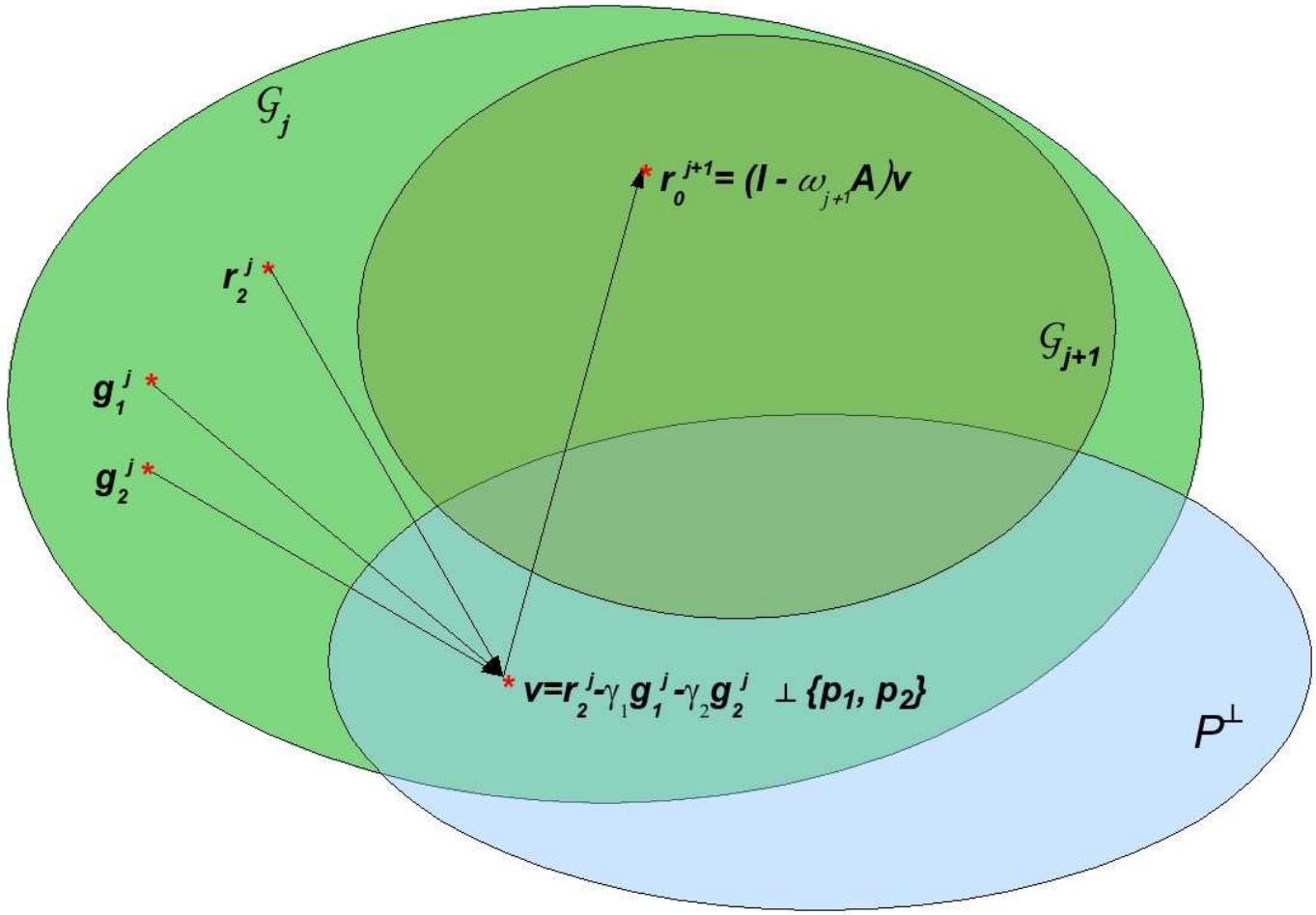
$$\boldsymbol{v} = \boldsymbol{r}_s^j - \sum_{i=1}^{s} \gamma_i \boldsymbol{g}_i^j,$$

with γ_i such that $\boldsymbol{P}^H \boldsymbol{v} = 0$.

The first residual in $\mathcal{G}_{j+1}$ is then computed by

$$\boldsymbol{r}_0^{j+1} = (\boldsymbol{I} - \omega_{j+1}\boldsymbol{A})\boldsymbol{v}$$

The dimension reduction step



Consistent updates

We also need the corresponding iterate $\boldsymbol{x}_0^{j+1}$. In practice this means that if we update $\boldsymbol{x}$ with $\boldsymbol{u}$:

$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{u}$$

we have to update the residual by

$$\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{A}\boldsymbol{u}$$

The vectors $\boldsymbol{g}_i^j$ are update vectors for the residuals, so we need $\boldsymbol{u}_i^j$ such that

$$\boldsymbol{g}_i^j = \boldsymbol{A}\boldsymbol{u}_i^j$$

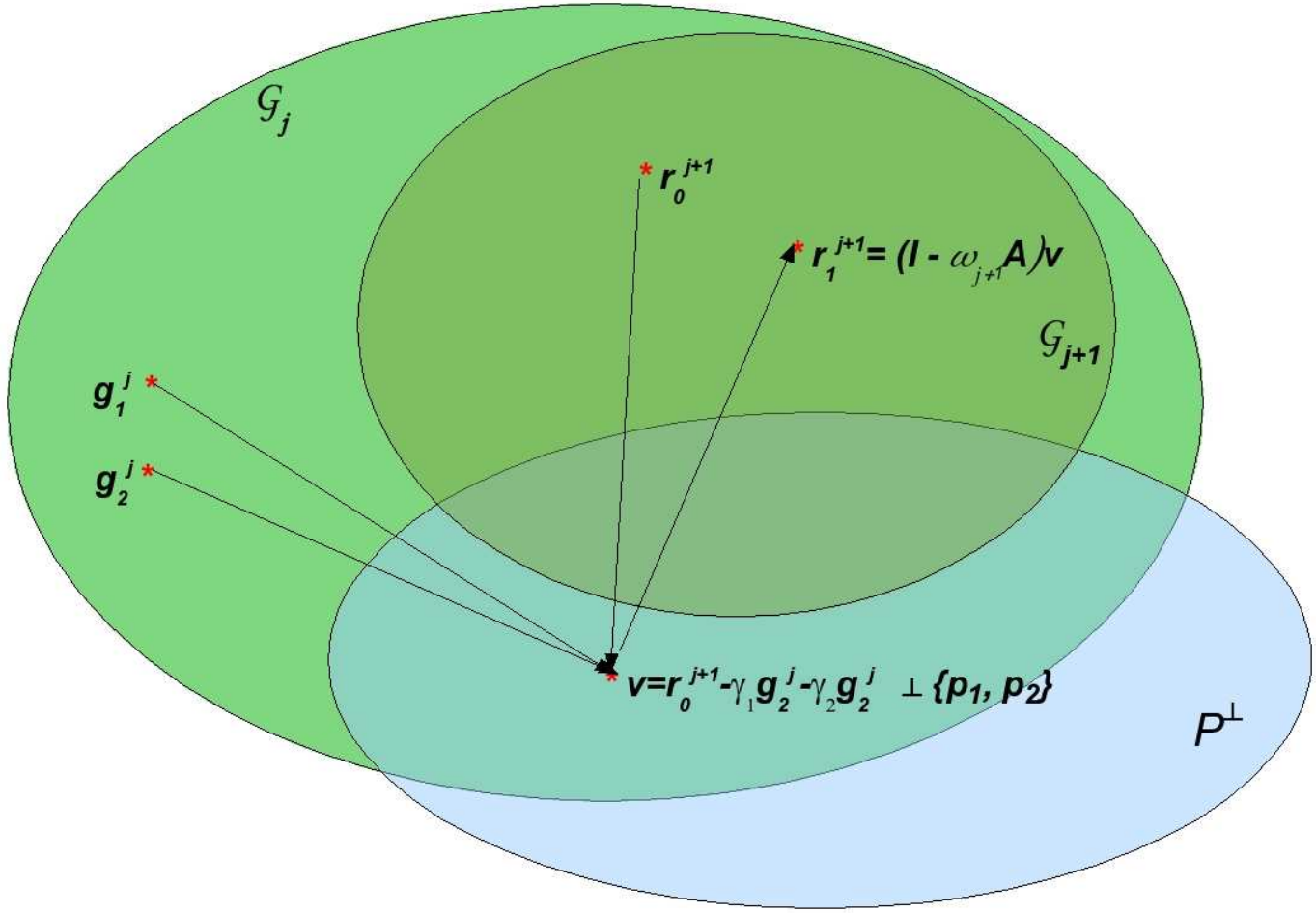
Intermediate steps

s intermediate steps are needed to compute
$\boldsymbol{g}_1^{j+1} \cdots \boldsymbol{g}_s^{j+1}, \boldsymbol{r}_s^{j+1} \in \mathcal{G}_{j+1}$.

New vectors $\in \mathcal{G}_{j+1}$ can be computed by repeating the algorithm.

ω_j has to be kept constant.

Example: computation of r_1^{j+1}



Prototype IDR(s) algorithm.

Calculate $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$;
$\boldsymbol{G} = \boldsymbol{O} \in \mathbb{C}^{N\times s}$; $\boldsymbol{U} = \boldsymbol{O} \in \mathbb{C}^{N\times s}$; $\boldsymbol{M} = \boldsymbol{I} \in \mathbb{C}^{s\times s}$; $\omega = 1$;
while $\|\boldsymbol{r}\| > TOL$ **do**
  for $k = 1$ to s **do**
    $\boldsymbol{f} = \boldsymbol{P}^H \boldsymbol{r}$;
    Solve $\boldsymbol{c}$ from $\boldsymbol{M}\boldsymbol{c} = \boldsymbol{f}$;
    $\boldsymbol{v} = \boldsymbol{r} - \boldsymbol{G}\boldsymbol{c}$;
    $\boldsymbol{u}_k = \boldsymbol{U}\boldsymbol{c} + \omega\boldsymbol{v}$; $\boldsymbol{g}_k = \boldsymbol{A}\boldsymbol{u}_k$;
    $\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{g}_k$; $\quad \boldsymbol{x} = \boldsymbol{x} + \boldsymbol{u}_k$;
    $\boldsymbol{G}(:,k) = \boldsymbol{g}_k$; $\boldsymbol{U}(:,k) = \boldsymbol{u}_k$; $\boldsymbol{M}(:,k) = \boldsymbol{P}^H \boldsymbol{g}_k$;
  end for
  Compute $\boldsymbol{f} = \boldsymbol{P}^H \boldsymbol{r}$;
  Solve $\boldsymbol{c}$ from $\boldsymbol{M}\boldsymbol{c} = \boldsymbol{f}$;
  $\boldsymbol{v} = \boldsymbol{r} - \boldsymbol{G}\boldsymbol{c}$;
  $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{v}$; $\omega = (\boldsymbol{t}^H \boldsymbol{v})/(\boldsymbol{t}^H \boldsymbol{t})$;
  $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{U}\boldsymbol{c} + \omega\boldsymbol{v}$; $\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{G}\boldsymbol{c} - \omega\boldsymbol{t}$;
end while

Stommel's model for ocean circulation

Balance between bottom friction, wind stress and Coriolis force.

$$-r\,\Delta\psi - \beta\,\frac{\partial\psi}{\partial x} - = (\nabla \times \mathbf{F})_z$$

plus circulation condition around islands k

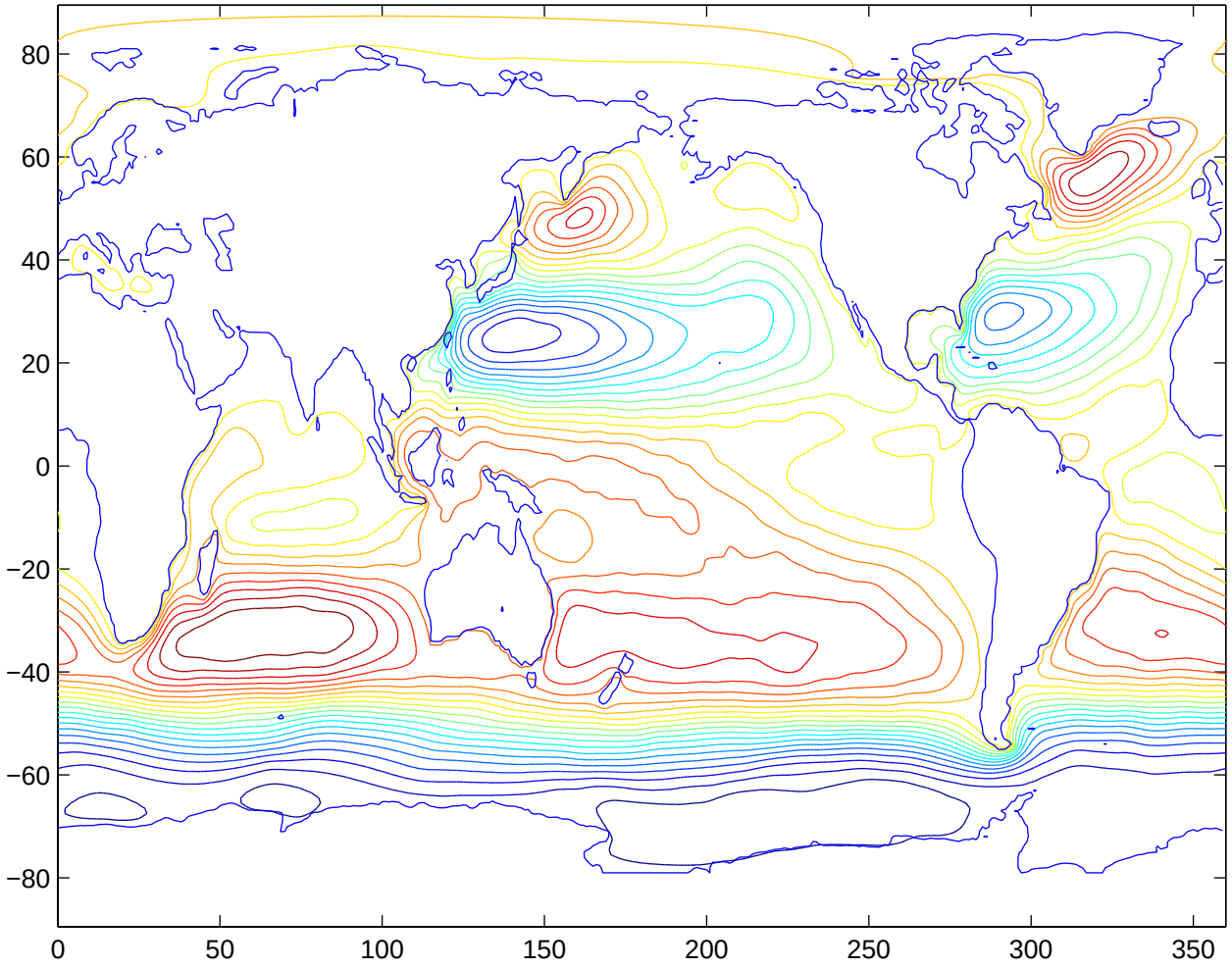
$$\oint_{\Gamma_k} r\,\frac{\partial\psi}{\partial n}\,ds = -\oint_{\Gamma_k} \mathbf{F}\cdot\mathbf{s}\,ds.$$

- ψ: streamfunction
- r: bottom friction parameter
- β: Coriolis parameter
- $\mathbf{F}$: Wind stress

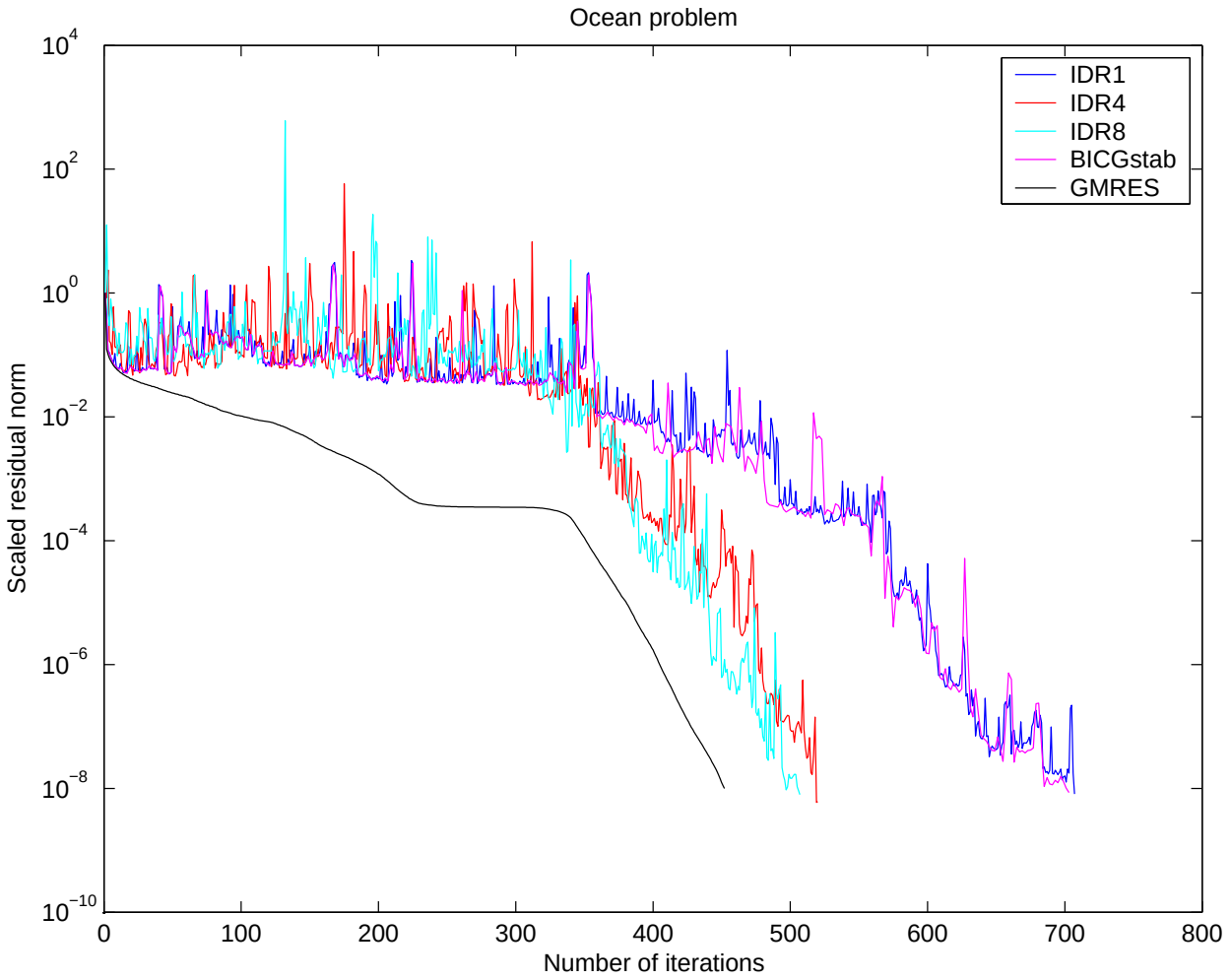
Discretization of the ocean problem

- Discretization with linear finite elements
- Results in nonsymmetric system of 42248
- Eigenvalues are (almost) real
- ILU(0.1) as preconditioner

Solution of the ocean problem



Convergence for the ocean problem



Relation with CG-type methods

The idea behind IDR(s) is to generate residuals that are in the nested subspace $\mathcal{G}_j$ of shrinking dimension.
The IDR theorem gives a recursive definition of these subspaces.

This viewpoint is different from the more conventional Krylov (CG-type) methods. Essential to these are the following ingredients:

- Implicitly or explicitly generate a basis for the Krylov subspace, using a Lanczos-type algorithm;
- Apply a (Petrov-)Galerkin condition to compute an approximate solution.

Different viewpoints on IDR

The difference in approach makes IDR(s) harder to understand. Several papers have appeared that put IDR(s) in a more conventional framework:

- M. Gutknecht, IDR explained, ETNA 2010 (report 2008)
- G.L.G. Sleijpen, P. Sonneveld, and M.B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method, APNUM to appear (report 2008)
- V. Simoncini and D. Szyld, Interpreting IDR as a Petrov-Galerkin method, SISC 2010 (report 2009)
- M. Gutknecht and J. Zemke , Eigenvalue computations based on IDR, (report 2010)

Polynomial analysis

- Why does IDR(s) converge?

Polynomial analysis

- Why does IDR(s) converge?
- Finite termination of IDR(s)?

Polynomial analysis

- Why does IDR(s) converge?
- Finite termination of IDR(s)?
- What is the relation with other Krylov methods?

Polynomial analysis

- Why does IDR(s) converge?
- Finite termination of IDR(s)?
- What is the relation with other Krylov methods?
- IDR(s) is a Krylov subspace method

Polynomial analysis

- Why does IDR(s) converge?
- Finite termination of IDR(s)?
- What is the relation with other Krylov methods?
- IDR(s) is a Krylov subspace method
- Residuals satisfy $\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$, Φ_n is an n-th degree polynomial.

Polynomial analysis

- Why does IDR(s) converge?
- Finite termination of IDR(s)?
- What is the relation with other Krylov methods?
- IDR(s) is a Krylov subspace method
- Residuals satisfy $\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$, Φ_n is an n-th degree polynomial.
- Analyze those polynomials

The CG algorithm

To relate IDR(s) to more conventional methods we first look at CG.

Regular steps in CG algorithm:

$$\rho_n = \boldsymbol{r}_n^T \boldsymbol{r}_n, \;\; \beta_n = \rho_n / \rho_{n-1}$$

$$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_n \boldsymbol{p}_{n-1}, \;\; \boldsymbol{q}_n = \boldsymbol{A}\boldsymbol{p}_n;$$

$$\sigma_n = \boldsymbol{p}_n^T \boldsymbol{q}_n, \;\; \alpha_n = \rho_n / \sigma_n$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{q}_n;$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n$$

The CG algorithm (2)

Recall that CG minimizes the $\boldsymbol{A}$-norm of the error.

This optimality property is (for $\boldsymbol{A}$ SPD) mathematically equivalent with

$$\boldsymbol{r}_{n+1} \perp \mathcal{K}^n(\boldsymbol{A}, \boldsymbol{r}_0) \ .$$

Hence $\boldsymbol{r}_{n+1} \perp \boldsymbol{r}_j, \ j = 1, \cdots n$.

This and the fact that CG is a Krylov subspace method make it possible to see CG as a construction method for polynomials that are orthogonal with respect to a special inner product.

The polynomial relations

- The relevant vectors satisfy:

$$\boldsymbol{r}_n = \varphi_n(\boldsymbol{A})\boldsymbol{r}_0, \;\; \boldsymbol{p}_n = \psi_n(\boldsymbol{A})\boldsymbol{r}_0$$

  where φ_n and ψ_n are polynomials of degree n.

- Define inner product between polynomials:

$$\langle \phi_1, \phi_2 \rangle = \boldsymbol{r}_0^T \phi_1(\boldsymbol{A})\phi_2(\boldsymbol{A})\boldsymbol{r}_0 = [\phi_1(\boldsymbol{A})\boldsymbol{r}_0]^T \phi_2(\boldsymbol{A})\boldsymbol{r}_0$$

- then orthogonality between $\boldsymbol{r}_n$ and $\boldsymbol{r}_k$ corresponds to orthogonality of φ_n and φ_k.

CG for orthogonal polynomials

The coefficients and polynomials can also be calculated by

$$\rho_n = \langle \varphi_n, \varphi_n \rangle, \ \beta_n = \rho_n / \rho_{n-1}$$
$$\psi_n(t) = \varphi_n(t) + \beta_n \psi_{n-1}(t),$$
$$\sigma_n = \langle \psi_n, t\psi_n \rangle, \ \alpha_n = \rho_n / \sigma_n$$
$$\varphi_{n+1}(t) = \varphi_n(t) - \alpha_n t \psi_n(t)$$

This is (part of) the CG-algorithm for orthogonal polynomials.

Nonsymmetric systems

It is not possible to make a Krylov method for nonsymmetric problems that use short recurrences and that minimize the error in some norm over the Krylov subspace.

The Bi-CG method is a generalization of CG that computes residuals that are orthogonal wrt. a 'shadow' Krylov subspace.

The method uses short recurrences, but has no optimality property for general systems.

The Bi-CG algorithm

Regular steps in Bi-CG algorithm:

$$\rho_n = \widetilde{\boldsymbol{r}}_n^T \boldsymbol{r}_n, \;\; \beta_n = \rho_n / \rho_{n-1}$$

$$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_n \boldsymbol{p}_{n-1}, \;\; \boldsymbol{q}_n = \boldsymbol{A}\boldsymbol{p}_n;$$

$$\widetilde{\boldsymbol{p}}_n = \widetilde{\boldsymbol{r}}_n + \beta_n \widetilde{\boldsymbol{p}}_{n-1}, \;\; \widetilde{\boldsymbol{q}}_n = \boldsymbol{A}^T \widetilde{\boldsymbol{p}}_n$$

$$\sigma_n = \widetilde{\boldsymbol{p}}_n^T \boldsymbol{q}_n, \;\; \alpha_n = \rho_n / \sigma_n$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{q}_n;$$

$$\widetilde{\boldsymbol{r}}_{n+1} = \widetilde{\boldsymbol{r}}_n - \alpha_n \widetilde{\boldsymbol{q}}_n$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n$$

The polynomial relations

- The relevant vectors satisfy:

$$\boldsymbol{r}_n = \varphi_n(\boldsymbol{A})\boldsymbol{r}_0, \;\; \boldsymbol{p}_n = \psi_n(\boldsymbol{A})\boldsymbol{r}_0$$

$$\widetilde{\boldsymbol{r}}_n = \varphi_n(\boldsymbol{A}^T)\widetilde{\boldsymbol{r}}_0, \;\; \widetilde{\boldsymbol{p}}_n = \psi_n(\boldsymbol{A}^T)\widetilde{\boldsymbol{r}}_0$$

  where φ_n and ψ_n are polynomials of degree n.

- Define 'inner product' between polynomials:

$$\langle \phi_1, \phi_2 \rangle = \widetilde{\boldsymbol{r}}_0^T \phi_1(\boldsymbol{A})\phi_2(\boldsymbol{A})\boldsymbol{r}_0 = [\phi_1(\boldsymbol{A}^T)\widetilde{\boldsymbol{r}}_0]^T \phi_2(\boldsymbol{A})\boldsymbol{r}_0$$

- then bi-orthogonality between $\boldsymbol{r}_n$ and $\widetilde{\boldsymbol{r}}_k$ corresponds to orthogonality of φ_n and φ_k.

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.
- Going down, finally : $\boldsymbol{r} = \Omega_j(\boldsymbol{A})\widetilde{\boldsymbol{r}}$, with $\widetilde{\boldsymbol{r}} \in \mathcal{G}_0$. Here $\Omega_j(t) = (1 - \omega_j t)(1 - \omega_{j-1} t).\cdots.(1 - \omega_1 t)$.

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.
- Going down, finally : $\boldsymbol{r} = \Omega_j(\boldsymbol{A})\widetilde{\boldsymbol{r}}$, with $\widetilde{\boldsymbol{r}} \in \mathcal{G}_0$. Here $\Omega_j(t) = (1 - \omega_j t)(1 - \omega_{j-1} t).\cdots.(1 - \omega_1 t)$.
- Hence Φ_{2j} is divisible by Ω_j, $\Phi_{2j}(t) = \Omega_j(t)\phi_j(t)$

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.
- Going down, finally : $\boldsymbol{r} = \Omega_j(\boldsymbol{A})\widetilde{\boldsymbol{r}}$, with $\widetilde{\boldsymbol{r}} \in \mathcal{G}_0$. Here $\Omega_j(t) = (1 - \omega_j t)(1 - \omega_{j-1} t).\cdots.(1 - \omega_1 t)$.
- Hence Φ_{2j} is divisible by Ω_j, $\Phi_{2j}(t) = \Omega_j(t)\phi_j(t)$
- From the intersections with $\mathcal{S} = \boldsymbol{p}^{\perp}$ follows for $l < j$: $\boldsymbol{p}^T \Omega_l(\boldsymbol{A})\phi_j(\boldsymbol{A})\boldsymbol{r}_0 = 0$, so $\Omega_l(\boldsymbol{A}^T)\boldsymbol{p} \perp \phi_j(\boldsymbol{A})\boldsymbol{r}_0$.
  Hence ϕ_j is the j-th Bi-CG - polynomial

The IDR(1) polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.
- Going down, finally : $\boldsymbol{r} = \Omega_j(\boldsymbol{A})\widetilde{\boldsymbol{r}}$, with $\widetilde{\boldsymbol{r}} \in \mathcal{G}_0$. Here $\Omega_j(t) = (1 - \omega_j t)(1 - \omega_{j-1} t).\cdots.(1 - \omega_1 t)$.
- Hence Φ_{2j} is divisible by Ω_j, $\Phi_{2j}(t) = \Omega_j(t)\phi_j(t)$
- From the intersections with $\mathcal{S} = \boldsymbol{p}^{\perp}$ follows for $l < j$: $\boldsymbol{p}^T \Omega_l(\boldsymbol{A})\phi_j(\boldsymbol{A})\boldsymbol{r}_0 = 0$, so $\Omega_l(\boldsymbol{A}^T)\boldsymbol{p} \perp \phi_j(\boldsymbol{A})\boldsymbol{r}_0$. Hence ϕ_j is the j-th Bi-CG - polynomial
- Obtained without calculating $\boldsymbol{A}^T$ products.

Historical remarks

- IDR(1) was already discovered by Sonneveld in 1980.
- The connection with Bi-CG, however, led him to develop another method: CGS.
- This method caused a revolution in the Krylov Subspace world: it turned out to be possible to avoid MATVECS with A^T, and to speed up the convergence at the same time!
- Bi-CGSTAB, proposed by Henk van der Vorst, was developed as a STABilised variant of CGS. It is mathematically equivalent to IDR(1).
  Bi-CGSTAB is still the most widely used Bi-CG method (almost 2000 citations).

The IDR(s) polynomials.

- A similar polynomial analysis for IDR(s) yields:

$$\boldsymbol{r}_n = \Omega_j(\boldsymbol{A})\Psi_{n-j}(\boldsymbol{A})\boldsymbol{r}_0$$

with

$$\Omega_j(t) = (1-\omega_j t)(1-\omega_{j-1}t)\cdots(1-\omega_1 t),\ \Omega_0(t) \equiv 1.$$

- and $\Psi_{n-j}(\boldsymbol{A})$ such that

$$\boldsymbol{p}_k^H \Omega_l(\boldsymbol{A})\Psi_{n-j}(\boldsymbol{A})\boldsymbol{r}_0 = 0,\ k = 1,2,\ldots,s,\ l = 0,1,\ldots,j-1\ .$$

Multi-orthogonality

These latter relations can be interpreted as formal orthogonality relations for the polynomial Ψ_{n-j}.
Define s formal inner products on the space of polynomials as follows:

$$[\varphi, \psi]_k = \boldsymbol{p}_k^H \phi(\boldsymbol{A})\psi(\boldsymbol{A})\boldsymbol{r}_0, \quad k = 1, 2, \ldots, s \ .$$

Then

$$[\Omega_l, \Psi_{n-j}]_k = 0, \quad k = 1, 2, \ldots, s, \quad l = 0, 1, \ldots, j-1 \ ,$$

and this is equivalent to formal orthogonality of Ψ_{n-j} to all polynomials in $\mathbb{P}^{j-1}$, *with respect to the* s *inner products* $[\,.\,,\,.\,]_k$.

Finite termination of IDR(s)

The degree of the polynomial Ψ_{n-j} determines the finite termination of the algorithm. In order to generate a polynomial of degree N, we need $n = N + N/s$ MATVECS, where $j = N/s$ is the degree of Ω_j.

Hence IDR(s) terminates in at most $N + N/s$ iterations (= MATVECS) at the exact solution.

A 1D Convection-Diffusion Problem

The first example illustrates the finite termination behaviour of IDR(s).

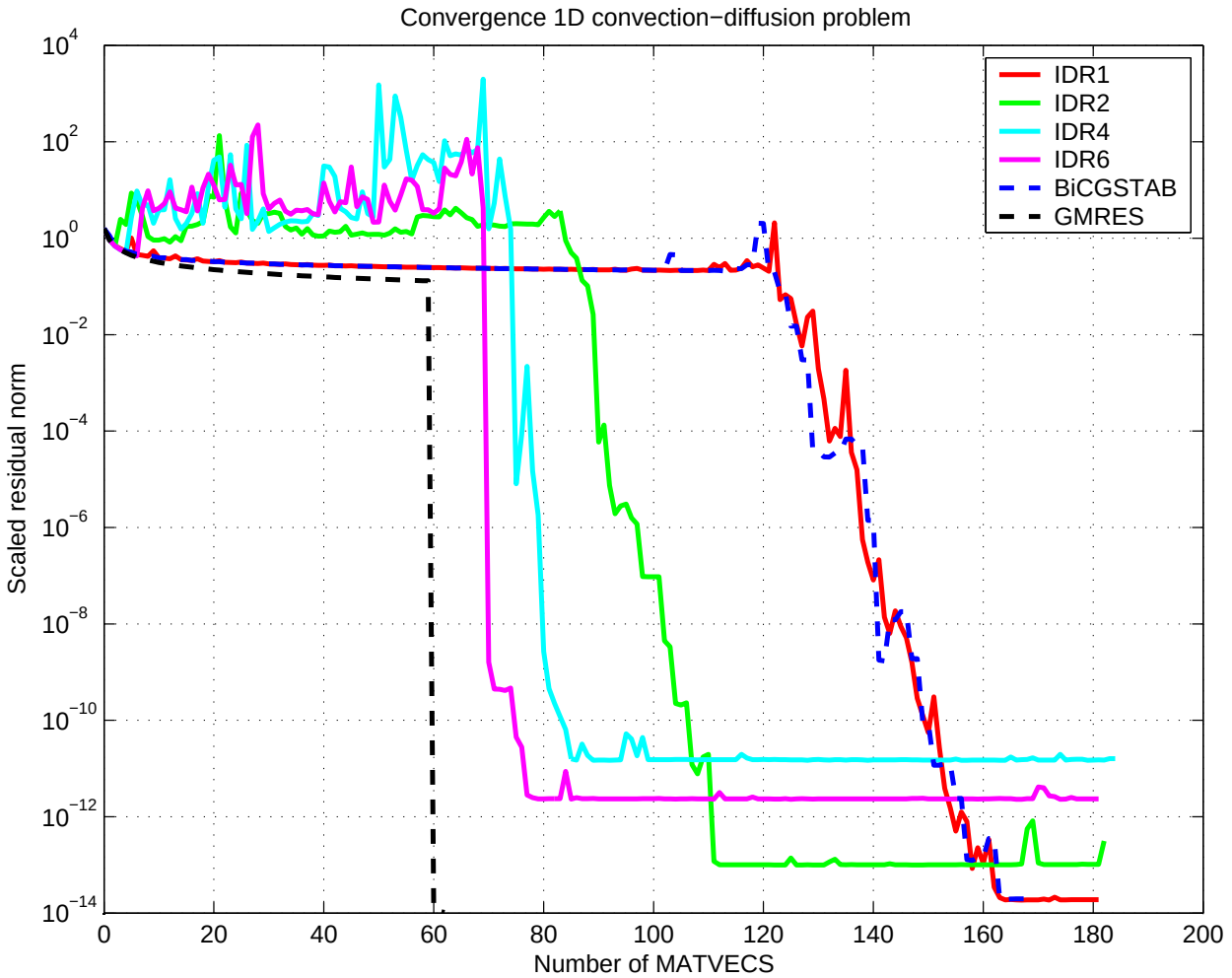
We consider the discretisation of

$$-\frac{d^2u}{dx^2} + w\frac{du}{dx} = 0 \quad x \in (0,1)$$

with $u(0) = u(1) = 1$ with the grid size $h = \frac{1}{61}$. The parameter w is such that $\frac{wh}{2} = 0.5$. The system consists of 60 equations.

$\boldsymbol{P}$ consists of $\boldsymbol{r}_0$ for comparison with Bi-CGSTAB, supplemented with $s-1$ random vectors.

Termination of IDR(s)



Observations

- Termination behaviour as predicted.
  - In theory after 120, 90, 75, and 70 MATVECS for s equal to 1, 2, 4, 6 resp.
  - In practice sharp drop of the residual norm around predicted termination point
- Convergence curves of IDR(1) and Bi-CGSTAB almost coincide
- Stagnation level of IDR(s) almost same as for Bi-CGSTAB. IDR(4) and IDR(6) slightly higher due to peaks in initial iterations.

Relation with other methods

As the polynomial analysis shows, IDR(s) is closely related to some Bi-CGSTAB methods:

- IDR(1) and Bi-CGSTAB yield the same residuals at the even steps.
- ML(k)BiCGSTAB (Yeung and Chan, SISC 1999) is closely related to IDR(s), but
  - Residual/approximate solution always different
  - More complicated than IDR(s), is based on polynomial approach.

Algorithmic variants

To develop an IDR-based method, there are three elements of choice.

- How to choose $\boldsymbol{P}$,

  $\rightarrow$ Orthogonalized random vectors

- How to choose ω_j,

  $\rightarrow$ To minimize the next residual norm
  plus modification by Sleijpen and van der Vorst aiming for accuracy

  $\rightarrow$ IDRstab (later this talk)

  $\rightarrow$ Precomputed, based on Ritzvalues (Simoncini and Szyld, SISC 2010)

- How to compute intermediate vectors in $\mathcal{G}_j$.

Exploiting bi-orthogonality

The prototype IDR(s) method that is described in the SISC 2008 paper is not always stable for large s ($s > 10$).
This can be overcome by exploit the property that $\mathcal{G}_j$ is a linear subspace. By making linear combination of vectors such that

- each vector $\boldsymbol{g}_i^j$ is made orthogonal to $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_{i-1}$;
- intermediate residuals $\boldsymbol{r}_i^j$ are made orthogonal to $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_i$

an algorithm can be derived that is

- More accurate for large s;
- Slightly cheaper, requires less vector updates.

Cost of the (new) algorithm

The number of operations per IDR(s) cycle is:

- $s+1$ MATVECS;
- s^2+s+2 inner products;
- $2s^2+2s+2$ vector updates. (Original IDR(s): $s^2+\frac{7}{2}s+\frac{5}{2}$)
- Inner products can be combined: only one synchronization per MATVEC.

New variant is accurate for large s

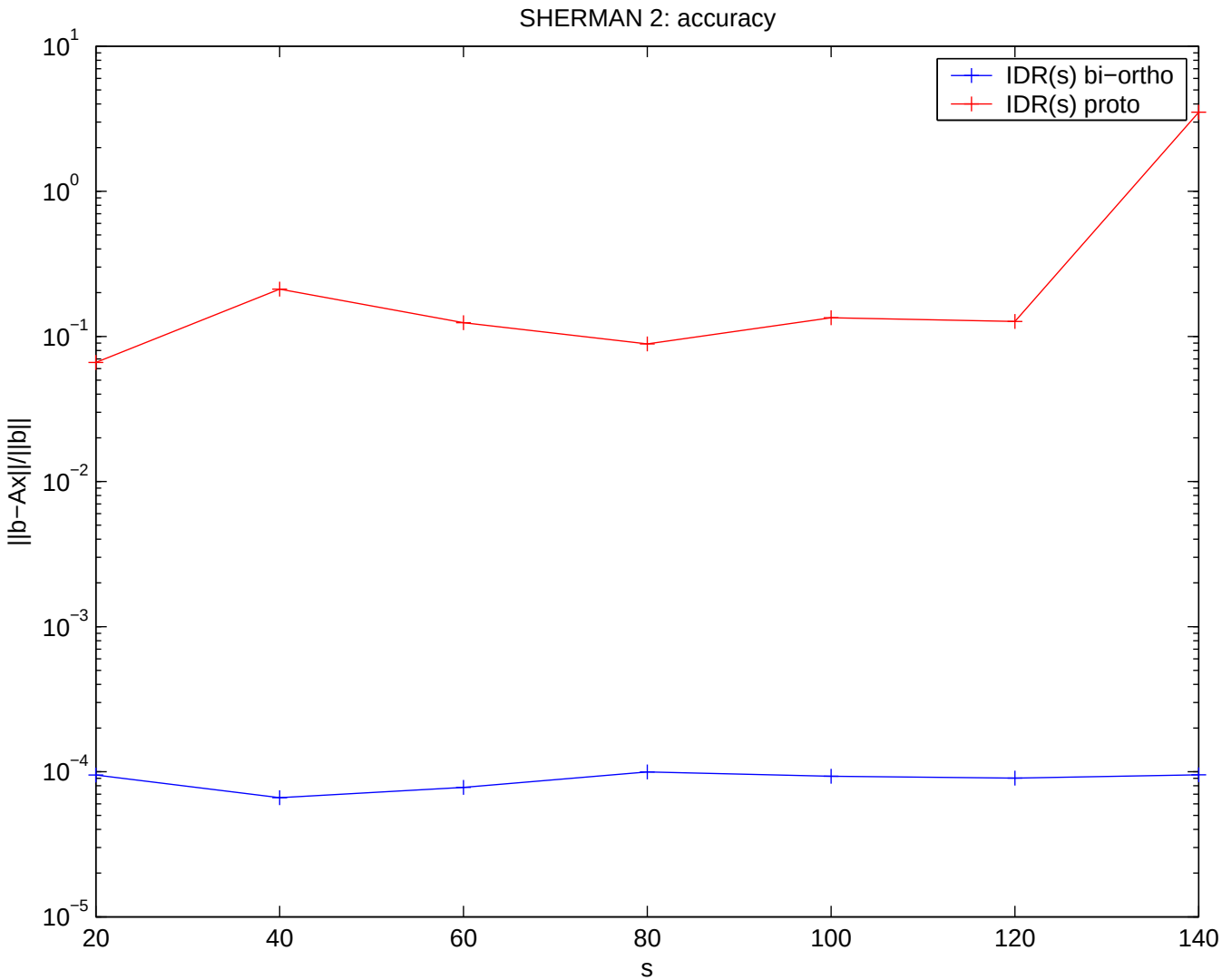
We illustrate the accuracy with the test problem SHERMAN 2 from the MATRIX MARKET collection

- Problem size 1080
- Estimated condition number 10^{12}
- Nonsymmetric and indefinite
- All standard short recurrence methods fail (QMR, Bi-CGSTAB, Bi-CG etc.)
- Nightmare problem

Convergence until $\|\boldsymbol{r}_n\|/\|\boldsymbol{b}\| < 10^{-4}$.

We plot the norm of the *true* residual after convergence.

How does the accuracy depend on s?



Almost skew symmetric problems

- Almost skew-symmetric matrices have eigenvalues with a large imaginary part.
- Bi-CGSTAB works poorly for such problems:
  - Method uses *linear* stabilization polynomials (minimal residual steps).
    Same is true for IDR(s)!
  - Cannot produce roots close to complex eigenvalues
  - Solution: use higher order stabilization polynomial:
    $\rightarrow$ BiCGstab2 (Gutknecht), BiCGstab(ℓ) (Sleijpen and Fokkema)
- Note: alternative solution (in IDR(s)): use complex P.

IDRStab

Can higher order stabilization polynomials be used with IDR(s)? Yes, by combining several dimension reductions steps It is possible to construct higher order stabilization polynomial.

Such a method was developed by Gerard Sleijpen and MvG. The name is **IDRStab**. (SISC 2010, to appear)

A similar method (GBiCGSTAB(s,l)) has been proposed by Tanio and Sugihara.

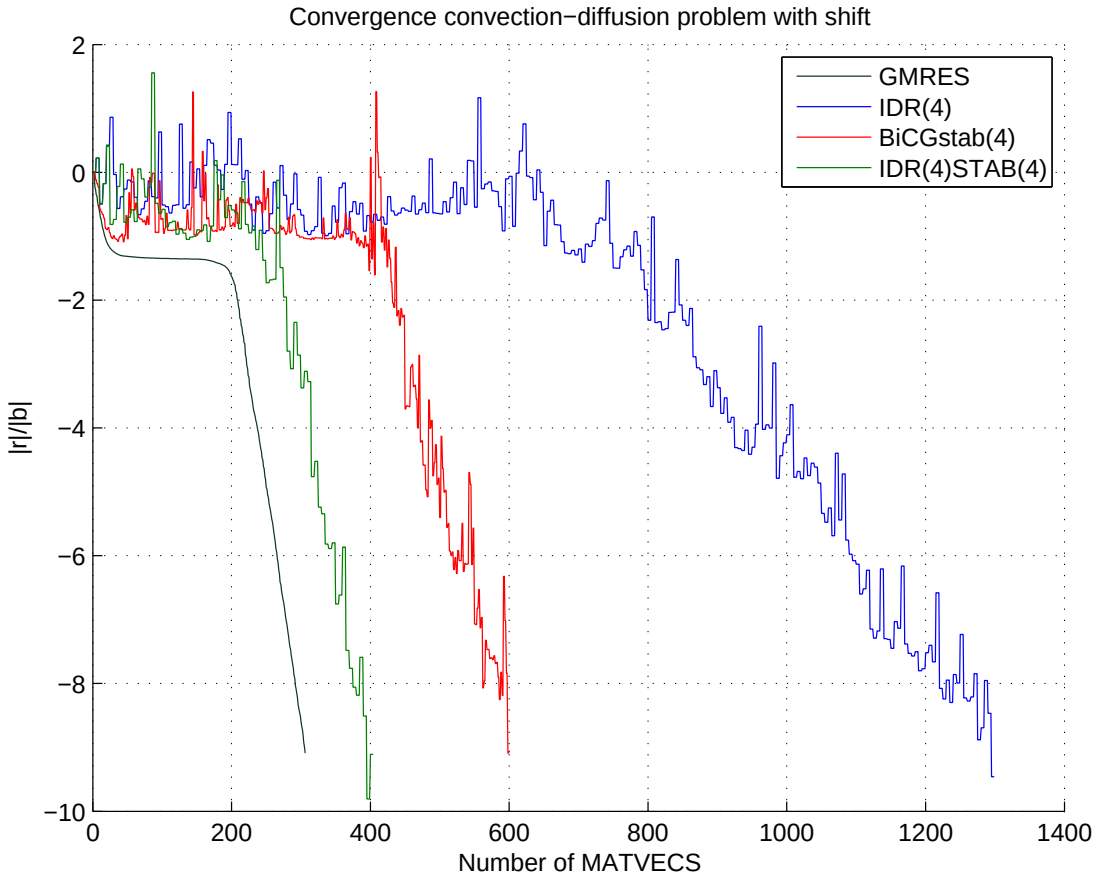
Example

Convection-diffusion problem with shift on unit square:

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

- Dirichlet conditions
- Discretised with the finite volume method on a 65×65 grid
- The right-hand side vector such that the solution is one.

Convergence of IDRStab



Other developments

We mention some other important developments:

- Collignon and MvG formulated IDR(s) variants with a minimal number of synchronisation points $\rightarrow$ talk at 14:00 by Tijmen Collignon.
- Fujino and co-workers study the choice of $\boldsymbol{P}^{\perp}$, and the combination of IDR(s) with basic iterative methods (Gauss-Seidel and Jacobi);
- Sonneveld made an analysis of the convergence behaviour of IDR(s);
- Zemke and Gutknecht investigate finite precision aspects of IDR(s) and IDR(s) for computing eigenvalues.

IDR for eigenvalues

Remember that

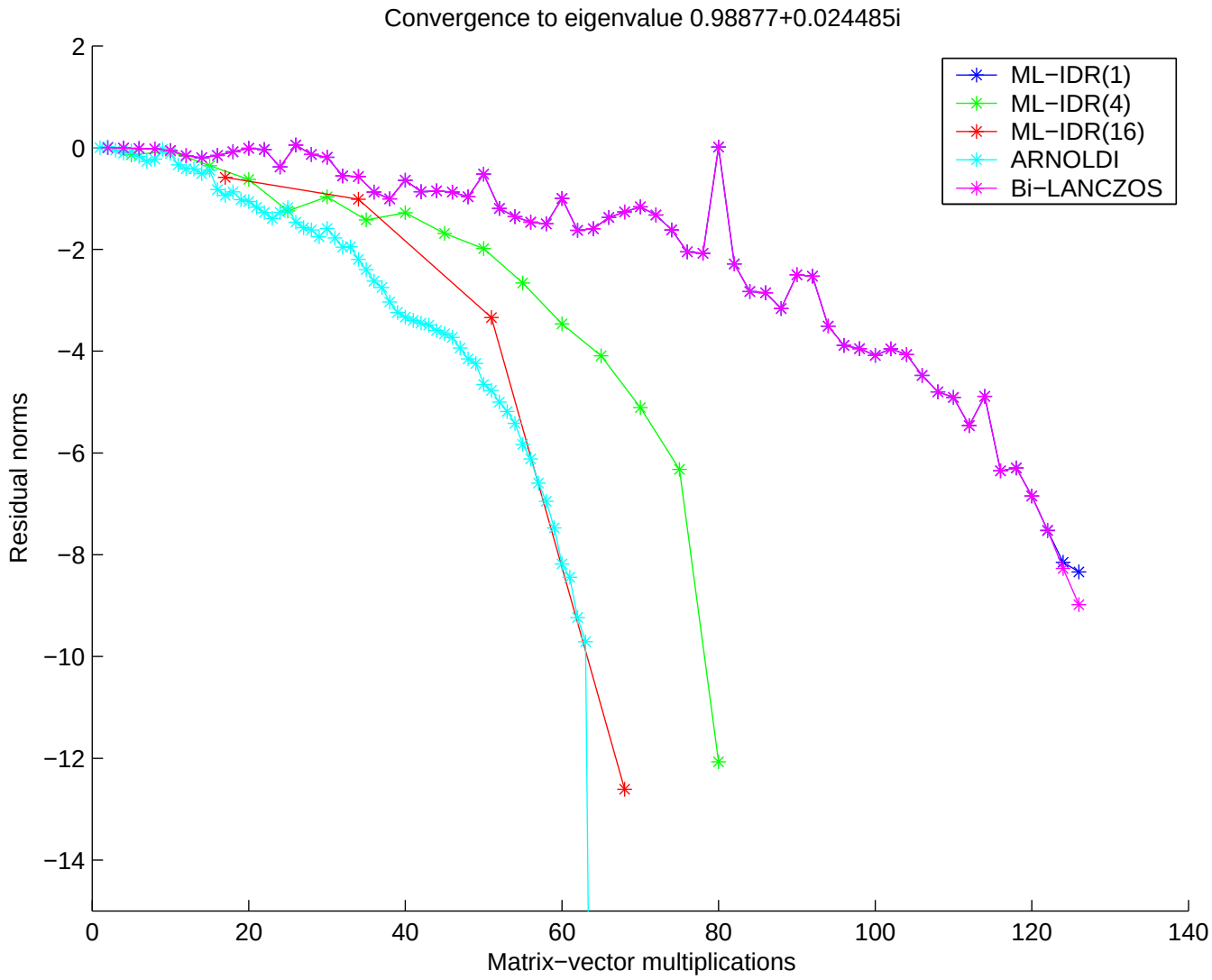
$$\boldsymbol{r}_n = \Omega_j(\boldsymbol{A})\Psi_{n-j}(\boldsymbol{A})\boldsymbol{r}_0 \quad .$$

The roots of the polynomial Ψ_{n-j} converge to the eigenvalues of $\boldsymbol{A}$ (are the Ritz values).
Hence it is possible to make and eigenvalue algorithm based on the IDR approach, see Gutknecht and Zemke.
In the next example we use an eigenvalue algorithm based on IDR(s) Bi-ortho.

Testproblem RANDOM

- Random matrix of order 64
- Compute 5 eigenvalues closest to 1
- Convergence if scaled residual norm < 10^{-8}
- Convergence plot for largest eigenvalue

Convergence RANDOM (residual norm)



Conclusions

- The IDR-theorem offers a flexible approach for the development of iterative solution algorithms.
- We have discussed several IDR(s) methods
- We have shown the potential of these methods with numerical examples.

More information: http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html

- Reports and papers,
- Matlab codes and test problems (about to be updated).