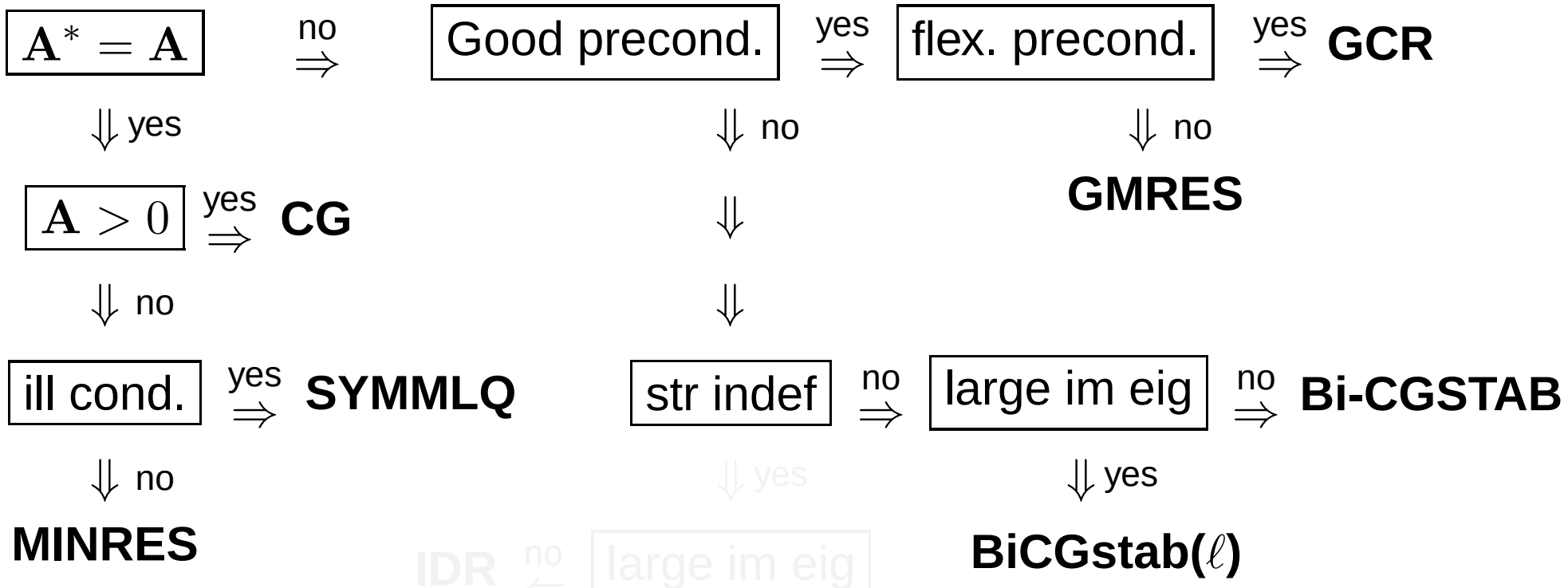
Numerical Linear algebra

Advanced topics

Martin van Gijzen and Gerard Sleijpen

November 28, 2016

Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, an overview



a **good** preconditioner is available

the preconditioner is flexible

$\mathbf{A} + \mathbf{A}^*$ is strongly indefinite

$\mathbf{A}$ has large imaginary eigenvalues

Iterative methods in Matlab

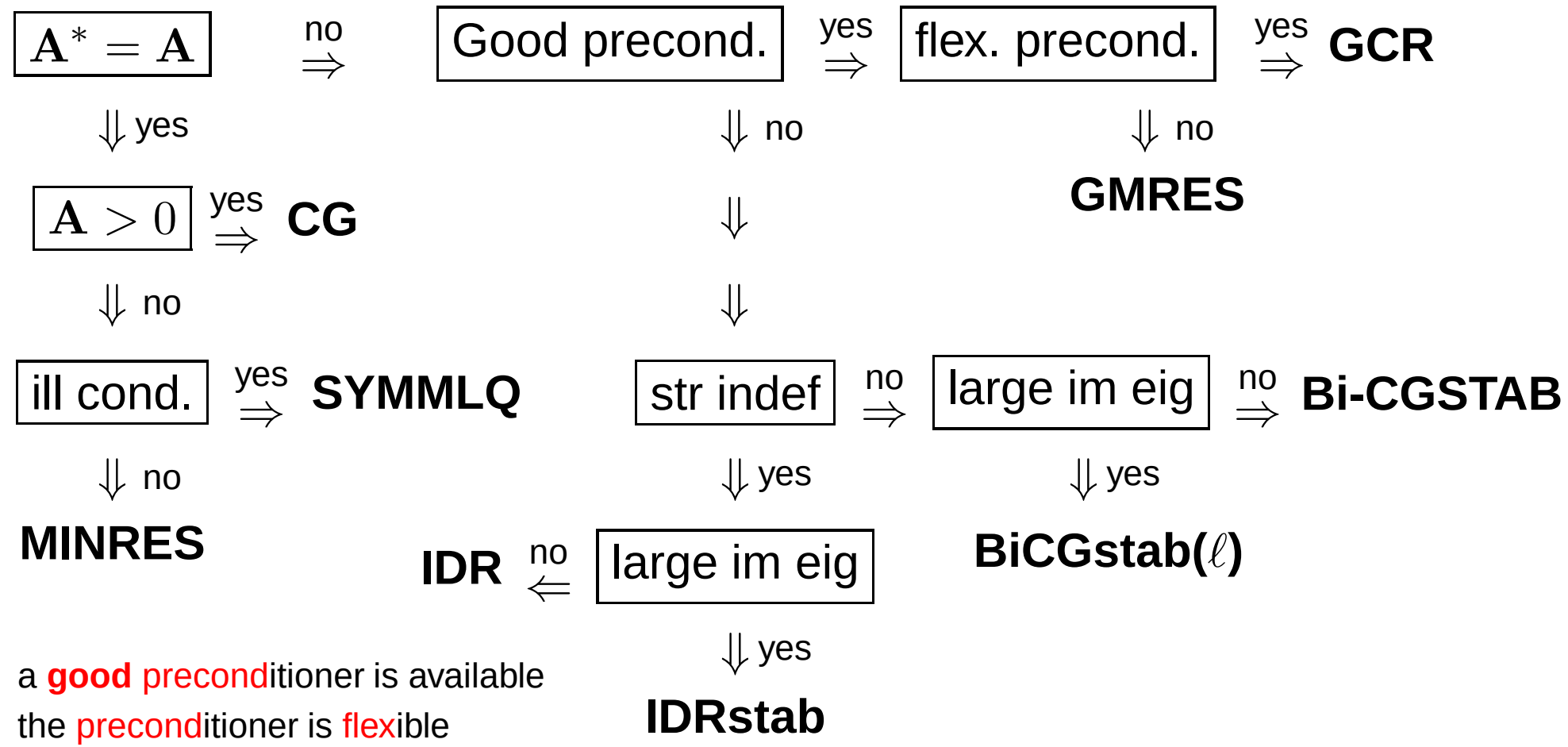
With “help PCG” Matlab shows the iterative methods for solving the square system $\mathbf{Ax} = \mathbf{b}$ that are coded in Matlab.

```
>> help pcg

    ...

    See also bicg, bicgstab, bicgstabl, cgs,
    gmres, lsqr, minres, qmr, symmlq, tfqmr,
    ichol, function_handle.
```

Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, an overview



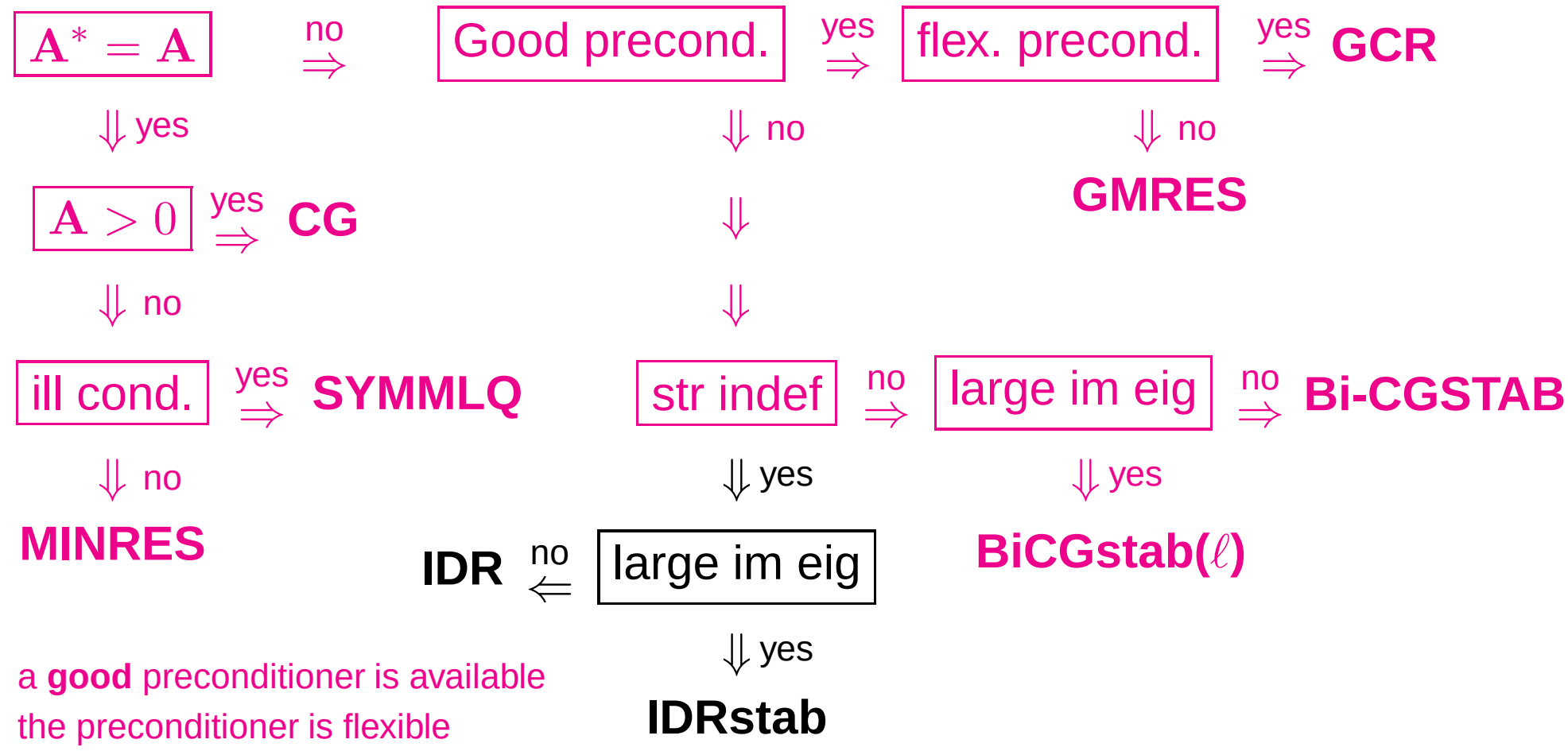
a **good** preconditioner is available

the preconditioner is flexible

$\mathbf{A} + \mathbf{A}^*$ is strongly indefinite

$\mathbf{A}$ has large imaginary eigenvalues

Solving $\mathrm{Ax} = \mathrm{b}$, an overview



a **good** preconditioner is available

the preconditioner is flexible

$\mathbf{A} + \mathbf{A}^*$ is strongly indefinite

$\mathbf{A}$ has large imaginary eigenvalues

Program Lecture 11

In this lecture $\mathbf{A}$ is an (complex) $n \times n$ matrix.

We are interested in solving $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$.

Advanced topics

- IDR(s): a ‘recent’ iterative solution method (for nonsymmetric $\mathbf{A}$)
- Preconditioning if $\mathbf{A}$ forms a KKT systems

Introduction

Today we will discuss two subjects of active research:

- IDR(s) is a ‘recent’ iterative method for nonsymmetric systems.
- KKT systems (or saddle point problems) play a role in many different areas and are notoriously difficult to solve.

Program Lecture 11

In this lecture $\mathbf{A}$ is an (complex) $n \times n$ matrix.

We are interested in solving $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$.

Advanced topics

- IDR(s): a 'recent' iterative solution method (for nonsymmetric $\mathbf{A}$)
- Preconditioning if $\mathbf{A}$ forms a KKT systems

Krylov subspace methods

Iterate:
- **Extract** $\mathbf{x}_k$ from $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$
- Stop if sufficiently accurate
- **Expand** $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ to $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$

Holy Grail

Krylov subspace method with the following properties
- Stable ('orthogonal' basis)
- 'Minimal' error
- Short recurrence

Krylov subspace methods

Iterate:
- **Extract** $\mathbf{x}_k$ from $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$
- Stop if sufficiently accurate
- **Expand** $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ to $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$

Holy Grail exists in ideal situations: if $\mathbf{A}$ is positive definite.

Example. CG for $\mathbf{A}$ positive definite.

Properties.
- $\mathbf{A}$-orthogonal basis
- minimal error w.r.t. the $\mathbf{A}$-norm
- per step: 1MV, 3 vector updates, 2 inner products

Krylov subspace methods

Iterate:
- **Extract** $\mathbf{x}_k$ from $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$
- Stop if sufficiently accurate
- **Expand** $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ to $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$

Holy Grail does not exist in general. For nonsymmetric systems it is not possible to combine optimal error reduction with short recurrences [Faber and Manteuffel, 1984].

Two different approaches have been taken:

- **GMRES approach.**

  Minimizes residual norm but uses long recursions.

- **Bi-CG approach.** Puts residuals (bi-)orthogonal.

  Uses short recursions, but no optimal reduction of an error norm.

Krylov subspace methods

Iterate:
- **Extract** $\mathbf{x}_k$ from $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$
- Stop if sufficiently accurate
- **Expand** $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ to $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$

Bi-CG approach:

Solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ and, at the same time,

solve a 'shadow' system $\mathbf{A}^* \, \widetilde{\mathbf{x}} = \widetilde{\mathbf{r}}_0$.

Extract $\mathbf{x}_k$ such that $\mathbf{r}_k \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$ (bi-orthogonal residuals).

It suffices to have $\mathbf{r}_k \perp \widetilde{\mathbf{r}}_{k-1}, \widetilde{\mathbf{r}}_{k-2}$ (short recurrences).

Expand with $\mathbf{r}_k$.

The Bi-CG approach

Bi-CG uses CG recursions but needs extra matvec with $\mathbf{A}^*$.

Idea of Sonneveld: use 'wasted' matvec in a more useful way.

Result:

hybrid Bi-CG methods, transpose-free, product Lanczos type:

$\mathbf{r}_k = q_k(\mathbf{A})\,\mathbf{r}_k^{Bi\text{-}CG}$ for some polynomial q_k of degree k.

Examples.

- CGS [Sonneveld, 1989]
- Bi-CGSTAB [Van der Vorst, 1992]
- BiCGstab(ℓ) [Sleijpen and Fokkema, 1993]
- Several other variants

IDR and IDR(s)

Sonneveld developed IDR in the 1970's. IDR is a finite termination Krylov method for solving nonsymmetric linear systems.

Analysis showed that IDR can be viewed as Bi-CG combined with local minimal residual steps.

This discovery led to the development of first CGS, and later of Bi-CGSTAB.

As a result of these developments the basic IDR idea was abandoned for the Bi-CG-approach.

Recently, it was realized that the IDR-approach was abandoned too soon and proposed a generalization of IDR: IDR(s).

The IDR approach

Generate residuals $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ that are in **nested** subspaces $\mathcal{G}_j$ of **decreasing** dimension.

Select an $n \times s$ matrix $\widetilde{\mathbf{R}}$, a so-called **IDR test matrix**.
Then the $\mathcal{G}_j$ are related by

$$\mathcal{G}_j = (\mu_j \mathbf{I} - \mathbf{A})(\mathcal{G}_{j-1} \cap \widetilde{\mathbf{R}}^{\perp}),$$

where

- $\widetilde{\mathbf{R}}^{\perp}$ is the subspace of vectors in $\mathbb{C}^n$ that are orthogonal to all columns $\widetilde{\mathbf{r}}_\ell$ $(\ell = 1, \ldots, s)$ of $\widetilde{\mathbf{R}}$,
- the μ_j are parameters in $\mathbb{C}$.

It can be proved that ultimately $\mathbf{r}_k \in \{\mathbf{0}\}$ (The IDR theorem).

The IDR theorem

Theorem. With $\widetilde{\mathbf{R}}$ an $n \times s$ matrix, (μ_j) a sequence of scalars, and $\mathcal{G}_0 \equiv \mathbb{C}^n$, define the sequence $(\mathcal{G}_j)$ of subspaces $\mathcal{G}_j$ by

$$\mathcal{G}'_j \equiv \mathcal{G}_j \cap \widetilde{\mathbf{R}}^{\perp}, \quad \mathcal{G}_{j+1} \equiv (\mu_{j+1}\mathbf{I} - \mathbf{A})\mathcal{G}'_j \quad (j = 0, 1, \ldots).$$

Then (i) the $\mathcal{G}_j$ are nested, i.e., $\mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ for all $j > 0$.

(ii) if $\widetilde{\mathbf{R}}^{\perp}$ does not contain an eigenvector of $\mathbf{A}$,

then $\mathcal{G}_{j+1} \subsetneqq \mathcal{G}_j$, unless $\mathcal{G}_j = \{\mathbf{0}\}$.

In particular,

(i′) $\mathbf{A}\mathcal{G}'_j \subset \mathcal{G}_j$.

Note that $\mathcal{G}_j = \frac{1}{\mu_j}\mathcal{G}_j = (\mathbf{I} - \frac{1}{\mu_j}\mathbf{A})\mathcal{G}'_{j-1}$ if $\mu_j \neq 0$.

The IDR theorem

Theorem. With $\widetilde{\mathbf{R}}$ an $n \times s$ matrix, (μ_j) a sequence of scalars, and $\mathcal{G}_0 \equiv \mathbb{C}^n$, define the sequence $(\mathcal{G}_j)$ of subspaces $\mathcal{G}_j$ by

$$\mathcal{G}'_j \equiv \mathcal{G}_j \cap \widetilde{\mathbf{R}}^{\perp}, \quad \mathcal{G}_{j+1} \equiv (\mu_{j+1}\mathbf{I} - \mathbf{A})\mathcal{G}'_j \quad (j = 0, 1, \ldots).$$

Then (i) the $\mathcal{G}_j$ are nested, i.e., $\mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ for all $j > 0$.

(ii) if $\widetilde{\mathbf{R}}^{\perp}$ does not contain an eigenvector of $\mathbf{A}$,

then $\mathcal{G}_{j+1} \subsetneqq \mathcal{G}_j$, unless $\mathcal{G}_j = \{\mathbf{0}\}$.

In particular,

(ii′) $\dim(\mathcal{G}_{j+1}) < \dim(\mathcal{G}_j)$ unless $\mathcal{G}_j = \{\mathbf{0}\}$.

Usually, $\dim(\mathcal{G}_{j+1}) = \dim(\mathcal{G}_j) - s$.

The IDR theorem

Theorem. With $\widetilde{\mathbf{R}}$ an $n \times s$ matrix, (μ_j) a sequence of scalars, and $\mathcal{G}_0 \equiv \mathbb{C}^n$, define the sequence $(\mathcal{G}_j)$ of subspaces $\mathcal{G}_j$ by

$$\mathcal{G}'_j \equiv \mathcal{G}_j \cap \widetilde{\mathbf{R}}^{\perp}, \quad \mathcal{G}_{j+1} \equiv (\mu_{j+1}\mathbf{I} - \mathbf{A})\mathcal{G}'_j \quad (j = 0, 1, \ldots).$$

Then (i) the $\mathcal{G}_j$ are nested, i.e., $\mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ for all $j > 0$.

(ii) if $\widetilde{\mathbf{R}}^{\perp}$ does not contain an eigenvector of $\mathbf{A}$, then $\mathcal{G}_{j+1} \subsetneqq \mathcal{G}_j$, unless $\mathcal{G}_j = \{\mathbf{0}\}$.

This explains naming: **Induced Dimension Reduction**.

IDR: constructs $\mathbf{r}_k$ in $\mathcal{G}_j \equiv (\mathbf{I} - \omega_j \mathbf{A})\mathcal{G}'_{j-1}$ with $\omega_j = \frac{1}{\mu_j}$.

The IDR theorem

Theorem. With $\widetilde{\mathbf{R}}$ an $n \times s$ matrix, (μ_j) a sequence of scalars, and $\mathcal{G}_0 \equiv \mathbb{C}^n$, define the sequence $(\mathcal{G}_j)$ of subspaces $\mathcal{G}_j$ by

$$\mathcal{G}'_j \equiv \mathcal{G}_j \cap \widetilde{\mathbf{R}}^{\perp}, \quad \mathcal{G}_{j+1} \equiv (\mu_{j+1}\mathbf{I} - \mathbf{A})\mathcal{G}'_j \quad (j = 0, 1, \ldots).$$

Then (i) the $\mathcal{G}_j$ are nested, i.e., $\mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ for all $j > 0$.

(ii) if $\widetilde{\mathbf{R}}^{\perp}$ does not contain an eigenvector of $\mathbf{A}$, then $\mathcal{G}_{j+1} \subsetneqq \mathcal{G}_j$, unless $\mathcal{G}_j = \{\mathbf{0}\}$.

Proof. See Exercise 11.1

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

How to compute the s columns in $\mathcal{G}_{j+1}$ of an $n \times s$ matrix $\mathbf{V}_{j+1}$?

Apply the above approach for "$\mathbf{r}_j \rightsquigarrow \mathbf{r}_{j+1}$" also to $\mathbf{V}_j$?

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

How to compute the s columns in $\mathcal{G}_{j+1}$ of an $n \times s$ matrix $\mathbf{V}_{j+1}$?

Apply the above approach for "$\mathbf{r}_j \rightsquigarrow \mathbf{r}_{j+1}$" also to $\mathbf{V}_j$?

$$\Pi_1 \mathbf{V} = \mathbf{0} \qquad \text{:-(}$$

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

Idee. $\mathbf{r}_{j+1} \in \mathcal{G}_j$, because $\mathcal{G}_{j+1} \subset \mathcal{G}_j$.

Repeat the above approach for "$\mathbf{r}_j \rightsquigarrow \mathbf{r}_{j+1}$" to construct

$$\mathbf{r}_{j+1}^{(0)} \equiv \mathbf{r}_{j+1},\ \mathbf{r}_{j+1}^{(1)},\ \mathbf{r}_{j+1}^{(2)}, \ldots \text{ in } \mathcal{G}_{j+1}.$$

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

Making an IDR algorithm

Note that $\mathbf{v} \in \widetilde{\mathbf{R}}^{\perp} \quad \Leftrightarrow \quad \widetilde{\mathbf{R}}^{*}\mathbf{v} = \mathbf{0}$.

Let $\mathbf{V} = \mathbf{V}_j$ be an $n \times s$ matrices with columns in $\mathcal{G}_j$.
With the $s \times s$ matrix $\sigma \equiv \widetilde{\mathbf{R}}^{*}\mathbf{V}$ and $\Pi_1 \equiv \mathbf{I} - \mathbf{V}\sigma^{-1}\widetilde{\mathbf{R}}^{*}$,

$$\mathbf{r}_j \in \mathcal{G}_j \quad \rightsquigarrow \quad \mathbf{r}'_j \equiv \Pi_1 \mathbf{r}_j \in \mathcal{G}'_j \quad \rightsquigarrow \quad \mathbf{r}_{j+1} \equiv \mathbf{r}'_j - \omega \mathbf{A}\mathbf{r}'_j \in \mathcal{G}_{j+1}$$

Note. Updating $\mathbf{x}_j$ with residual $\mathbf{r}_j$, requires $\mathbf{U}$ with $\mathbf{V} = \mathbf{A}\mathbf{U}$.

$\mathbf{V}$ can be computed from, for instance, residual differences

$$\mathbf{V}_{j+1} = [\mathbf{v}_1, \ldots, \mathbf{v}_s], \quad \text{where} \quad \mathbf{v}_i \equiv \Delta\mathbf{r}^{(i)}_{j+1} \equiv \mathbf{r}^{(i-1)}_{j+1} - \mathbf{r}^{(i)}_{j+1}.$$

$$\mathbf{U}_{j+1} = [\mathbf{u}_1, \ldots, \mathbf{u}_s], \quad \text{where} \quad \mathbf{u}_i \equiv \Delta\mathbf{x}^{(i)}_{j+1} \equiv \mathbf{x}^{(i)}_{j+1} - \mathbf{x}^{(i-1)}_{j+1}.$$

Note that $\mathbf{v}_i \in \mathcal{G}_j$ since both $\mathbf{r}^{(i-1)}_{j+1}$ and $\mathbf{r}^{(i)}_{j+1}$ in $\mathcal{G}_j$.

Making an IDR algorithm (2)

Note. On the previous transparency, we suggested to form an $n \times s$ matrix $\mathbf{V}_{j+1}$ in $\mathcal{G}_{j+1}$ **after** the formation of the $s+1$ residual vectors $\mathbf{r}_{j+1}^{(i)} \quad (i = 0, \ldots, s)$.

However, if we repeatedly replace the oldest column in $\mathbf{V}$ by the newest residual difference, then all of the columns of $\mathbf{V}$ are in $\mathcal{G}_j$ (some are in $\mathcal{G}_{j+1}$) and $\mathbf{V}$ can also be used in the projection Π_1 for forming vectors in $\mathcal{G}_j'$. After $s+1$ of these replacements, the resulting $\mathbf{V}$ is a matrix with all columns in $\mathcal{G}_{j+1}$.

The first approach requires more memory (the old $\mathbf{V}$ and $\mathbf{U}$ have to be stored unto all new vectors are available).

Both approaches can be shown to lead to the same $\mathbf{r}_{j+1}'$.

Prototype IDR(s) algorithm.

for $k = 0, 1, 2, \ldots$ do

$\sigma = \widetilde{\mathbf{R}}^* \mathbf{V}$, $\vec{\beta} = \widetilde{\mathbf{R}}^* \mathbf{r}$. Solve $\sigma \vec{\alpha} = \vec{\beta}$ for $\vec{\alpha}$

$\mathbf{v} = \mathbf{V}\,\vec{\alpha}$, $\quad \mathbf{u} = \mathbf{U}\,\vec{\alpha}$,

$\mathbf{r}' = \mathbf{r} - \mathbf{v}$, $\ \mathbf{s} = \mathbf{A}\mathbf{r}'$, $\quad \mathbf{x}' = \mathbf{x} + \mathbf{u}$,

if $k = 0 \bmod (s+1)$, $\quad \omega = (\mathbf{s}^*\mathbf{r}')/(\mathbf{s}^*\mathbf{s})$

$\mathbf{r} = \mathbf{r}' - \omega\,\mathbf{s}$, $\quad \mathbf{x} = \mathbf{x}' + \omega\,\mathbf{r}'$

$\mathbf{v} \leftarrow \mathbf{v} + \omega\,\mathbf{s}$, $\quad \mathbf{u} \leftarrow \mathbf{u} + \omega\,\mathbf{r}'$,

$\mathbf{V} \leftarrow [\widehat{\mathbf{V}}, \mathbf{v}]$, $\quad \mathbf{U} \leftarrow [\widehat{\mathbf{U}}, \mathbf{u}]$

if ($\|\mathbf{r}\| <$ tol), break

end do

Here, $\widehat{\mathbf{W}} \equiv [\mathbf{w}_2, \ldots, \mathbf{w}_s]$ if $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_s]$.

Initialising IDR.

$\widetilde{\mathbf{R}}$: Select $\widetilde{\mathbf{R}}$ (orthonormal) randomly. Then

i) 100% probability that $\widetilde{\mathbf{R}}^{\perp}$ does not contain an eigenvector of $\mathbf{A}$.

ii) appears to work well in practise.

To have $\widetilde{\mathbf{R}}$ orthonormal, take the $\mathbf{Q}$ of a QR-decomposition of an $n \times s$ random matrix.

$\mathbf{V}$: With $\mathbf{U}$ with columns spanning $\mathcal{K}_s(\mathbf{A}, \mathbf{r}_0)$, the IDR algorithm is also a Krylov subspace method:

$$\mathbf{r}_j^{(i)} \in \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) \cap \mathcal{G}_j \quad \text{for} \quad k \geq j(s+1), \quad i = 0, \ldots, s.$$

For instance, with $\mathbf{V} = \mathbf{U} = [\,]$,

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, and $\mathbf{x} = \mathbf{x}_0$,

start with s steps of the Local Minimal Residual method';

for $k = 1, \ldots, s$,

$\mathbf{u} = \mathbf{r}$, $\mathbf{v} = \mathbf{A}\mathbf{u}$

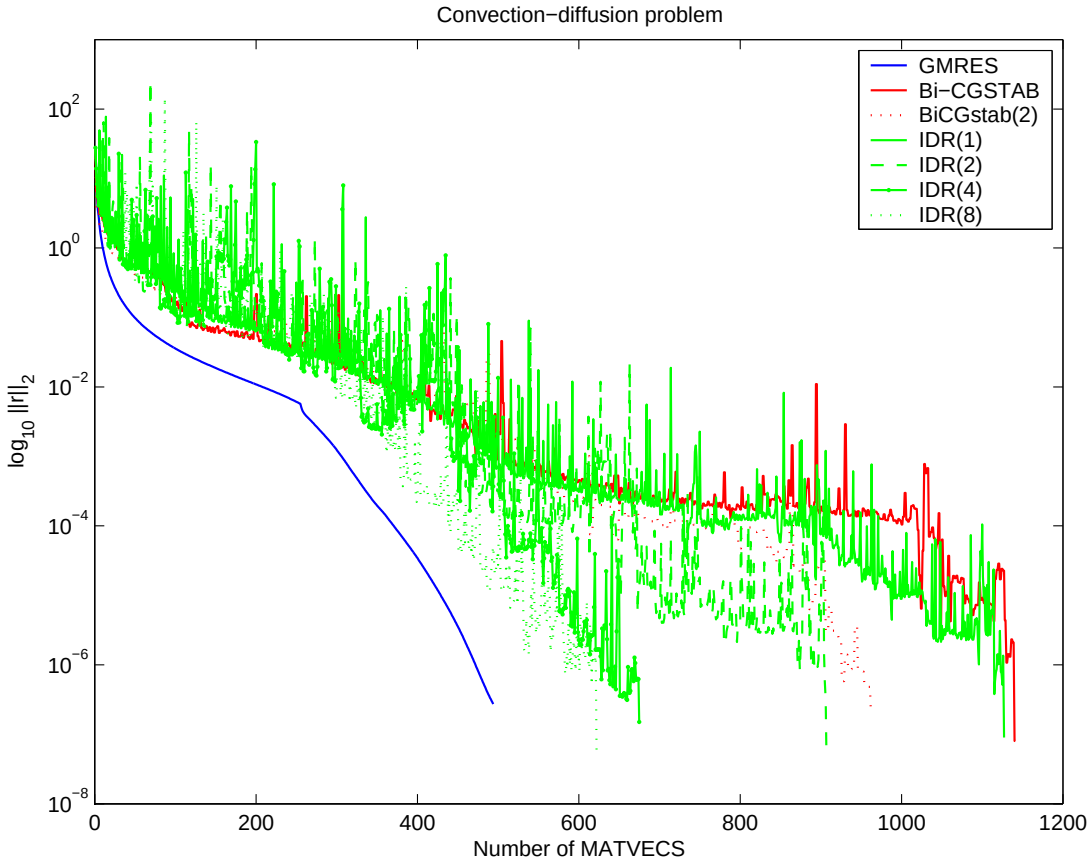
compute ω

$\mathbf{r} \leftarrow \mathbf{r} - \omega\mathbf{v}$, $\quad \mathbf{x} \leftarrow \mathbf{x} + \omega\mathbf{u}$

$\mathbf{V} \leftarrow [\mathbf{V}, \mathbf{v}]$, $\quad \mathbf{U} \leftarrow [\mathbf{U}, \mathbf{u}]$

Convergence of IDR(s)

The figure below shows typical convergence of Bi-CGSTAB, GMRES (optimal) and IDR(s).



Convergence of IDR(s) (2)

Typically, the number of MVs required by IDR(s) equals

$\frac{s+1}{s}$ times the number of MVs required by GMRES:

Theorem. $\frac{s+1}{s} \times$ # MVs GMRES = # MVs IDR(s) for $s \to \infty$.

Convergence of IDR(s) (2)

Typically, the number of MVs required by IDR(s) equals

$\frac{s+1}{s}$ times the number of MVs required by GMRES:

Theorem. $\frac{s+1}{s} \times$ # MVs GMRES = # MVs IDR(s) for $s \to \infty$.

Practise. $\infty = 8$.

IDR and Bi-CG

The IDR 'philosophy' is quite different from the Bi-CG approach:

- Bi-CG methods compute residuals in a sequence of growing subspaces: the Krylov subspace $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$ grows in every iteration (in dimension).
- In IDR the residuals are forced to be in decreasing subspaces $\mathcal{G}_j$.

However, the two approaches are mathematically closely related:

- Bi-CG computes the kth residual in $\mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)^\perp$, that is, residuals from a sequence of shrinking spaces.
- with a Krylov subspace start, IDR find residuals in $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

IDR and Bi-CG

The IDR ‘philosophy’ is quite different from the Bi-CG approach:

- Bi-CG methods compute residuals in a sequence of growing subspaces: the Krylov subspace $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{r}_0)$ grows in every iteration (in dimension).
- In IDR the residuals are forced to be in decreasing subspaces $\mathcal{G}_j$.

However, the two approaches are mathematically closely related:

- The link is closest with Bi-CGSTAB:
  IDR(1) and Bi-CGSTAB are (mathematically) equivalent.

IDR and block Krylov.

The **block Krylov subspace** $\mathcal{K}_j(\mathbf{A}^*, \widetilde{\mathbf{R}})$ is defined by

$$\mathcal{K}_j(\mathbf{A}^*, \widetilde{\mathbf{R}}) \equiv \left\{ \sum_{i=0}^{j-1} (\mathbf{A}^*)^i \, \widetilde{\mathbf{R}} \, \vec{\alpha}_i \mid \vec{\alpha}_i \in \mathbb{C}^s \right\}.$$

For any polynomial q, with $j \equiv \text{degree}(q)$,
define the **Sonneveld subspace** $\mathcal{G}(q, \mathbf{A}, \widetilde{\mathbf{R}})$ by

$$\mathcal{G}(q, \mathbf{A}, \widetilde{\mathbf{R}}) \equiv \{ q(\mathbf{A})\mathbf{v} \mid \mathbf{v} \perp \mathcal{K}_j(\mathbf{A}^*, \widetilde{\mathbf{R}}) \}.$$

Theorem. Let $q_0(\lambda) \equiv 1$, and, for $j = 1, 2, \ldots,$

$\omega_j \equiv \frac{1}{\mu_j}$, $q_j(\lambda) \equiv (1 - \omega_j \, \lambda) \, q_{j-1}(\lambda)$. Then, $\mathcal{G}_j = \mathcal{G}(q_j, \mathbf{A}, \widetilde{\mathbf{R}})$.

IDR and hybrid Bi-CG

Recall that $\mathbf{r}_k^{\text{Bi-CG}} \perp \mathcal{K}_k(\mathbf{A}^*, \widetilde{\mathbf{r}}_0)$

- The CGS residual can be written as the residual Bi-CG polynomial $p_k \equiv p_k^{\text{Bi-CG}}$ squared times the initial residual $\mathbf{r}_0$:

$$\mathbf{r}_k^{\text{CGS}} = p_k(\mathbf{A})p_k(\mathbf{A})\mathbf{r}_0 = p_k(\mathbf{A})\mathbf{r}_k^{\text{Bi-CG}} \in \mathcal{G}(p_k, \mathbf{A}, [\widetilde{\mathbf{r}}_0]).$$

- Bi-CGSTAB residual can be written as a 'stab' polynomial $q_k \equiv q_k^{\text{Bi-CGSTAB}}$ in $\mathbf{A}$ times the Bi-CG residual $\mathbf{r}_k^{\text{Bi-CG}}$:

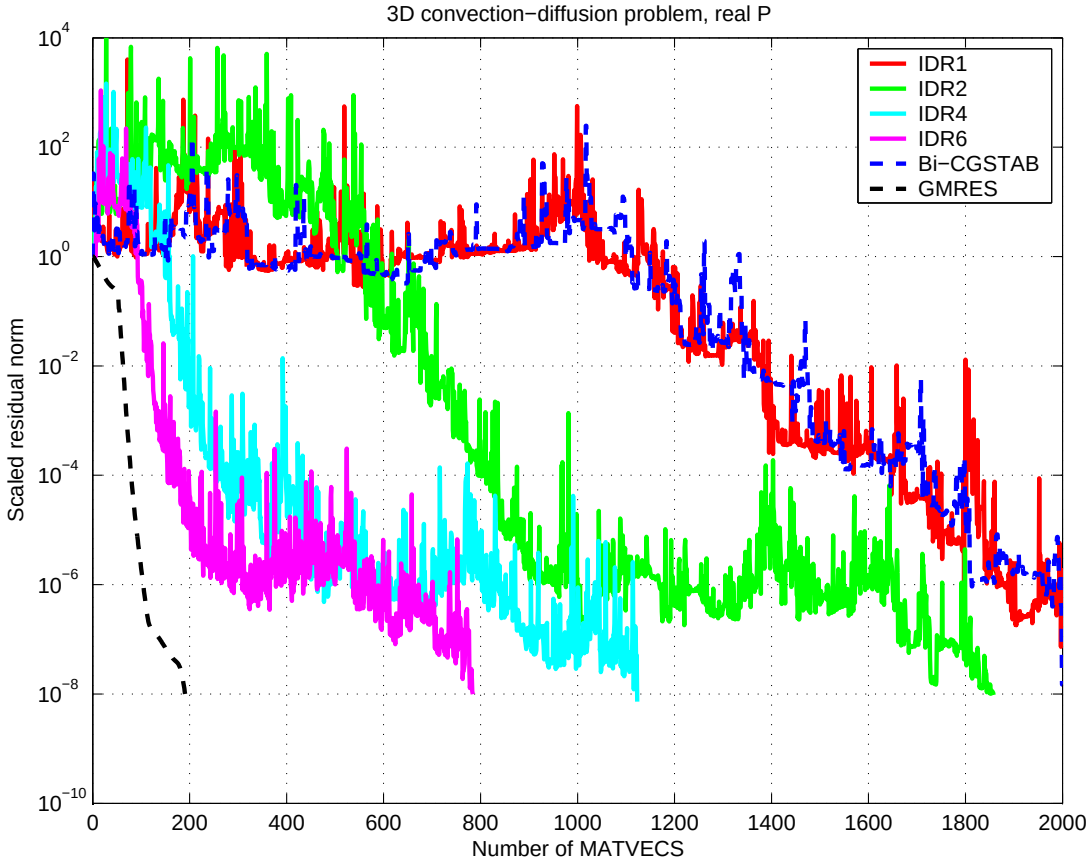
$$\mathbf{r}_k^{\text{Bi-CGSTAB}} = q_k(\mathbf{A})\mathbf{r}_k^{\text{Bi-CG}}.$$

Recall that in Bi-CGSTAB, $q_k(\lambda) = (1 - \omega_k\lambda)\, q_{k-1}(\lambda)$.

Hence, $\mathbf{r}_k^{\text{Bi-CGSTAB}} = \mathbf{r}_k^{\text{IDR(1)}} \in \mathcal{G}_k = \mathcal{G}(q_k, \mathbf{A}, [\widetilde{\mathbf{r}}_0])$.

Convergence of IDR(s) (3)

IDR(s) does not work well on strongly a-symmetric matrices:

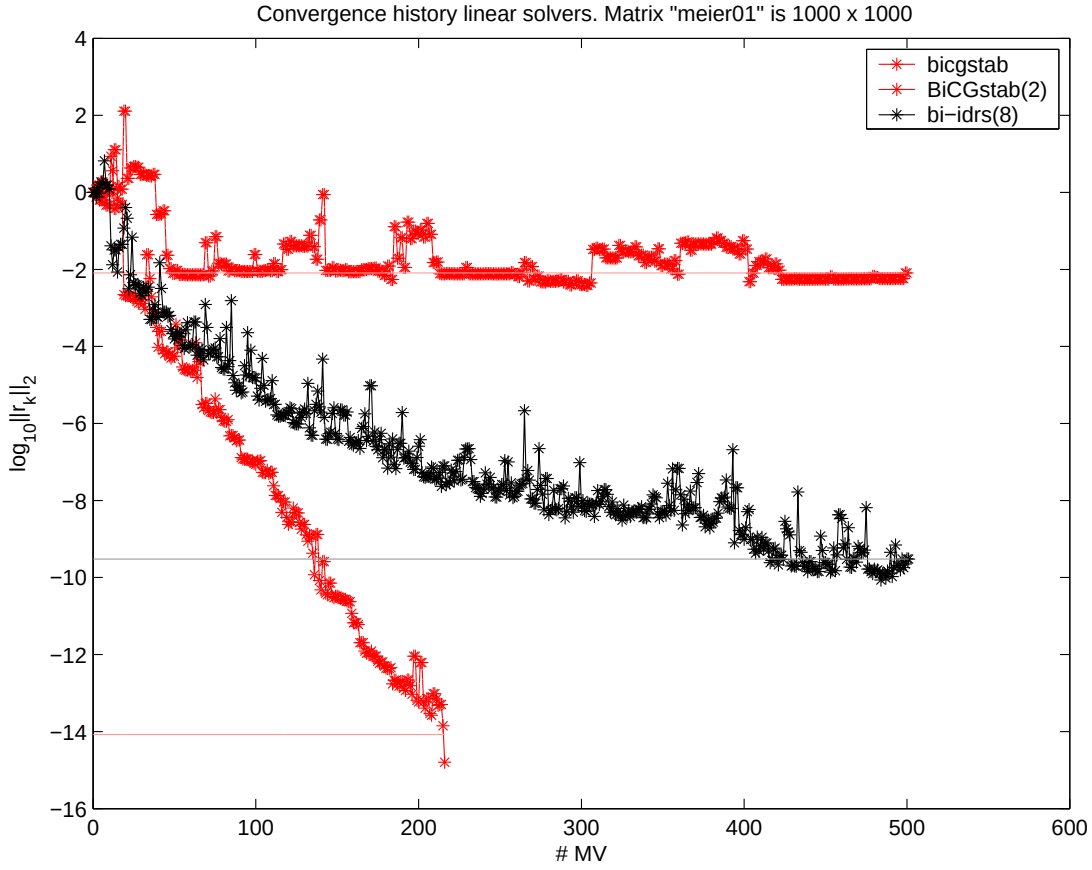


$$u_{xx} + u_{yy} + u_{zz} + 1000u_x = f.$$

Discretized with $50 \times 50 \times 50$ volumes. No preconditioning.

Convergence of IDR(s) (4)

IDR(s) does not work well on strongly a-symmetric matrices:



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

Discretized with $10 \times 10 \times 10$ volumes. No preconditioning.

IDRstab

For almost skew matrices the ‘stabilisation’ polynomial,

$$q_j(\lambda) \equiv (1 - \omega_j \lambda)(1 - \omega_{j-1} \lambda) \cdots (1 - \omega_1 \lambda).$$

based on linear factors does not work well (should have complex zeros). An alternative is to use stabilisation polynomials with higher order factors (as in BiCGstab(ℓ)).

The resulting method is called **IDRstab**,
with corresponding algorithm `IDR(s)stab(`ℓ`)`.

IDRstab

For almost skew matrices the ‘stabilisation’ polynomial,

$$q_j(\lambda) \equiv (1 - \omega_j \lambda)(1 - \omega_{j-1}\lambda) \cdots (1 - \omega_1 \lambda).$$

based on linear factors does not work well (should have complex zeros). An alternative is to use stabilisation polynomials with higher order factors (as in BiCGstab(ℓ)).

The resulting method is called **IDRstab**,
with corresponding algorithm `IDR(s)stab(`ℓ`)`.

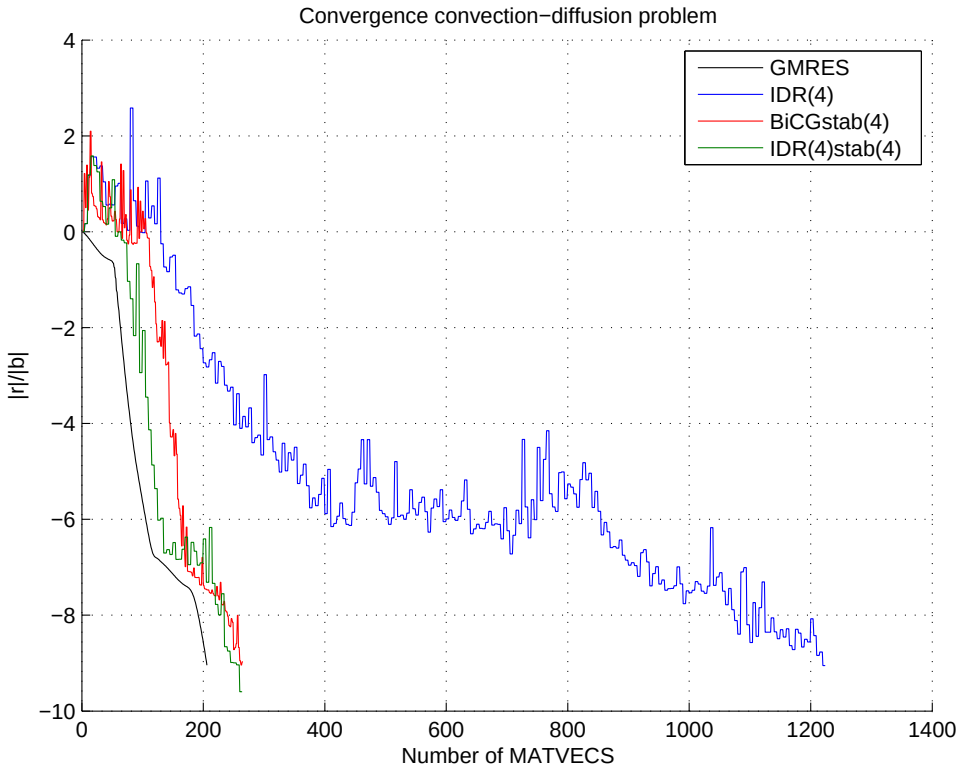
The IDR theorem. For each j and polynomials t_j of exact degree ℓ, define

$$\mathcal{G}_j' \equiv \mathcal{G}_j \cap \mathcal{K}_\ell(\mathbf{A}^*, \widetilde{\mathbf{R}})^\perp, \quad \mathcal{G}_{j+1} \equiv t_j(\mathbf{A})\, \mathcal{G}_j'.$$

Then, $\mathcal{G}_{j+1} \subsetneqq \mathcal{G}_j, \ldots$. Usually, $\dim(\mathcal{G}_{j+1}) = \dim(\mathcal{G}_j) - s\ell$.

Convergence IDRstab

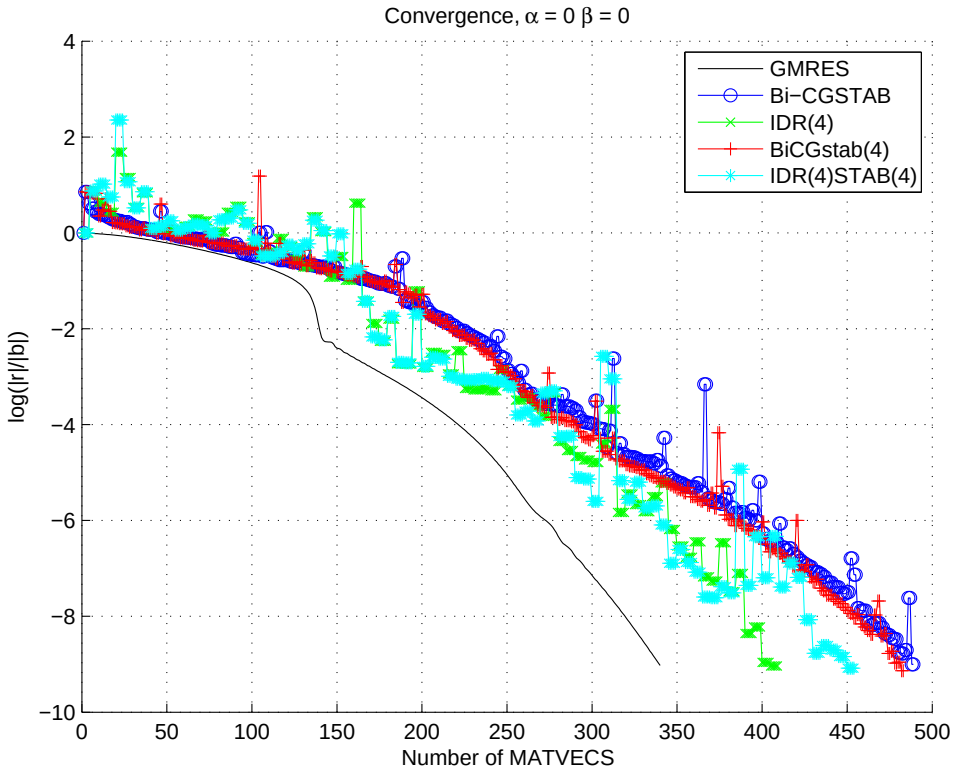
The picture below shows the convergence for the same problem



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

Discretized with $50 \times 50 \times 50$ volumes. No preconditioning.

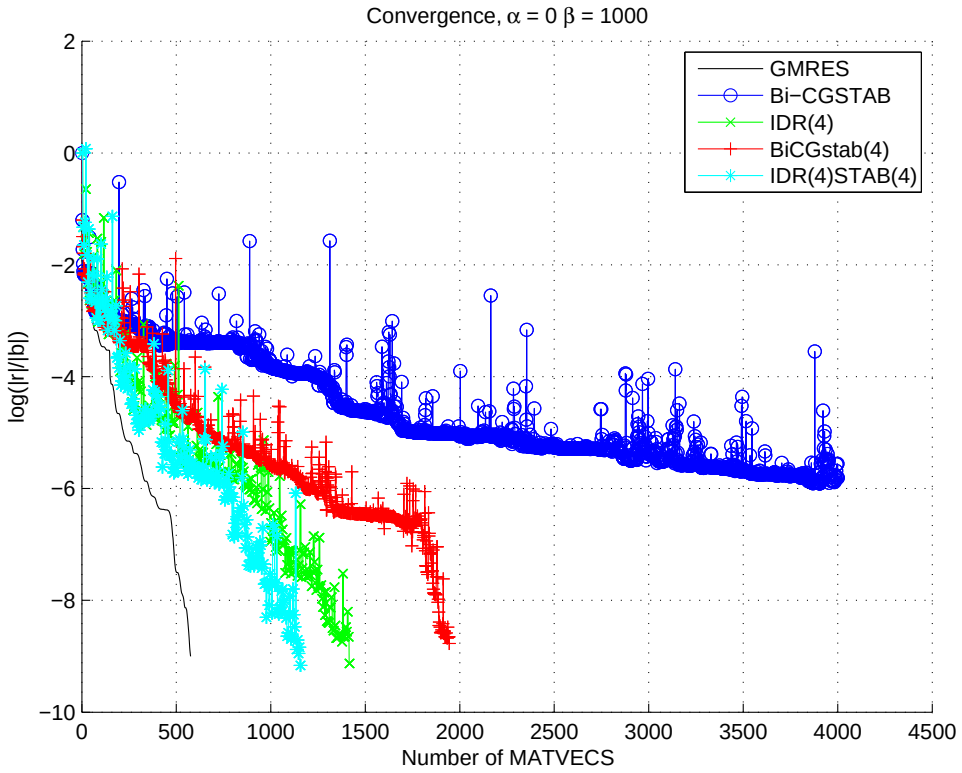
Convergence IDRstab (2)



$$-u_{xx} - u_{yy} + (\alpha/\sqrt{2})\,(u_x + u_y) - \beta u = f$$

on unit-square, homogenuous Dirichlet BC, f st solution $u = xy(1-x)(1-y)$, (201×201) finite differences. No preconditioning.

Convergence IDRstab (3)

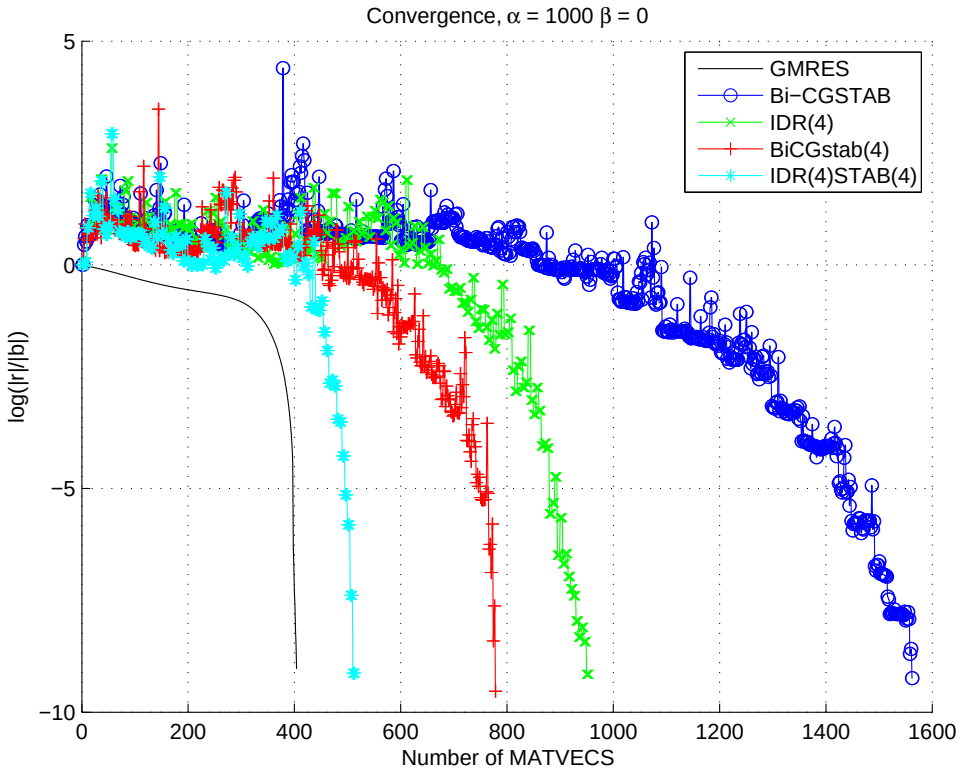


$$-u_{xx} - u_{yy} + (\alpha/\sqrt{2})\,(u_x + u_y) - \beta u = f$$

on unit-square, homogenuous Dirichlet BC, f st solution $u = xy(1-x)(1-y)$,

(201×201) finite differences. No preconditioning.

Convergence IDRstab (4)

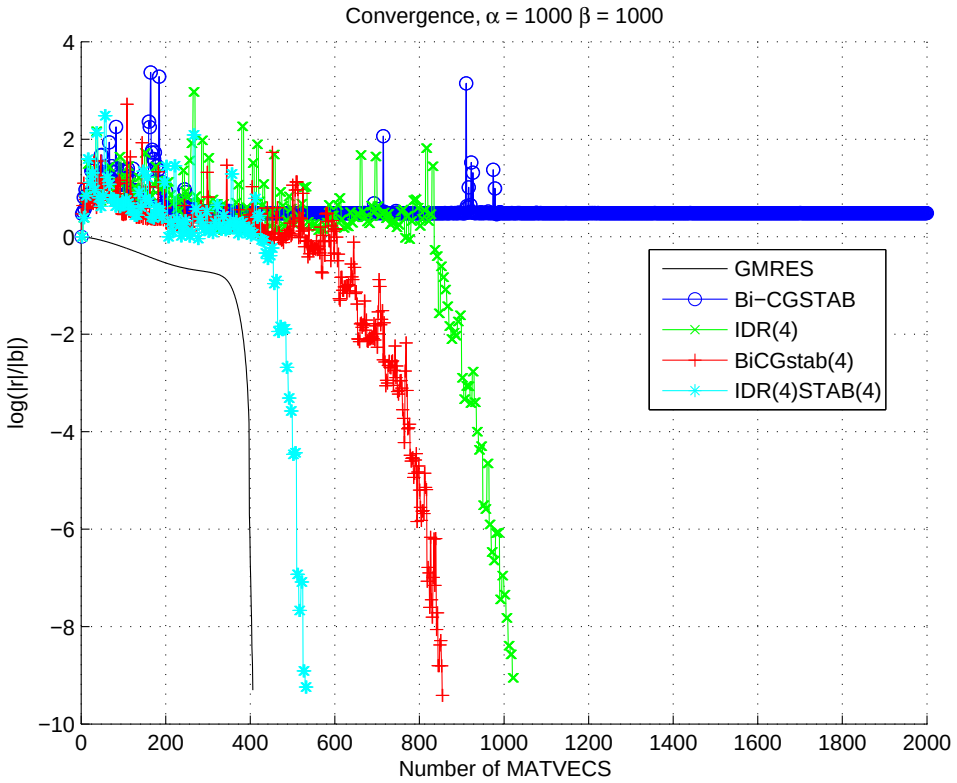


$$-u_{xx} - u_{yy} + (\alpha/\sqrt{2})\,(u_x + u_y) - \beta u = f$$

on unit-square, homogenuous Dirichlet BC, f st solution $u = xy(1-x)(1-y)$,

(201×201) finite differences. No preconditioning.

Convergence IDRstab (5)



$$-u_{xx} - u_{yy} + (\alpha/\sqrt{2})\,(u_x + u_y) - \beta u = f$$

on unit-square, homogenuous Dirichlet BC, f st solution $u = xy(1-x)(1-y)$,

(201×201) finite differences. No preconditioning.

Program Lecture 11

In this lecture $\mathbf{A}$ is an (complex) $n \times n$ matrix.
We are interested in solving $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$.

Advanced topics

- IDR(s): a 'recent' iterative solution method (for nonsymmetric $\mathbf{A}$)
- Preconditioning if $\mathbf{A}$ forms a KKT systems

KKT or saddle point systems

Karush-Kuhn-Tucker systems frequently arise in optimisation.

KKT systems are also know as **saddle-point systems**.

The matrix $\mathbf{A}$ of a KKT system has the following block structure

$$\mathbf{A} = \begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{B} & -\mathbf{C} \end{bmatrix}$$

with $\mathbf{H}$ and $\mathbf{C}$ symmetric positive definite matrices:

$\mathbf{A}$ is symmetric.

Systems with such a matrix arise in many other applications, for example in CFD (Stokes problem) and in structural engineering.

The matrix $\mathbf{C}$ is often a stabilisation term.

Optimisation

Consider the **Quadratic Program**

$$\min(\tfrac{1}{2}\mathbf{x}^*\mathbf{H}\mathbf{x} + \mathbf{q}^*\mathbf{x}) \quad \text{st} \quad \mathbf{B}\mathbf{x} = \mathbf{b},$$

i.e., we minimise over all $\mathbf{x} \in \mathbb{C}^n$ for which $\mathbf{B}\mathbf{x} = \mathbf{b}$.

Here, $\mathbf{H}$ is a $n \times n$ positive definite matrix,

$\mathbf{B}$ is $m \times n$ with $m \leq n$, $\mathbf{q}$ and $\mathbf{b}$ are given vectors.

At the solution $\mathbf{x}_\star$, we have the **KKT conditions**:

$$\mathbf{H}\mathbf{x}_\star + \mathbf{B}^*\lambda_\star + \mathbf{q} = 0 \quad \& \quad \mathbf{B}\mathbf{x}_\star = \mathbf{b}.$$

Here, $\lambda_\star$ is an m-vector of Lagrange multipliers.

This leads to a KKT system with $\mathbf{C} = \mathbf{0}$.

Computational Fluid Dynamics

$\Delta \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the **Laplacian**, $\nabla \equiv (\frac{\partial}{\partial x}, \frac{\partial}{\partial x})^T$ is the **gradient**, $\nabla^* \equiv (\frac{\partial}{\partial x}, \frac{\partial}{\partial x})$ is the **divergence**. All operators here in 2-dim.

The 2-d **Stokes equation**

$$\begin{bmatrix} -\mu\Delta & \nabla \\ \nabla^* & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ p \end{bmatrix} \begin{bmatrix} -\vec{f} \\ 0 \end{bmatrix}$$

(& BC) governs the motion of a slow viscous flow: $\vec{u} = (u, v)^T$ is the velocity field, p the pressure, $\vec{f}$ is a known external force field (as gravitation, etc.).

Discretisation leads to a KKT system with $\mathbf{C} = \mathbf{0}$.

Tykhonov regularisation

The Tychonov regularisation problem, here with $\mathbf{A}$ $n \times k$,

$$\text{argmin}\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \tau^2 \|\mathbf{x}\|_2^2 \mid \mathbf{x} \in \mathbb{C}^k\}$$

is equivalent to the KKT system

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^* & -\tau^2 \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{A}^* \mathbf{b} \end{bmatrix}$$

Remarks. Here, $\mathbf{H} = \mathbf{I}$, $\mathbf{B} = \mathbf{A}^*$, $\mathbf{C} = \tau^2 \mathbf{I}$.

Generally, $\mathcal{C}_2(\mathbf{H})$ will be large.

Tykhonov regularisation

The Tychonov regularisation problem, here with $\mathbf{A}$ $n \times k$,

$$\text{argmin}\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \tau^2 \|\mathbf{x}\|_2^2 \mid \mathbf{x} \in \mathbb{C}^k\}$$

is equivalent to the KKT system

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^* & -\tau^2 \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{A}^* \mathbf{b} \end{bmatrix}$$

Remarks. In Tomography, noise in the solution is large in directions of singular vectors with small singular values and are to be supressed, while in CFD these singular vectors correspond to ‘smooth’ components of the solution and have to be resolved.

Tykhonov regularisation

The Tychonov regularisation problem, here with $\mathbf{A}$ $n \times k$,

$$\text{argmin}\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \tau^2 \|\mathbf{x}\|_2^2 \mid \mathbf{x} \in \mathbb{C}^k\}$$

is equivalent to the KKT system

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^* & -\tau^2 \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{A}^* \mathbf{b} \end{bmatrix}$$

Remarks. The term $-\tau^2 \mathbf{I}$ is added to enhance stability of the problem.

Preconditioning a KKT-system (1)

Many preconditioners have been developed that make use of the fact that of the system matrix $\mathbf{A}$ a **block LU-decomposition** can be made:

$$\begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O}^* \\ \mathbf{B}\mathbf{H}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{O} & -\mathbf{M}_S \end{bmatrix} .$$

Here, $\mathbf{M}_S \equiv \mathbf{B}\mathbf{H}^{-1}\mathbf{B}^* + \mathbf{C}$.

$-\mathbf{M}_S$ is the so-called **Schur complement** (of $\mathbf{H}$ in $\mathbf{A}$).

Preconditioning a KKT-system (1)

Many preconditioners have been developed that make use of the fact that of the system matrix $\mathbf{A}$ a **block LU-decomposition** can be made:

$$\begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O}^* \\ \mathbf{B}\mathbf{H}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{O} & -\mathbf{M}_S \end{bmatrix} .$$

Here, $\mathbf{M}_S \equiv \mathbf{B}\mathbf{H}^{-1}\mathbf{B}^* + \mathbf{C}$.

$-\mathbf{M}_S$ is the so-called **Schur complement** (of $\mathbf{H}$ in $\mathbf{A}$).

Note that **multiplication** by $\mathbf{M}_S$ (as required in a Krylov method for solving $\mathbf{M}\mathbf{s} = \mathbf{r}$ for $\mathbf{s}$) involves **solving** a system for $\mathbf{H}$.

May require **Nesting** methods.

Preconditioning a KKT-system (2)

Idea (e.g., Elman, Silvester, Wathen): as (right) preconditioner, take

$$\mathbf{P} \equiv \begin{bmatrix} \mathbf{H} & \mathbf{B}^* \\ \mathbf{O} & -\mathbf{M}_S \end{bmatrix}, \quad \mathbf{A}\mathbf{P}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{O}^* \\ \mathbf{B}\mathbf{H}^{-1} & \mathbf{I} \end{bmatrix}$$

has only eigenvalue 1 and GMRES is ready in 2 iterations.

BUT

- The preconditioner $\mathbf{P}$ is nonsymmetric (and the eigensystem is extremely sensitive to perturbations).
- The Schur complement is too expensive to compute and has to be approximated
- resulting in a 'perturbed' $\mathbf{P}$.

Preconditioning a KKT-system (3)

An SPD block-diagonal preconditioner:

$$\mathbf{P} \equiv \begin{bmatrix} \mathbf{H} & \mathbf{O}^* \\ \mathbf{O} & \mathbf{M}_S \end{bmatrix}$$

Can be used with MINRES (short recurrences).

If $\mathbf{C} = \mathbf{0},$ then the preconditioned matrix has three distinct eigenvalues $\Rightarrow$ MINRES needs three iterations.

- The main question is again how to make a cheap approximation to the Schur complement.
- The eigensystem is well conditioned (less sensitive to perturbations).

Concluding remarks

In the MATLAB assignment you will apply IDR(s) to KKT systems from optimization.

You will:

- Construct an efficient KKT preconditioner
- Compare preconditioned IDR(s) with several other iterative methods