Università di Bologna

Recent developments in Krylov Subspace Methods for Scientific Computations

V. Simoncini

Dipartimento di Matematica and CIRSA, Bologna

valeria@dm.unibo.it

The Problem

$$Ax = b \qquad \text{or} \quad AX = B, \quad B = [b_1, \ldots, b_s]$$

$A \in \mathbb{C}^{n \times n}$, B full column rank, $s \ll n$

- A large and sparse
- A large and structured: blocks, banded, ...
- A functional: $A = CS^{-1}D$, preconditioned, integral, ...
- ....

The Problem

$$Ax = b \qquad \text{or} \quad AX = B, \quad B = [b_1, \ldots, b_s]$$

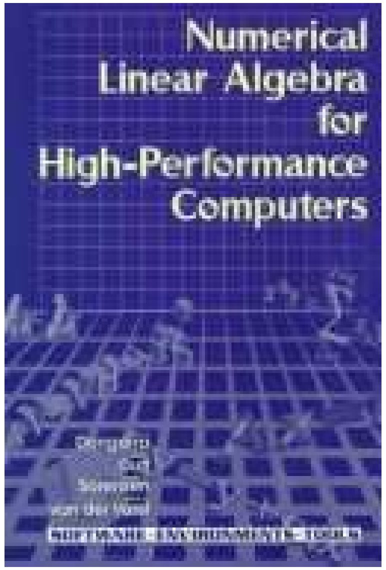
$A \in \mathbb{C}^{n \times n}$, B full column rank, $s \ll n$

- A large and sparse
- A large and structured: blocks, banded, ...
- A functional: $A = CS^{-1}D$, preconditioned, integral, ...
- ....

The solution approach. Generate sequence of approximate solutions:

$$\{x_0, x_1, x_2, \ldots\}, \qquad x_k \to_{k \to \infty} x$$

An excellent starting point



Comprehensive treatment of iterative methods and related computational aspects

Relevant Bibliographic Pointer

V. Simoncini and D. B. Szyld

Recent developments in Krylov Subspace Methods for linear systems

Numerical Linear Algebra with Appl., v. 14, n.1 (2007), pp.1-59.

The ideal case: $A = A^*$ Hermitian, A positive definite

Classical Conjugate Gradient:

Given x_0. Set $r_0 = b - Ax_0$, $p_0 = r_0$

for $i = 0, 1, \ldots$

$$\alpha_i = \frac{r_i^* r_i}{p_i^* A p_i}$$

$$x_{i+1} = x_i + p_i \alpha_i$$

$$r_{i+1} = r_i - A p_i \alpha_i$$

$$\beta_{i+1} = \frac{r_{i+1}^* A p_i}{p_i^* A p_i}$$

$$p_{i+1} = r_i + p_i \beta_{i+1}$$

end

At each iteration: 1 `Mxv`, 3 `-axpys`, 2 `-dots`

The Conjugate Gradient. Implementation aspects

for $k = 0, 1, \ldots$

$$\alpha_k = \frac{r_k^* r_k}{p_k^* A p_k}$$

$$x_{k+1} = x_k + p_k \alpha_k$$

$$r_{k+1} = r_k - A p_k \alpha_k$$

$$\beta_{k+1} = \frac{r_{k+1}^* A p_k}{p_k^* A p_k}$$

$$p_{k+1} = r_k + p_k \beta_{k+1}$$

end

* Reduced latency when computing Ap_k

* `-dots`: 2 Synchronization points.

* x_k updated only periodically: $x_{k+1} = x_{k-\ell} + [p_{k-\ell}, \ldots, p_k]\boldsymbol{\alpha}_k$

Rich literature on HPC with Conjugate Gradient

The **Block** Conjugate Gradient. Implementation aspects

$R_0 = B - AX_0$, $P_0 = R_0 \in \mathbb{C}^{n\times s}$

for $k = 0, 1, \ldots$

$\boldsymbol{\alpha}_k = (P_k^* A P_k)^{-1}(R_k^* R_k) \in \mathbb{C}^{s\times s}$

$X_{k+1} = X_k + P_k \boldsymbol{\alpha}_k$

$R_{k+1} = R_k - A P_k \boldsymbol{\alpha}_k$

$\boldsymbol{\beta}_{k+1} = (P_k^* A P_k)^{-1}(R_{k+1}^* A P_k) \in \mathbb{C}^{s\times s}$

$P_{k+1} = R_k + P_k \boldsymbol{\beta}_{k+1}$

end

* Higher data locality associated with AP_k

* Rich in BLAS3 computations

It might be worth even for $Ax = b$

A more general picture

- A normal, $AA^* = A^*A$
- A (highly) non-normal, $\|AA^* - A^*A\| \gg 0$
- A “Hermitian” in disguise:

A more general picture

- A normal, $AA^* = A^*A$
- A (highly) non-normal, $\|AA^* - A^*A\| \gg 0$
- A "Hermitian" in disguise:
  - $\star$ $A = M + \sigma I,\ \sigma \in \mathbb{C}$, $M \in \mathbb{R}^{n\times n}$ symmetric

A more general picture

- A normal, $AA^* = A^*A$
- A (highly) non-normal, $\|AA^* - A^*A\| \gg 0$
- A “Hermitian” in disguise:
  - ⋆ $A = M + \sigma I, \ \sigma \in \mathbb{C}, M \in \mathbb{R}^{n \times n}$ symmetric
  - ⋆ $A = M + \sigma H, \ \sigma \in \mathbb{C}, M, H \in \mathbb{R}^{n \times n}$ symmetric

A more general picture

- A normal, $AA^* = A^*A$
- A (highly) non-normal, $\|AA^* - A^*A\| \gg 0$
- A "Hermitian" in disguise:
  - ⋆ $A = M + \sigma I$, $\sigma \in \mathbb{C}$, $M \in \mathbb{R}^{n\times n}$ symmetric
  - ⋆ $A = M + \sigma H$, $\sigma \in \mathbb{C}$, $M, H \in \mathbb{R}^{n\times n}$ symmetric
  - ⋆ There exists nonsing. Herm. $H \in \mathbb{C}^{n\times n}$ such that $HA = A^*H$, e.g. M, C Hermitian

$$A = \begin{bmatrix} M & B \\ -B^* & C \end{bmatrix}, \quad H = \begin{bmatrix} I & \\ & -I \end{bmatrix},$$

A more general picture

- A normal, $AA^* = A^*A$
- A (highly) non-normal, $\|AA^* - A^*A\| \gg 0$
- A "Hermitian" in disguise:

  ⋆ $A = M + \sigma I,\ \sigma \in \mathbb{C},\ M \in \mathbb{R}^{n\times n}$ symmetric

  ⋆ $A = M + \sigma H,\ \sigma \in \mathbb{C},\ M, H \in \mathbb{R}^{n\times n}$ symmetric

  ⋆ There exists nonsing. Herm. $H \in \mathbb{C}^{n\times n}$ such that $HA = A^*H$, e.g. M, C Hermitian

$$A = \begin{bmatrix} M & B \\ -B^* & C \end{bmatrix}, \quad H = \begin{bmatrix} I & \\ & -I \end{bmatrix},$$

⋆ $Ax = b \quad \Leftrightarrow \quad A^*Ax = A^*b$ (not recommended in general...)

Outline

- What is the problem with A non-Hermitian ?
- How to handle "Symmetry in disguise"
- Non-normal (non-Hermitian) case
  - ⋆ Long-term recurrences and their problems in HPC
  - ⋆ Living with them ⇒ Restarted, truncated, flexible
  - ⋆ Making it without ⇒ short-term recurrences
- Tricks for all platforms

What goes "wrong" with A non-Hermitian. I

$$\{x_k\}, \quad \text{with} \quad x_k \in x_0 + K_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$$

Let $V_k = [v_1, \ldots, v_k]$ be a (orthogonal) basis of $K_k(A, r_0)$. Then

$$x_k = x_0 + V_k y_k, \qquad y_k \in \mathbb{C}^k$$

A condition is required to specify y_k.

What goes "wrong" with A non-Hermitian. I

$$\{x_k\}, \quad \text{with} \quad x_k \in x_0 + K_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$$

Let $V_k = [v_1, \ldots, v_k]$ be a (orthogonal) basis of $K_k(A, r_0)$. Then

$$x_k = x_0 + V_k y_k, \qquad y_k \in \mathbb{C}^k$$

A condition is required to specify y_k. For instance:

$$r_k := b - Ax_k = r_0 - AV_k y_k \quad \perp \quad K_k(A, r_0) \qquad V_k^* r_k = 0$$

so that

$$0 = V_k^* r_k = V_k^* r_0 - V_k^* A V_k y_k \quad \Leftrightarrow \quad y_k \; s.t. \; (V_k^* A V_k) y_k = V_k^* r_0$$

What goes "wrong" with A non-Hermitian. I

$$\{x_k\}, \quad \text{with} \quad x_k \in x_0 + K_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$$

Let $V_k = [v_1, \ldots, v_k]$ be a (orthogonal) basis of $K_k(A, r_0)$. Then

$$x_k = x_0 + V_k y_k, \qquad y_k \in \mathbb{C}^k$$

A condition is required to specify y_k. For instance:

$$r_k := b - Ax_k = r_0 - AV_k y_k \quad \perp \quad K_k(A, r_0) \qquad V_k^* r_k = 0$$

so that

$$0 = V_k^* r_k = V_k^* r_0 - V_k^* A V_k y_k \quad \Leftrightarrow \quad y_k \ s.t. \ (V_k^* A V_k) y_k = V_k^* r_0$$

Hence

$$x_k = x_0 + V_k (V_k^* A V_k)^{-1} V_k^* r_0 \qquad \text{with} \quad V_k^* r_0 = e_1 \|r_0\|$$

And: $V_k^* A V_k$ upper Hessenberg (Gram-Schmidt procedure to build V_k)

What goes "wrong" with A non-Hermitian. II

If A were Hpd $\Rightarrow$ $V_k^* A V_k$ also Hpd $\Rightarrow$ tridiagonal

$$V_k^* A V_k = L_k L_k^* \qquad L_k \text{ bidiagonal}$$

$$\begin{aligned} x_k &= x_0 + V_k L_k^{-*} L_k^{-1} e_1 \|r_0\| \\ &= x_0 + V_{k-1} L_{k-1}^{-*} L_{k-1}^{-1} e_1 \|r_0\| + p_k \alpha_k \\ &= x_{k-1} + p_k \alpha_k \end{aligned}$$

with $p_k \in \text{span}\{v_{k-1}, v_k\}$

(development underlying Conjugate Gradient)

What goes "wrong" with A non-Hermitian. II

If A were Hpd $\Rightarrow$ $V_k^* A V_k$ also Hpd $\Rightarrow$ tridiagonal

$$V_k^* A V_k = L_k L_k^* \qquad L_k \text{ bidiagonal}$$

$$\begin{aligned} x_k &= x_0 + V_k L_k^{-*} L_k^{-1} e_1 \|r_0\| \\ &= x_0 + V_{k-1} L_{k-1}^{-*} L_{k-1}^{-1} e_1 \|r_0\| + p_k \alpha_k \\ &= x_{k-1} + p_k \alpha_k \end{aligned}$$

with $p_k \in \text{span}\{v_{k-1}, v_k\}$

(development underlying Conjugate Gradient)

A non-Hermitian $\Rightarrow$ $V_k^* A V_k$ only upper Hessenberg

$$p_k \in \text{span}\{v_1, \ldots, v_k\}$$

What goes "wrong" with A non-Hermitian. III

$$x_k = x_{k-1} + p_k \alpha_k, \quad p_k \in \text{span}\{v_1, \ldots, v_k\}$$

with $\{v_1, \ldots, v_k\}$ orthogonal basis

Alternatives

- Give up orthogonal basis, $V_k^* V_k = I_k$
- Give up optimality condition, e.g. $r_k \perp K_k(A, r_0)$
- Resume symmetry

Symmetry in disguise

Case 1: $A = M + \sigma I, \quad M \in \mathbb{R}^{n \times n}, \sigma \in \mathbb{C}$

Trick: replace $*$ (conj. transp.) with $\top$ (transp.)

$$A = A^\top \qquad \text{complex symmetric}$$

Apply CG with $\top$

Given x_0. Set $r_0 = b - Ax_0$, $p_0 = r_0$

for $i = 0, 1, \ldots$

$$\alpha_i = \frac{r_i^\top r_i}{p_i^\top A p_i}$$

$$x_{i+1} = x_i + p_i \alpha_i$$

$$r_{i+1} = r_i - A p_i \alpha_i$$

$$\beta_{i+1} = \frac{r_{i+1}^\top A p_i}{p_i^\top A p_i}$$

$$p_{i+1} = r_i + p_i \beta_{i+1}$$

end

...and Complex Symmetric Matrices

$A = M + \sigma I$: Apply CG with $\top$

Properties:

- V_k real: $\quad K_k(A, r_0) = K_k(A + \sigma I, r_0)$
- $\top$ does not define an inner product!
- $V_k^\top A V_k = V_k^\top M V_k + \sigma I$

  If $\Im(\sigma) \neq 0$ then $V_k^\top A V_k$ is nonsingular $\Rightarrow$ No breakdown

The same code applies in case of any A complex symmetric $(A = A^\top)$

H-symmetry

A is H-Hermitian if there exists $H \in \mathbb{C}^{n \times n}$ Hermitian, nonsingular s.t.

$$HA = A^* H$$

(H-symmetric if $HA = A^\top H$ with H is symmetric)

H-symmetry

A is H-Hermitian if there exists $H \in \mathbb{C}^{n\times n}$ Hermitian, nonsingular s.t.

$$HA = A^*H$$

(H-symmetric if $HA = A^\top H$ with H is symmetric)

If H is Hpd (and HA is also Hpd), use CG in the H-inner product:

Given x_0. Set $r_0 = b - Ax_0$, $p_0 = r_0$

for $i = 0, 1, \ldots$

$$\alpha_i = \frac{r_i^* H r_i}{p_i^* HA p_i}$$

$$x_{i+1} = x_i + p_i \alpha_i$$

$$r_{i+1} = r_i - Ap_i\alpha_i$$

$$\beta_{i+1} = \frac{r_{i+1}^* HAp_i}{p_i^* HAp_i}$$

$$p_{i+1} = r_i + p_i\beta_{i+1}$$

end

(H not Hpd $\Rightarrow$ see later lecture)

First Summary

Symmetry in disguise:

- Shifted matrices, $A = M + \sigma I$, M real symmetric
- Complex symmetric matrices
- H-symmetric or H-Hermitian matrices

Long-term recurrences

$$K_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}, \quad V_k \text{ orth. basis}$$

1. Arnoldi process : $v_{k+1} \leftarrow Av_k - \sum_{j=1}^{k} v_j h_{j,k}$, that is

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^* = V_{k+1} \underline{H}_k \qquad (H_k = V_k^* A V_k)$$

2. $x_k = x_0 + V_k y_k$

Long-term recurrences

$$K_k(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}, \quad V_k \text{ orth. basis}$$

1. Arnoldi process : $v_{k+1} \leftarrow Av_k - \sum_{j=1}^{k} v_j h_{j,k}$, that is

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^* = V_{k+1} \underline{H}_k \qquad (H_k = V_k^* A V_k)$$

2. $x_k = x_0 + V_k y_k$

- GMRES. Particular Petrov-Galerkin condition:

$$r_k \perp AK_k \Rightarrow \quad y_k \text{ s.t. } \min_y \|r_0 - AV_k y\|$$

- FOM. Galerkin condition: (H_k nonsingular)

$$r_k \perp K_k \Rightarrow \quad y_k \text{ s.t. } H_k y = e_1 \|r_0\|$$

GMRES and HPC

$$\|r_0 - AV_k y\| = \| V_{k+1} e_1 \|r_0\| - V_{k+1} \underline{H}_k y \| = \| e_1 \|r_0\| - \underline{H}_k y \|$$

$$\underline{H}_k = \begin{bmatrix} \underline{H}_{k-1} & h \\ 0 & h_{k+1,k} \end{bmatrix} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & 0 & \ddots & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

GMRES and HPC

$$\|r_0 - AV_k y\| \quad = \quad \| V_{k+1} e_1 \|r_0\| - V_{k+1} \underline{H}_k y \| = \| e_1 \|r_0\| - \underline{H}_k y \|$$

$$\underline{H}_k = \begin{bmatrix} \underline{H}_{k-1} & h \\ 0 & h_{k+1,k} \end{bmatrix} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & 0 & \ddots & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

QR decomposition: $$\begin{bmatrix} R_k \\ 0 \end{bmatrix} = \Omega_k \Omega_{k-1} \cdots \Omega_1 \underline{H}_k =: Q_k^* \underline{H}_k$$

GMRES and HPC

$$\|r_0 - AV_k y\| \quad = \quad \| V_{k+1} e_1 \|r_0\| - V_{k+1} \underline{H}_k y \| = \| e_1 \|r_0\| - \underline{H}_k y \|$$

$$\underline{H}_k = \begin{bmatrix} \underline{H}_{k-1} & h \\ 0 & h_{k+1,k} \end{bmatrix} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & 0 & \ddots & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

QR decomposition: $\begin{bmatrix} R_k \\ 0 \end{bmatrix} = \Omega_k \Omega_{k-1} \cdots \Omega_1 \underline{H}_k =: Q_k^* \underline{H}_k$

$$\min_y \| e_1 \|r_0\| - \underline{H}_k y \| = \min_y \| Q_k^* e_1 \|r_0\| - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y \| = |e_{k+1}^* Q_k^* e_1| \, \|r_0\|$$

The whole process

```
k=1
while (not converged)
  V(:,k+1) = a*V(:,k);      % Mxv
  for j=1:k                 % Update basis (Modified Gram-Schmidt)
     H(j,k) = V(:,j)'*V(:,k+1);
     V(:,k+1)=V(:,k+1)-H(j,k)*V(:,j);
  end;
  H(k+1,k) = norm(V(:,k+1));
  if (H(k+1,k) ~= 0.) V(:,k+1)=V(:,k+1)/H(k+1,k); end;

  Update Rotations
  Check Convergence
  k=k+1
end
Compute y, Compute x:  x=V(:,1:end)*y
```

1 Mxv, k -axpys, $k+1$ -dots (highly sequential, high latency)

Block GMRES

$$R_0 = B - AX_0, \qquad K_k(A, R_0) = \text{span}\{R_0, AR_0, \ldots, A^{k-1}R_0\},$$

$\mathcal{U}_k$ orth. basis, $\mathcal{U}_k = [U_1, U_2, \ldots, U_k] \in \mathbb{C}^{n \times ks}$

Block Arnoldi process (s MxV + Gram-Schmidt)

$\Rightarrow \quad A\mathcal{U}_k = \mathcal{U}_k \mathcal{H}_k + U_{k+1} \chi_{k+1,k} E_k^* = \mathcal{U}_{k+1} \underline{\mathcal{H}}_k \qquad (\mathcal{H}_k = \mathcal{U}_k^* A \mathcal{U}_k)$

$$\min_Y \|R_0 - A\mathcal{U}_k Y\| = \min_Y \| E_1 \boldsymbol{\rho} - \underline{\mathcal{H}}_k Y \| \qquad R_0 = U_1 \boldsymbol{\rho}$$

$$\underline{\mathcal{H}}_k = \begin{bmatrix} \square & \square & \cdots & \square \\ \square & \square & \cdots & \square \\ O & \square & \cdots & \square \\ O & O & \ddots & \square \\ O & O & O & \square \end{bmatrix}$$

Block GMRES: The whole process

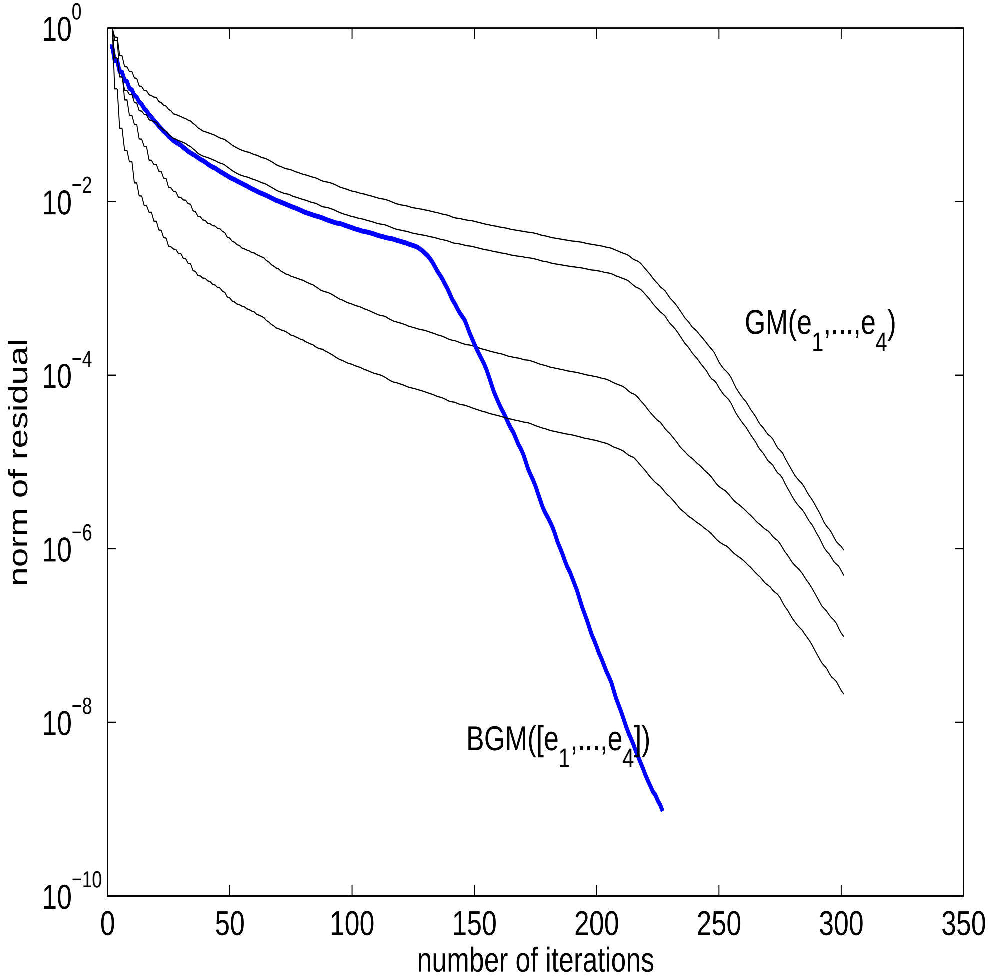
```
[U(:,1:s,1),ibeta]=MGS(R0);
k=1
while (not converged)
  U(:,1:s,k+1) = a*U(:,1:s,k);    % Mxv
  for j=1:k                    % Update basis (Modified Gram-Schmidt)
     t = U(:,1:s,j)'*U(:,1:s,k+1);
     U(:,1:s,k+1)=U(:,1:s,k+1)-U(:,1:s,j)*t;
     Update H
  end;
  [U(:,1:s,k+1),ibeta]=MGS(U(:,1:s,k+1));
  Update H

  Update Rotations
  Check Convergence
  k=k+1
end
Compute y, Compute x
```

1 Mxv, k -axpys, $k+1$ -dots (highly sequential, high latency)

Block GMRES

$A \in \mathbb{R}^{6400\times 6400}$: FD discretiz. of $\mathcal{L}(u) = -\Delta u + \frac{1000}{x+y} u_x$ in $[-1,1]^2$



Strategies to enhance the Modified Arnoldi procedure in HPC

- Classical Gram-Schmidt (loss of orthogonality)
- Double Classical Gram-Schmidt (...better)

Strategies to enhance the Modified Arnoldi procedure in HPC

- Classical Gram-Schmidt (loss of orthogonality)
- Double Classical Gram-Schmidt (...better)
- Selective orthogonalization (local orthogonality)

Strategies to enhance the Modified Arnoldi procedure in HPC

- Classical Gram-Schmidt (loss of orthogonality)
- Double Classical Gram-Schmidt (...better)
- Selective orthogonalization (local orthogonality)
- Blocking:

  Compute $[r_0, Ar_0, \ldots, A^\ell r_0]$ then orthogonalize

  Unstable procedure $\Rightarrow$ $[r_0, p_1(A)r_0, \ldots, p_\ell(A)r_0]$ Stable

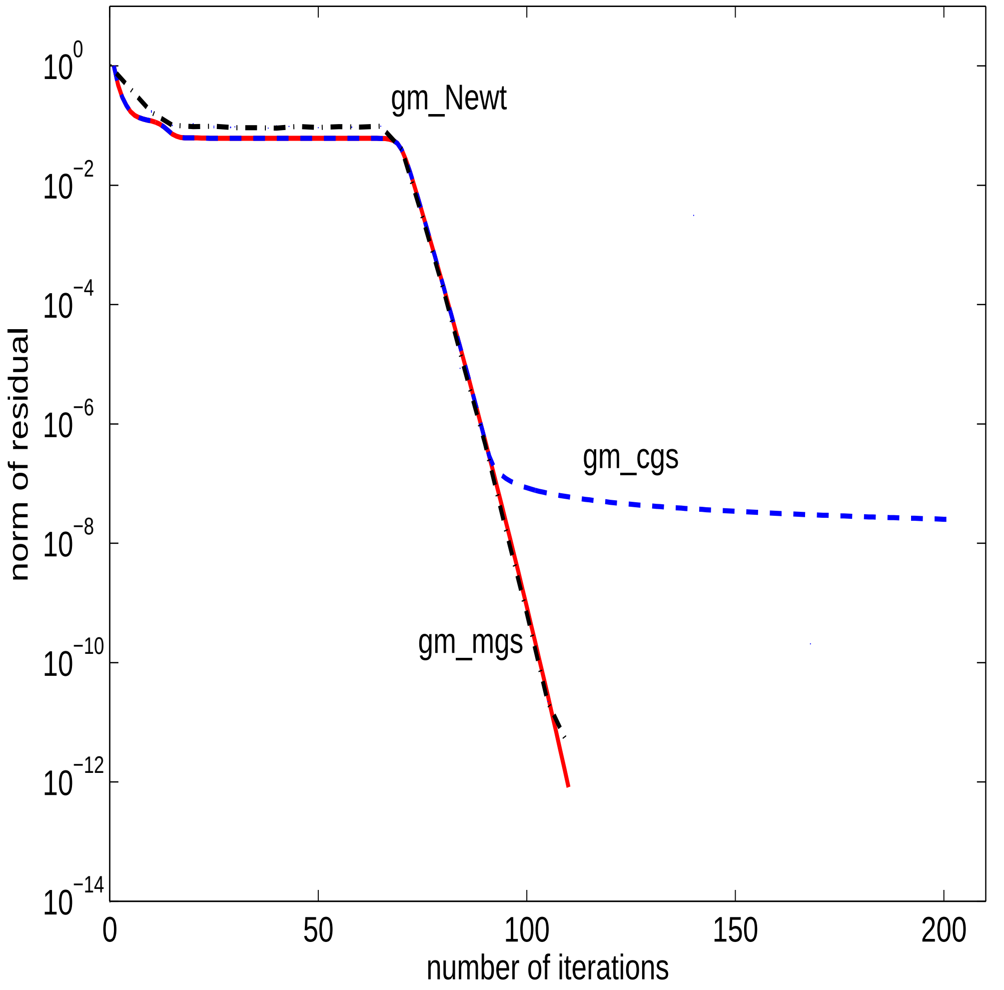
Strategies to enhance the Modified Arnoldi procedure in HPC

- Classical Gram-Schmidt (loss of orthogonality)
- Double Classical Gram-Schmidt (...better)
- Selective orthogonalization (local orthogonality)
- Blocking:

  Compute $[r_0, Ar_0, \ldots, A^\ell r_0]$ then orthogonalize

  Unstable procedure $\quad \Rightarrow \quad [r_0, p_1(A)r_0, \ldots, p_\ell(A)r_0]$ Stable
- Block Arnoldi: preferable situation ($\rightarrow$ dynamic block size)

HPC and Arnoldi recurrence: $A = \mathrm{diag}(1:200) + \mathrm{diag}(\mathbf{1}, 1)$



Living with long-term recurrences

Restarted, Truncated, Flexible variants.

Living with long-term recurrences

Restarted, Truncated, Flexible variants.

Restarted: Choose $m_{\max}$.

Set $x = x_0$, $r_0 = b - Ax_0$

for $i = 1, 2, \ldots$

$z \leftarrow$ GMRES($A, r_0, m_{\max}$) (or other method)

$x \leftarrow x + z, \quad r_0 = b - Ax$

Check Convergence

Pros and Cons

Pros:

- Shorter dependencies
- Lower and fixed memory requirements

Pros and Cons

Pros:

- Shorter dependencies
- Lower and fixed memory requirements

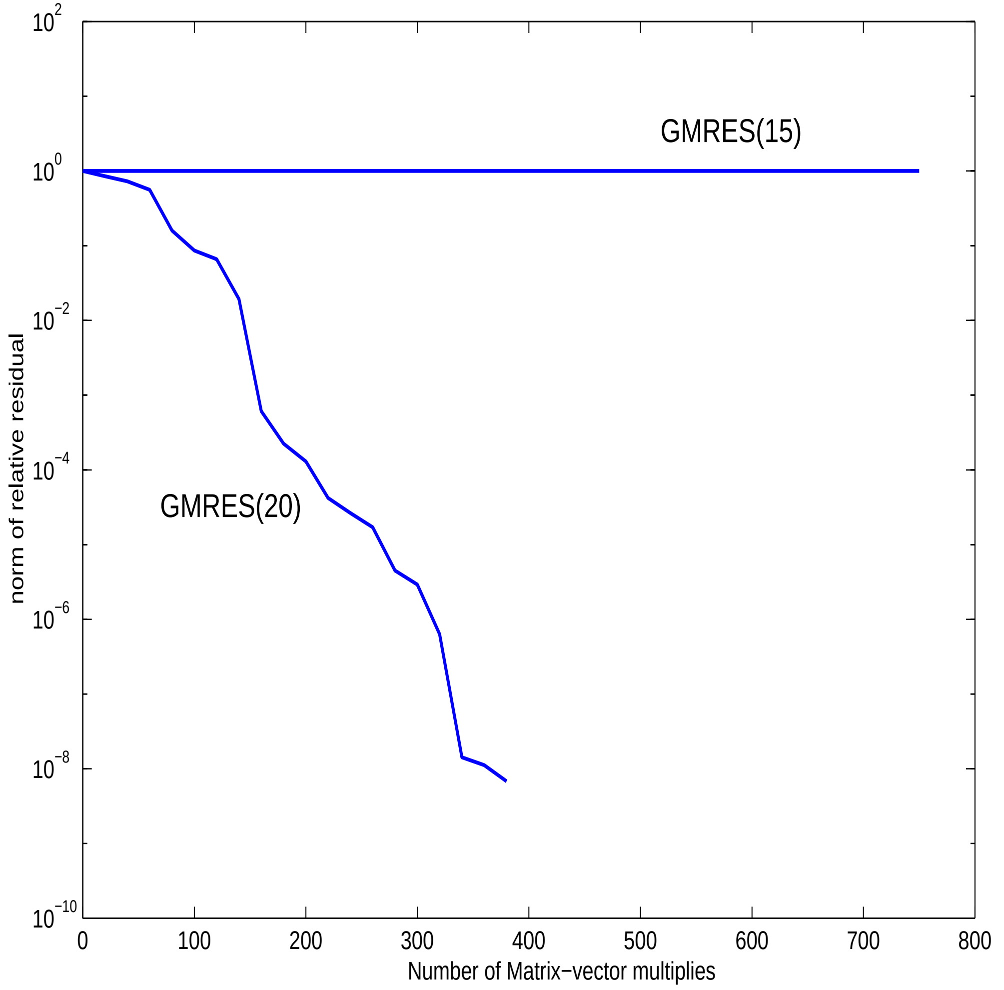
Cons:

- All optimality properties are lost

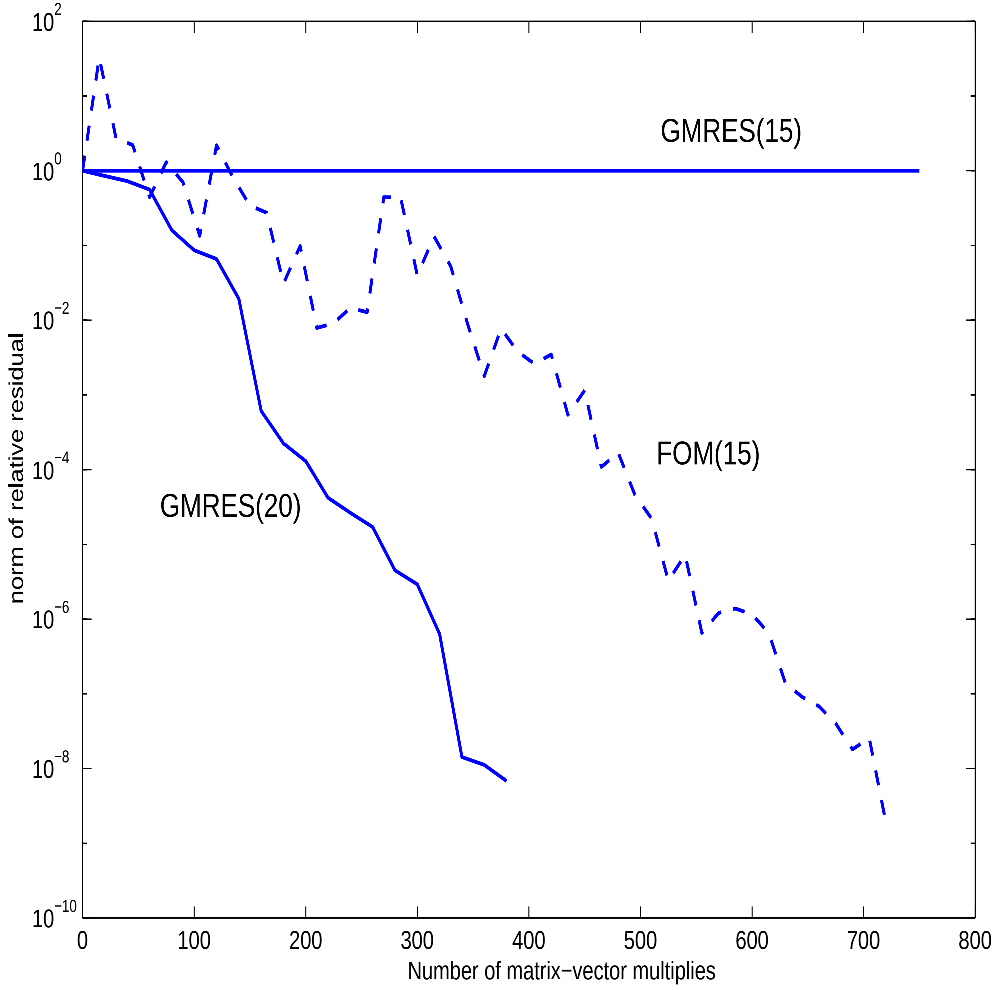
$$K_{m_{\max}}(A, r_0^{(0)}) + K_{m_{\max}}(A, r_0^{(1)}) + \ldots K_{m_{\max}}(A, r_0^{(k)}) + \ldots$$

- Additional parameter. What value for $m_{\max}$??

A problem with the restarting parameter? ...



A problem with the restarting parameter? ... or with the method?



Explanation

$$K_{m_{\max}}(A, r_0^{(0)}) + K_{m_{\max}}(A, r_0^{(1)}) + \ldots K_{m_{\max}}(A, r_0^{(k)}) + \ldots$$

GMRES: $r_0^{(k)} \in \mathrm{span}(V_{m_{\max}+1}^{(k-1)})$. Almost stagnation: $\longrightarrow r_0^{(k)} \propto v_1^{(k-1)}$

Explanation

$$K_{m_{\max}}(A, r_0^{(0)}) + K_{m_{\max}}(A, r_0^{(1)}) + \ldots K_{m_{\max}}(A, r_0^{(k)}) + \ldots$$

GMRES: $r_0^{(k)} \in \mathrm{span}(V_{m_{\max}+1}^{(k-1)})$. Almost stagnation: $\rightarrow r_0^{(k)} \propto v_1^{(k-1)}$

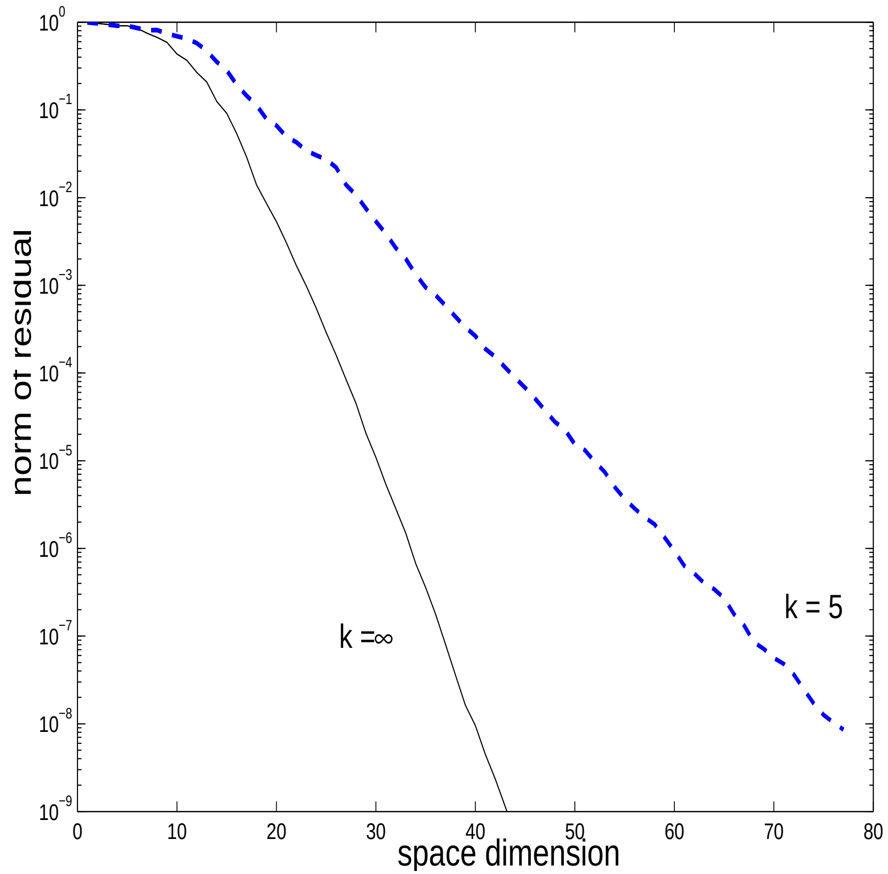
FOM: $r_0^{(k)} \propto v_{m_{\max}+1}^{(k-1)}$ Subspace keeps growing

Truncating

Only local orthogonalization (k-term recurrence, H_m banded)

Truncating

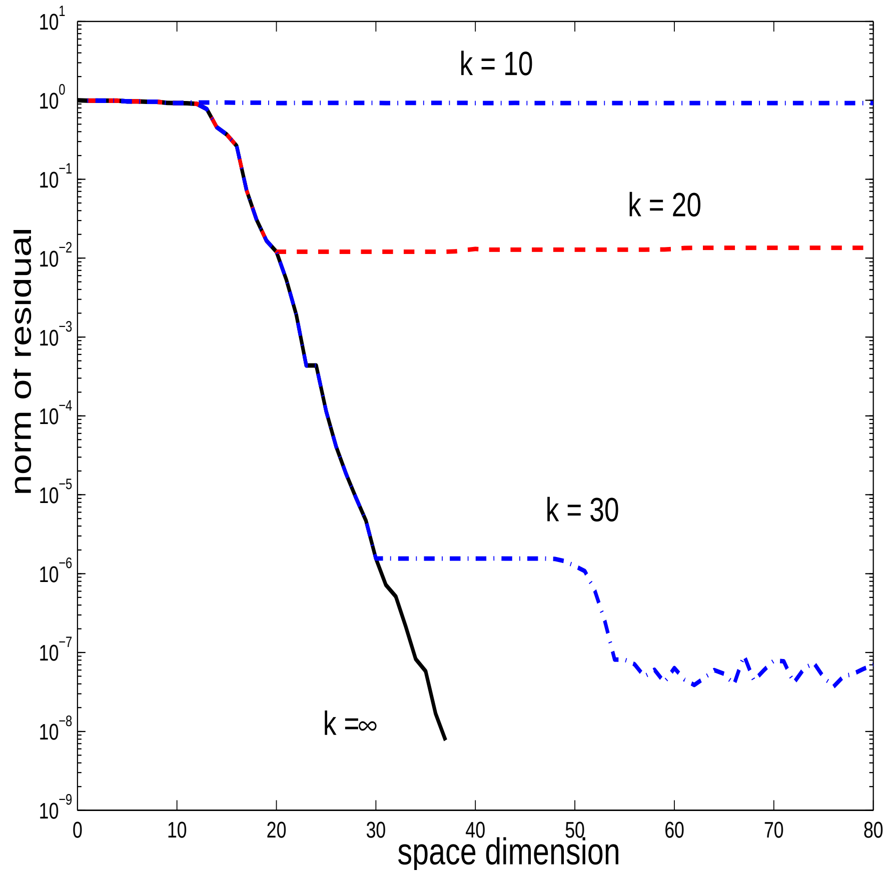
Only local orthogonalization (k-term recurrence, H_m banded)



a reasonable strategy (low latency and dependencies)

Truncating

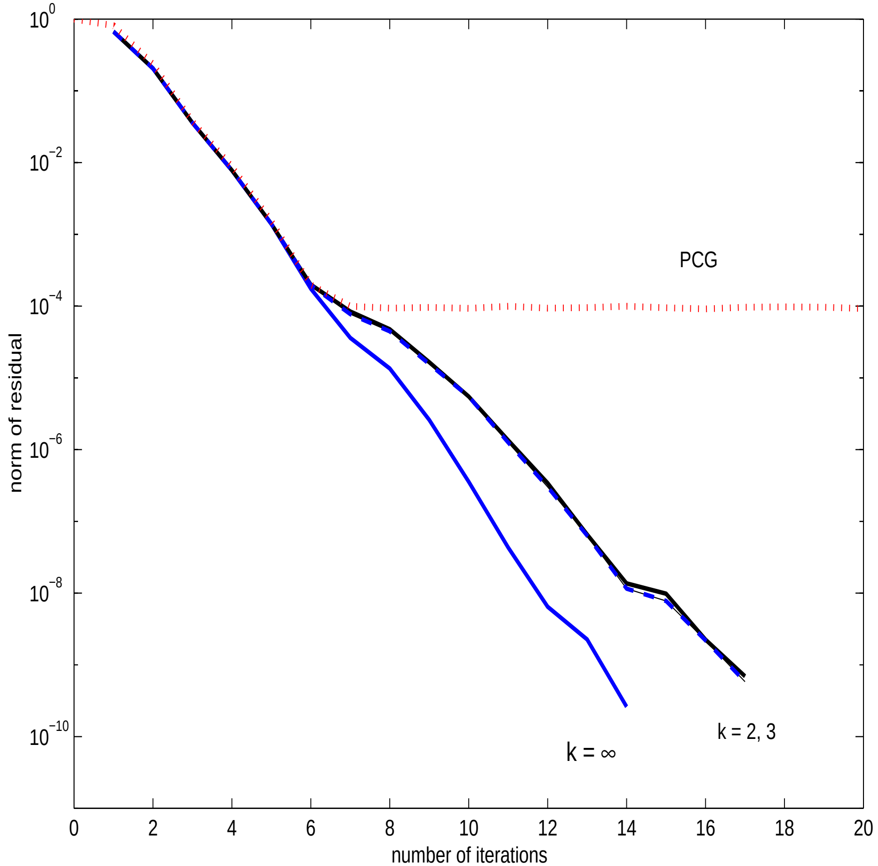
...but not always good



Truncating

A good strategy for P-CG with A symmetric and P inexact precond

$$w = P^{-1}Av + \epsilon\mathbf{1}, \quad \epsilon = 10^{-5}$$



Changing K_k. Flexible methods

Original problem

$$AP^{-1}x = b \qquad P \text{ preconditioner}$$

$$\mathcal{K}_k(AP^{-1}, r_0) = \text{span}\{r_0, AP^{-1}r_0, \ldots, (AP^{-1})^{k-1}r_0\}$$

at each iteration i: $\qquad z_i = P^{-1}v_i$

Changing K_k. Flexible methods

Original problem

$$AP^{-1}x = b \qquad P \text{ preconditioner}$$

$$\mathcal{K}_k(AP^{-1}, r_0) = \text{span}\{r_0, AP^{-1}r_0, \ldots, (AP^{-1})^{k-1}r_0\}$$

at each iteration i: $\qquad z_i = P^{-1}v_i$

Flexible variant:

$$\text{Iteration } i: \qquad z_i = P^{-1}v_i \qquad \Rightarrow \qquad z_i = P_i^{-1}v_i$$

$$\widetilde{x}_m \in \text{span}\{r_0, z_1, z_2, \ldots, z_{m-1}\} \neq \mathcal{K}_k(AP^{-1}, r_0)$$

FGMRES: The whole process

```
k=1
while (not converged)
  Z(:,k) = prec(V(:,k));    % Precond step
  V(:,k+1) = a*Z(:,k);      % Mxv step
  for j=1:k                 % Update basis (Modified Gram-Schmidt)
     H(j,k) = V(:,j)'*V(:,k+1);
     V(:,k+1)=V(:,k+1)-H(j,k)*V(:,j);
  end;
  H(k+1,k) = norm(V(:,k+1));
  if (H(k+1,k) ~= 0.) V(:,k+1)=V(:,k+1)/H(k+1,k); end;

  Update Rotations
  Check Convergence
  k=k+1
end
Compute y, Compute x:   x = Z(:,1:end)*y
```

1 Mxv, 1 prec, k -axpys, $k+1$ -dots, Memory: V, Z

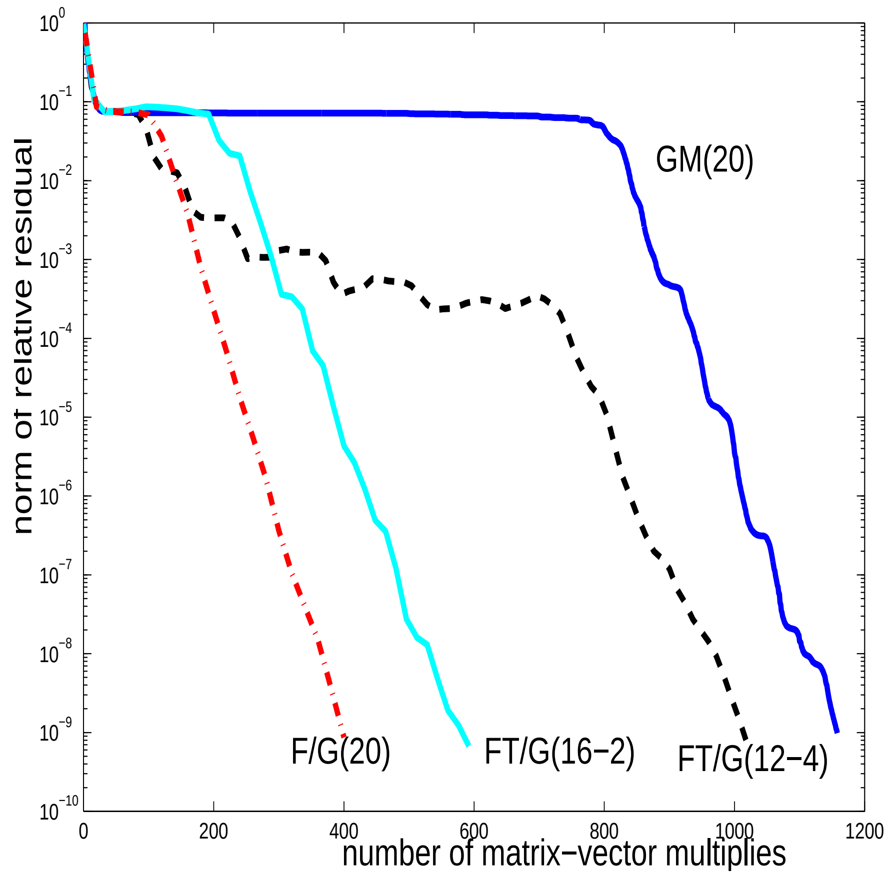
Flexible and Truncated method. An example

$z = P^{-1}v \quad \Leftrightarrow z \approx A^{-1}v$ span$\{r_0, z_1, z_2, \dots, z_{m-1}\}$

Flexible and Truncated method. An example

$z = P^{-1}v \quad \Leftrightarrow z \approx A^{-1}v$ span$\{r_0, z_1, z_2, \ldots, z_{m-1}\}$

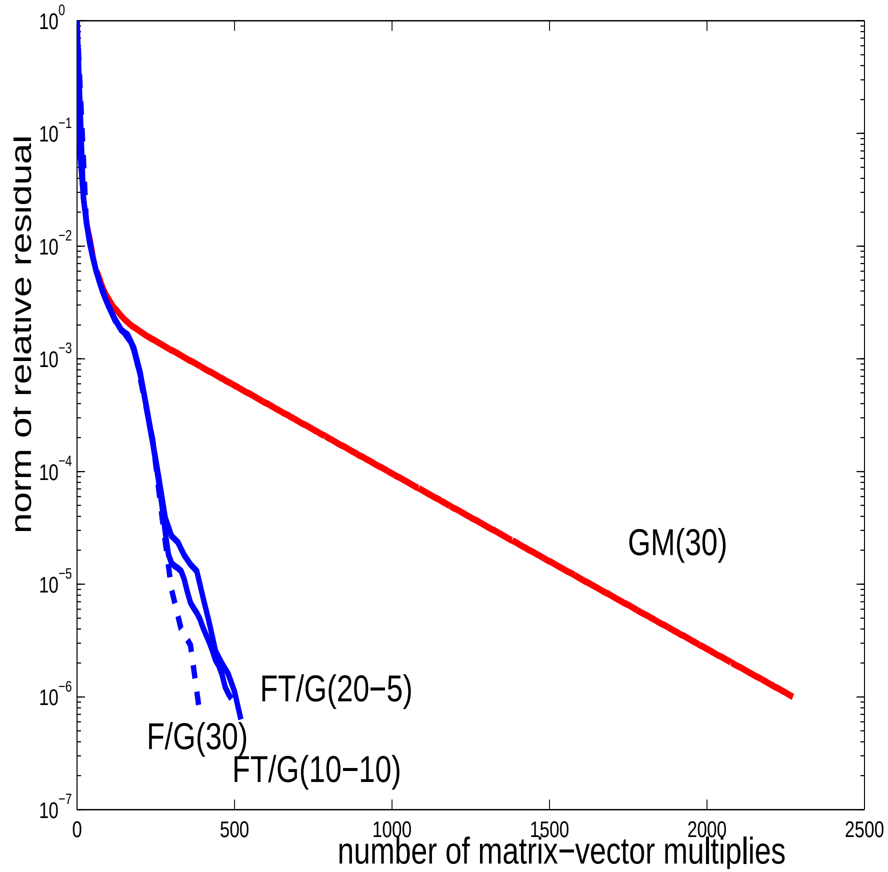
A from $L(u) = -\Delta u + 1000 x u_x$ $n = 900$



Flexible and Truncated method. A second example

$z = P^{-1}v \quad \Leftrightarrow z \approx A^{-1}v$ span$\{r_0, z_1, z_2, \ldots, z_{m-1}\}$

$$L(u) = -1000\Delta u + 2e^{4(x^2+y^2)}u_x - 2e^{4(x^2+y^2)}u_y \qquad n = 40\,000$$



Second Summary

Long-term recurrences:

- Optimal methods (e.g. GMRES), single and multiple right-hand sides
- HPC issues (Gram-Schmidt), Mxv not treated
- Restarted, truncated, flexible (and combinations thereof)

Making it without: short-term recurrences for A non-Hermitian

Change optimality condition: Non-Hermitian Lanczos

$$r_k \perp K_k(A^\top, \tilde{r}_0), \qquad \tilde{r}_0 \text{ freely chosen}$$

Range(V_k)$= K_k(A, r_0)$, Range(W_k)$= K_k(A^\top, \tilde{r}_0)$ and s.t.

$$W_k^\top V_k \text{ diagonal}$$

$$AV_k = V_k T_k + v_{k+1} t_{k+1,k} e_k^\top, \quad A^\top W_k = W_k T_k^\top + w_{k+1} t_{k,k+1} e_k^\top,$$

Bi-orthogonal recurrence, T_k tridiagonal $\Rightarrow$ 3-term recurrence

Making it without: short-term recurrences for A non-Hermitian

Change optimality condition: Non-Hermitian Lanczos

$$r_k \perp K_k(A^\top, \tilde{r}_0), \qquad \tilde{r}_0 \text{ freely chosen}$$

Range(V_k)$= K_k(A, r_0)$, Range(W_k)$= K_k(A^\top, \tilde{r}_0)$ and s.t.

$W_k^\top V_k$ diagonal

$$AV_k = V_k T_k + v_{k+1} t_{k+1,k} e_k^\top, \quad A^\top W_k = W_k T_k^\top + w_{k+1} t_{k,k+1} e_k^\top,$$

Bi-orthogonal recurrence, T_k tridiagonal $\Rightarrow$ 3-term recurrence

$*$ Requires $A^\top$

$*$ Robustness problems

$\Rightarrow$ Special case: Simplified Lanczos

Simplified Lanczos

The typical problem

$$AH^{-1}x = b, \quad A, H \text{ symmetric,}$$

Range(V_k)$= K_k(AH^{-1}, r_0)$, Range(W_k)$= K_k(H^{-1}A, \tilde{r}_0)$ and s.t.

$$W_k^\top V_k \qquad \text{diagonal}$$

Simplified Lanczos

The typical problem

$$AH^{-1}x = b, \quad A, H \text{ symmetric,}$$

Range(V_k)= $K_k(AH^{-1}, r_0)$, Range(W_k)= $K_k(H^{-1}A, \tilde{r}_0)$ and s.t.

$$W_k^\top V_k \qquad \text{diagonal}$$

$\star$ If $\tilde{r}_0 = H^{-1}r_0$ then $W_k = H^{-1}V_k$

$\Rightarrow$ W_k obtained for free

- Short-term recurrence (cost similar to that of CG)
- Used for A, H indefinite (e.g. Saddle point problems)

An example: $AP^{-1}x = b$

$$A = \begin{bmatrix} M & B^\top \\ B & -C \end{bmatrix} \text{ symmetric} \qquad P = \begin{bmatrix} \widetilde{M} & B^\top \\ B & -\widetilde{C} \end{bmatrix} \text{ symmetric}$$

P: Constraint Preconditioner - used in (cheaper!) factored form

An example: $AP^{-1}x = b$

$$A = \begin{bmatrix} M & B^\top \\ B & -C \end{bmatrix} \text{ symmetric} \qquad P = \begin{bmatrix} \widetilde{M} & B^\top \\ B & -\widetilde{C} \end{bmatrix} \text{ symmetric}$$

P: Constraint Preconditioner - used in (cheaper!) factored form

Apply Simplified Lanczos-type method: Quasi Minimal Residual

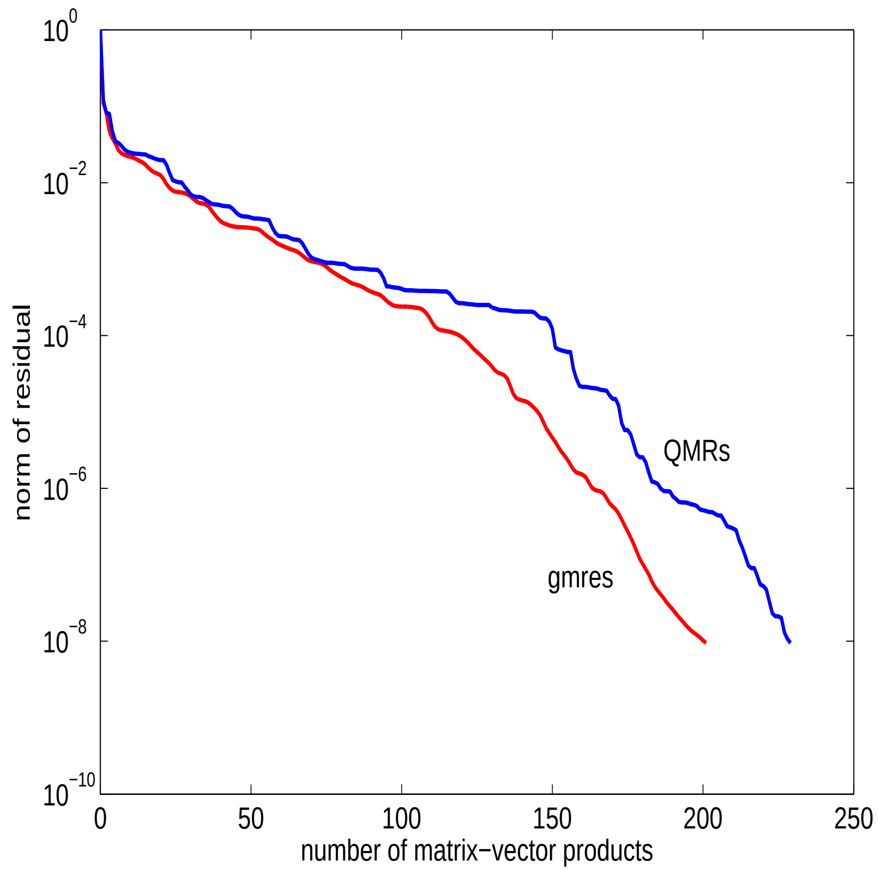
$$\|b - Ax_k\| = \|V_{k+1} \left(e_1 \|r_0\| - \underline{T}_k y\right) \|$$

$$\min_y \|e_1 \|r_0\| - \underline{T}_k y\|$$

V_{k+1} not orthogonal

An example: Stokes problem

Lid Driven Cavity problem from IFISS. Default params. A of size 49666



CPU Time: GMRES $=$ 51.66 secs, QMRs $=$ 6.26 secs

(my own GMRES code)

Short-term recurrences

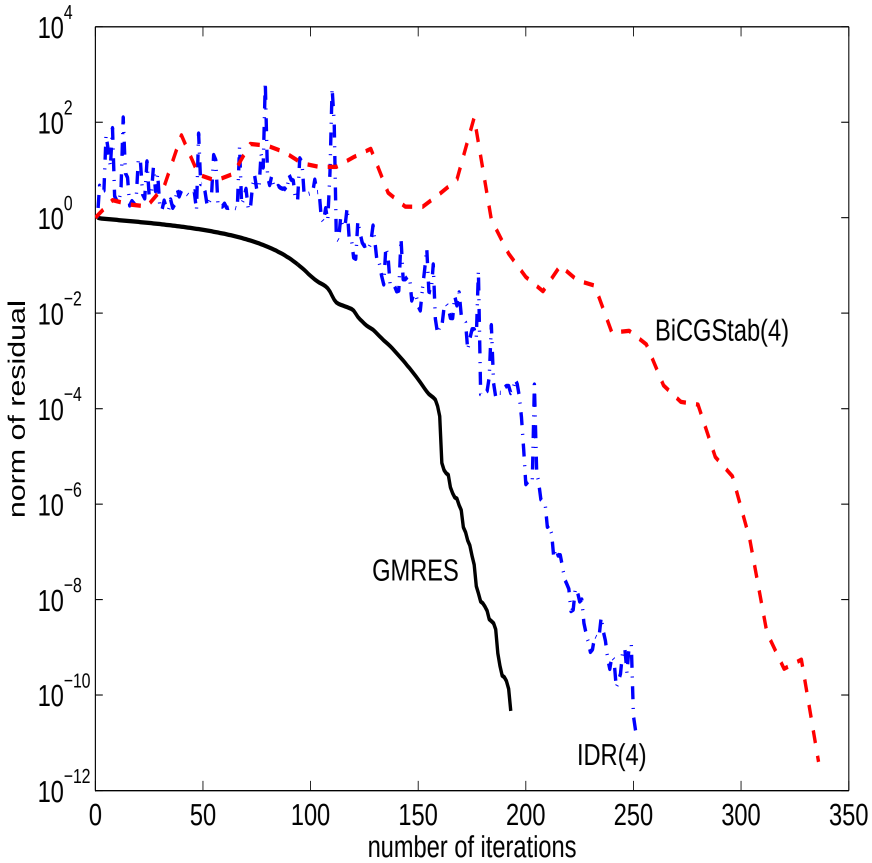
Local optimality conditions:

Polynomial methods, like CG:

- BiCGStab(ℓ): ℓ iterations of GMRES at every step
- IDR(s): $r_k \in \mathcal{G}_k$, where $\mathcal{G}_{k+1} \subset \mathcal{G}_k$

$$\mathcal{G}_{k+1} = (\mu_{k+1} I - A)(\mathcal{G}_k \cap \widetilde{R}_0^{\perp}), \quad \widetilde{R}_0 \in \mathbb{C}^{n\times s},\ \mathcal{G}_0 = \mathbb{C}.$$

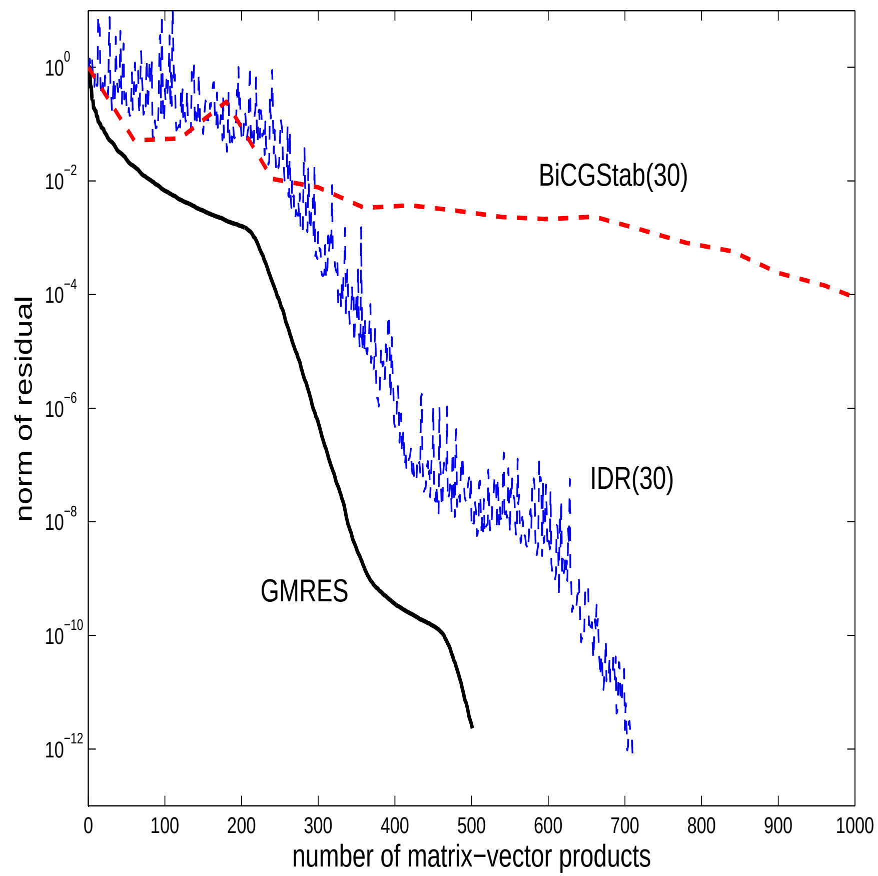
An Example: $L(u) = -\Delta u + 50(x+y)(u_x + u_y)$ $\qquad n = 6\,400$



CPU Time: GMRES = 3.65 secs, IDR(4) = 0.22 secs, BiCGStab(4) = 0.32 secs

(Matlab version of GMRES)

An Example: $L(u) = -\Delta u + 1000/(x+y)u_x$ $\qquad n = 6\,400$



CPU Time: GMRES = 24 secs, IDR(30) = 2.58 secs, BiCGStab(30) = 20 secs

(Matlab version of GMRES)

Tricks for all platforms

- Stopping criterion
- Operator inexactness

Tricks for all platforms

- Stopping criterion
- Operator inexactness

Stopping criterion:

- Problem dependent
- Matrix dependent

Stopping criterion within Rayleigh Quotient Iteration

Problem: Compute smallest eigenvalue(s) of A

Stopping criterion within Rayleigh Quotient Iteration

Problem: Compute smallest eigenvalue(s) of A

Rayleigh Quotient iteration:

Given y_0, compute $\theta_0 = y_0^* A y_0, \quad s_0 = A y_0 - y_0 \theta_0$

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

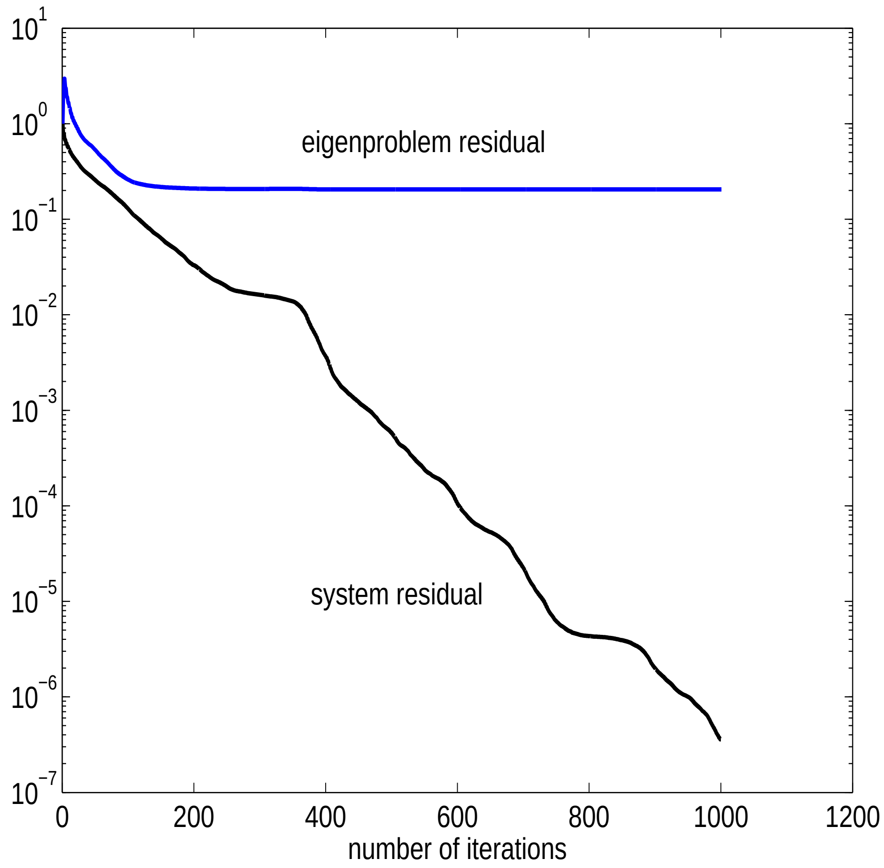
Solve $(A - \theta_k I)t = y_k$

Set $y_{k+1} = t / \|t\|, \;\; \theta_{k+1} = y_{k+1}^* A y_{k+1}$

$s_{k+1} = A y_{k+1} - y_{k+1} \theta_{k+1}$

$\theta_k \to \lambda$, $y_k \to x$ with (λ, x) eigenpair of A

An Example: A 2D Laplace operator



Generic kth RQI iteration. System to be solved: $(A - \theta_k I)t = y_k$

Stopping criterion: Problem dependence

Choice of tolerance:

- Direct method accurate up to machine precision (likely)
- Iterative method accurate up to what is wanted (hopefully)

Stopping criterion: Problem dependence

Choice of tolerance:

- Direct method accurate up to machine precision (likely)
- Iterative method accurate up to what is wanted (hopefully)

Algebraic problem: Discretization of PDEs

$$\text{error} \quad \to O(h^2)$$

h discretization parameter...

Stopping criterion: Problem dependence

Choice of criterion and norm:

$$\|b - Ax_k\|_2 \qquad \text{vs.} \qquad \|b - Ax_k\|_*$$

Stopping criterion: Problem dependence

Choice of criterion and norm:

$$\|b - Ax_k\|_2 \qquad \text{vs.} \qquad \|b - Ax_k\|_*$$

For instance, CG optimal: $(\|x\|_A^2 = x^* Ax)$

$$\min_{x_k \in x_0 + K_k(A, r_0)} \|b - Ax_k\|_{A^{-1}} = \min_{x_k \in x_0 + K_k(A, r_0)} \|x - x_k\|_A$$

Available: Cheap, reliable estimates of $\|x - x_k\|_A$

Stopping criterion: Problem dependence

Choice of criterion and norm:

$$\|b - Ax_k\|_2 \qquad \text{vs.} \qquad \|b - Ax_k\|_*$$

For instance, CG optimal: ($\|x\|_A^2 = x^* Ax$)

$$\min_{x_k \in x_0 + K_k(A, r_0)} \|b - Ax_k\|_{A^{-1}} = \min_{x_k \in x_0 + K_k(A, r_0)} \|x - x_k\|_A$$

Available: Cheap, reliable estimates of $\|x - x_k\|_A$

For instance, matrix G associated with FE error measure:

$$\min_{x_k} \|b - Ax_k\|_G$$

Matrix dependence

A may be very ill-conditioned

$\Rightarrow$ small residual does not necessarily imply small error

Well-known fact, but often not used

$$\frac{\|b - Ax_k\|}{\|b\|} \quad \text{vs} \quad \frac{\|b - Ax_k\|}{\|b\| + \|A\|_\star \|x_k\|}$$

(here $x_0 = 0$)

Matrix dependence

Inner-outer methods. e.g. Solve

$$BM^{-1}B^\top x = b$$

Each multiplication with $A = BM^{-1}B^\top$ requires solving a system with M

$$u = Av \quad \Leftrightarrow \quad \begin{aligned} &\tilde{u} = B^\top v \\ &\tilde{\tilde{u}} \text{ solves } M\tilde{\tilde{u}} = \tilde{u} \\ &u = B\tilde{\tilde{u}} \end{aligned}$$

How accurately should one solve with M?

Matrix dependence

Inner-outer methods. e.g. Solve

$$BM^{-1}B^\top x = b$$

Each multiplication with $A = BM^{-1}B^\top$ requires solving a system with M

$$u = Av \quad \Leftrightarrow \quad \begin{array}{l} \tilde{u} = B^\top v \\ \tilde{\tilde{u}} \text{ solves } M\tilde{\tilde{u}} = \tilde{u} \\ u = B\tilde{\tilde{u}} \end{array}$$

How accurately should one solve with M?

Note: True residual $r_k = b - BM^{-1}B^\top x_k$ not available!

How accurately should one solve with M?

Typically: Inner tolerance $<$ Outer tolerance

But: if optimal Krylov method is used to solve $BM^{-1}B^\top x = b$ then:

$$\text{Inner tolerance} = \text{c} \cdot \frac{\text{Outer tolerance}}{\text{current outer residual}}$$

The inexact key relation

$$A_{\epsilon_j} v = Av + f_j \qquad \|f_j\| = O(\epsilon_j), \quad j = 1, 2, \ldots$$

$$AV_m = V_{m+1} \underline{H}_m + \underbrace{F_m}_{[f_1, f_2, \ldots, f_m]} \qquad F_m \text{ error matrix}$$

How large is F_m allowed to be?

Claim: the perturbation induced by ϵ_j may be far less devastating for $x_m \to x$ than $|\epsilon_j|$ would predict

$$Ax_m = AV_m y_m = V_{m+1} \underline{H}_m y_m + F_m y_m$$

$$\|F_m y_m\| \quad \text{small then} \quad V_{m+1} \underline{H}_m y_m \approx Ax_m$$

A dynamic setting

$$F_m y = [f_1, f_2, \dots, f_m] \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix} = \sum_{i=1}^{m} f_i \eta_i$$

◇ The terms $f_i \eta_i$ need to be small:

$$\|f_i \eta_i\| < \frac{1}{m}\epsilon \quad \forall i \quad \Rightarrow \quad \|F_m y\| < \epsilon$$

◇ If $|\eta_i|$ small $\quad \Rightarrow \quad \|f_i\|$ is allowed to be large

⋆ In several problems it can be shown that $\quad |\eta_i| \le \gamma_m \|r_{i-1}\|$

Relaxing the accuracy in linear systems

$A \cdot v_i$ not performed exactly $\quad \Rightarrow \quad (A + E_i)v_i = Av_i + f_i$

$$b - Ax_m = V_{m+1}(e_1\beta - \underline{H}_m y_m) - F_m y_m$$

E.g., for GMRES: If $\quad \|E_i\| \le \frac{\gamma}{m} \frac{1}{\|\tilde{r}_{i-1}\|}\varepsilon \quad i = 1, \ldots, m$ $(\gamma = \gamma(A))$, then

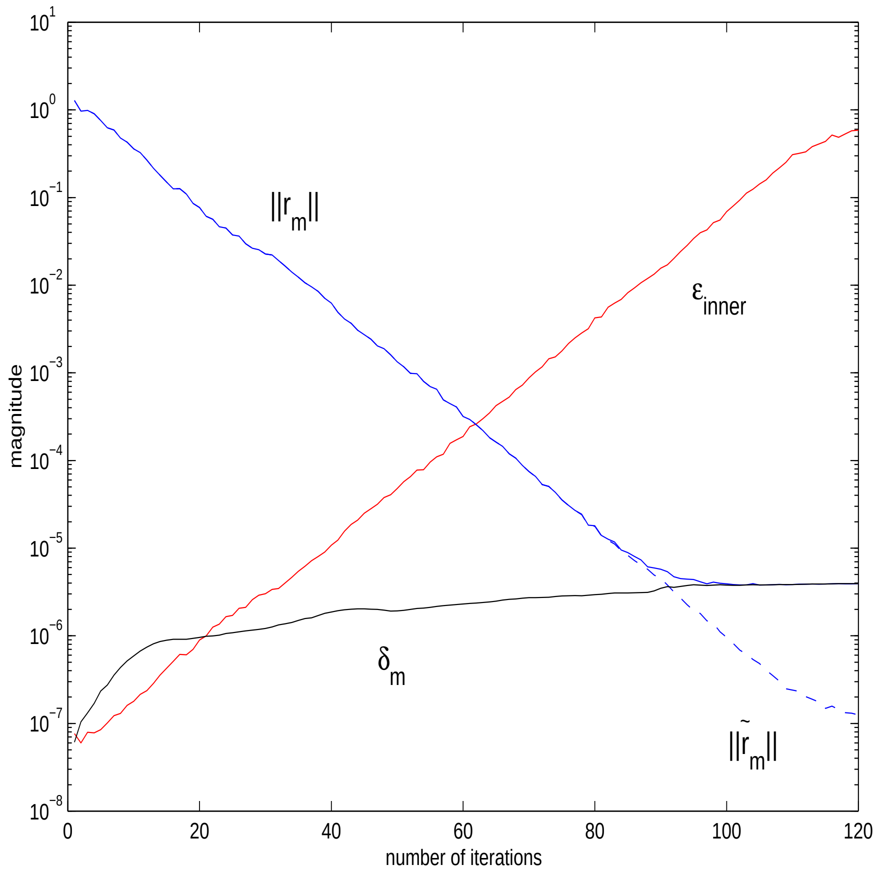
$\|F_m y_m\| \le \sum_{i=1}^{m} \|E_i\| |\eta_i| \le \varepsilon$ so that

$$\|(b - Ax_m) - V_{m+1}(e_1\beta - \underline{H}_m y_m)\| \le \varepsilon$$

Note: $\quad \|b - Ax_m\| \le \varepsilon$ final attainable residual norm

An example. GMRES

$$\varepsilon_{\text{inner}} = \frac{10^{-8}}{\|\tilde{r}_m\|}$$



$$r_m := \|b - Ax_m\|, \quad \tilde{r}_m := \|e_1 \|r_0\| - \underline{H}_m y_m\|, \quad \delta_m := \|r_m - \tilde{r}_m\|$$

Relaxed iteration

- Less and less accurate solution of inner system and still converge
- General procedure for any inexact/expensive A
- Save up to 30% computational time

Alternative Perspective. Matrix Functions

An evolution problem:

$$\begin{cases} \frac{\partial u(x,y,t)}{\partial t} = \Delta u, \quad (x,y) \in (0,1)^2 \quad t \in [0, 0.1] \\ u(x,y,t) = 0, \quad (x,y) \in \partial([0,1]^2) \\ u(x,y,0) = 1, \quad (x,y) \in [0,1]^2 \end{cases}$$

Implicit Euler: $u_{i+1} = (I + \delta t A)^{-1} u_i, \quad i = 0, 1, ...$

$\Rightarrow$ linear system to be solved....

Alternative Perspective. Matrix Functions

An evolution problem:

$$\begin{cases} \frac{\partial u(x,y,t)}{\partial t} = \Delta u, & (x,y) \in (0,1)^2 \quad t \in [0, 0.1] \\ u(x,y,t) = 0, & (x,y) \in \partial([0,1]^2) \\ u(x,y,0) = 1, & (x,y) \in [0,1]^2 \end{cases}$$

Implicit Euler: $\qquad u_{i+1} = (I + \delta t A)^{-1} u_i, \quad i = 0, 1, \ldots$

$\Rightarrow$ linear system to be solved....

$$u_{i+1} = f(A) u_i, \qquad f(\lambda) = (1 + \delta t \lambda)^{-1}$$

With FOM:

$$\begin{aligned} u_{i+1} \quad &\approx \quad V_k (V_k^* (I + \delta t A) V_k)^{-1} (V_k^* u_i) \\ & \qquad V_k (I + \delta t H_k)^{-1} (V_k^* u_i) \\ & \qquad V_k f(H_k) (V_k^* u_i) \end{aligned}$$

Becoming greedy...

For this evolution problem:

$$u(t) = \exp(-tA)u_0 \quad t = 0.1$$

Exponential integrator:

$$u_{i+1} \approx V_k f(H_k)(V_k^* u_i), \qquad f(\lambda) = \exp(-tH_k)$$

Becoming greedy...

For this evolution problem:

$$u(t) = \exp(-tA)u_0 \quad t = 0.1$$

Exponential integrator:

$$u_{i+1} \approx V_k f(H_k)(V_k^* u_i), \qquad f(\lambda) = \exp(-tH_k)$$

	Euler		Exp	
step δt	CPU	error	CPU	error (#its*)
0.001	1.9	$2 \cdot 10^{-3}$	0.09	$9 \cdot 10^{-4}$(37)
0.005	0.4	$1 \cdot 10^{-2}$	0.07	$4 \cdot 10^{-3}$(28)
0.01	0.2	$2 \cdot 10^{-2}$	0.05	$1 \cdot 10^{-2}$(25)

* : Stopping criterion tolerance related to timestep

$\Rightarrow$ More general exponential integrators

More efficient approximations

- Rational function approximation

$$f(\lambda) \approx \omega_0 + \sum_{k=1}^{\nu} \frac{\omega_k}{\lambda - \xi_k}$$

Therefore

$$f(A)b \approx \omega_0 b + \sum_{k=1}^{\nu} \omega_k (A - \xi_k I)^{-1} b$$

More efficient approximations

- Rational function approximation

$$f(\lambda) \approx \omega_0 + \sum_{k=1}^{\nu} \frac{\omega_k}{\lambda - \xi_k}$$

Therefore

$$f(A)b \approx \omega_0 b + \sum_{k=1}^{\nu} \omega_k (A - \xi_k I)^{-1} b$$

- Acceleration method: Extended Krylov Subspace Method (EKSM)

$$f(A)v \approx x_k \in K_k(A, v) + K_k(A^{-1}, A^{-1}v)$$

Comparisons: CPU Time in Matlab (space dim.)

A from FD discretization of (unit square, Dirichlet hom. bc.)

$$\begin{aligned}\mathcal{L}_1(u) &= -100u_{x_1x_1} - u_{x_2x_2} + 10x_1u_{x_1}, \\ \mathcal{L}_2(u) &= -\mathrm{div}(e^{3x_1x_2}\mathrm{grad}u) + \frac{1}{x_1+x_2}u_{x_1}\end{aligned}$$

Comparisons: CPU Time in Matlab (space dim.)

A from FD discretization of (unit square, Dirichlet hom. bc.)

$$\mathcal{L}_1(u) = -100u_{x_1x_1} - u_{x_2x_2} + 10x_1u_{x_1},$$
$$\mathcal{L}_2(u) = -\mathrm{div}(e^{3x_1x_2}\mathrm{grad}u) + \frac{1}{x_1+x_2}u_{x_1}$$

f	Oper.	n	SI-Arnoldi	EKSM	Std Krylov
$\lambda^{1/2}$	$\mathcal{L}_1$	2500	0.9 (*59*)	0.6 (*48*)	7 (*193*)
		10000	4.0 (*66*)	3.6 (*68*)	*46 (*300*)
		160000	642.9(*246*)	219.7(*122*)	*458(*300*)
	$\mathcal{L}_2$	40000	41.1(*117*)	25.4(*106*)	*89 (*300*)
		160000	580.2(*442*)	231.2(*144*)	*461 (*300*)

Conclusions

- Emphasized HPC operations involving communications (except Mxv)
- Short-term recurrences preferable
- Other tricks can be used (but not usually in black-box routines)
- Many ideas have wider applicability

`http://www.dm.unibo.it/~simoncin`   `valeria@dm.unibo.it`