Recent Developments in Krylov Subspace Methods

Summerschool Linear Algebra on High-Performance Computer

Martin van Gijzen

Delft University of Technology

June 6, 2023

Outline

Introduction

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?
- For symmetric matrices this is the case:
  - CG if the system matrix A is SPD.
  - CR (or MINRES) for general symmetric matrices.

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?
- For symmetric matrices this is the case:
  - CG if the system matrix A is SPD.
  - CR (or MINRES) for general symmetric matrices.
- For nonsymmetric the situation is less clear: many algortihms have been proposed.

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?
- For symmetric matrices this is the case:
  - CG if the system matrix A is SPD.
  - CR (or MINRES) for general symmetric matrices.
- For nonsymmetric the situation is less clear: many algortihms have been proposed.
- Today I will explain why and provide an overview.

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?
- For symmetric matrices this is the case:
  - CG if the system matrix A is SPD.
  - CR (or MINRES) for general symmetric matrices.
- For nonsymmetric the situation is less clear: many algortihms have been proposed.
- Today I will explain why and provide an overview.
- I will also discuss the relatively new class of IDR(s) methods, and their software implementation

Introduction

- Many Krylov method exists, and one may wonder why. Isn't there a best method?
- For symmetric matrices this is the case:
  - CG if the system matrix A is SPD.
  - CR (or MINRES) for general symmetric matrices.
- For nonsymmetric the situation is less clear: many algortihms have been proposed.
- Today I will explain why and provide an overview.
- I will also discuss the relatively new class of IDR(s) methods, and their software implementation
- Finally I will discuss polynomial preconditioners.

Optimality and short recurrences

The CG method combines two very attractive properties:

- It is optimal: it minimizes the error (in A-norm) over the Krylov subspace.
- It uses short recurrences; the number of vectors needed to carry out the process is fixed (only three vectors are needed)

Optimality and short recurrences

The CG method combines two very attractive properties:

- It is optimal: it minimizes the error (in A-norm) over the Krylov subspace.
- It uses short recurrences; the number of vectors needed to carry out the process is fixed (only three vectors are needed)

In this sense CG is the best method. No method can be found that converges faster in A norm. Several implementations of CG exist, but three vectors of storage is the minimum.

Optimality and short recurrences

The CG method combines two very attractive properties:

- It is optimal: it minimizes the error (in A-norm) over the Krylov subspace.
- It uses short recurrences; the number of vectors needed to carry out the process is fixed (only three vectors are needed)

In this sense CG is the best method. No method can be found that converges faster in A norm. Several implementations of CG exist, but three vectors of storage is the minimum.

Can we find such a method for nonsymmetric problems?

Faber-Manteuffel theorem

Faber-Manteuffel theorem

- For a long time people sought a Krylov method for nonsymmetric systems that combine short recurrences with an optimality property for the error

Faber-Manteuffel theorem

- For a long time people sought a Krylov method for nonsymmetric systems that combine short recurrences with an optimality property for the error
- In 1984, Faber and Manteuffel proved that such a method does not exist for general nonsymmetric problems.

Optimality or short recurrences

Optimality or short recurrences

So we have to give up one of the two properties:

- We give up short recurrences. This leads to GMRES. It minimizes the residual over the Krylov subspace, but needs to store a basis for this subspace. This means that computations grow quadratic with the number of iterations, because of the orthogonalisation of the basis vectors. Also, every iteration an additional vector has to be stored.
- We give up optimality but maintain the short recurrences and fixed memory requirements.

Optimality or short recurrences

So we have to give up one of the two properties:

- We give up short recurrences. This leads to GMRES. It minimizes the residual over the Krylov subspace, but needs to store a basis for this subspace. This means that computations grow quadratic with the number of iterations, because of the orthogonalisation of the basis vectors. Also, every iteration an additional vector has to be stored.
- We give up optimality but maintain the short recurrences and fixed memory requirements. **The best methods of this type originate from Delft!**

GMRES

GMRES

- GMRES gives up the short recurrences.

GMRES

- GMRES gives up the short recurrences.
- It minimized the residual over the Krylov subspace.

GMRES

- GMRES gives up the short recurrences.
- It minimized the residual over the Krylov subspace.
- The method is easy to understand and seems to have become the best known and most popular method for nonsymmetric systems.

GMRES

- GMRES gives up the short recurrences.
- It minimized the residual over the Krylov subspace.
- The method is easy to understand and seems to have become the best known and most popular method for nonsymmetric systems.
- This doesn't mean that GMRES is actually the best method for many problems!!

GMRES, algorithm

GMRES uses the following two ingredients:

GMRES, algorithm

GMRES uses the following two ingredients:

- Build orthogonal basis for the Krylov subspace

  $$K_m(A, \mathbf{r}_0) := \mathsf{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\} \ ,$$

  with Arnoldi's method

  $$AQ_k = Q_{k+1}\underline{H}_k.$$

  $Q_k = [\mathbf{q}_1 \ \mathbf{q}_2 \cdots \mathbf{q}_k]$ has orthonormal columns, $\underline{H}_k$ Hessenberg.

GMRES, algorithm

GMRES uses the following two ingredients:

- Build orthogonal basis for the Krylov subspace

$$K_m(A, \mathbf{r}_0) := \text{span}\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\} \ ,$$

  with Arnoldi's method

$$AQ_k = Q_{k+1}\underline{H}_k.$$

  $Q_k = [\mathbf{q}_1 \ \mathbf{q}_2 \cdots \mathbf{q}_k]$ has orthonormal columns, $\underline{H}_k$ Hessenberg.
- Take $\mathbf{q}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|$. Construct solution $\mathbf{x}_k = Q_k\mathbf{y}_k$. Minimize the residual

$$\begin{aligned} \|\mathbf{r}_k\| &= \|\|\mathbf{r}_0\|\mathbf{q}_1 - AQ_k\mathbf{y}_k\| \\ &= \|\|\mathbf{r}_0\|Q_{k+1}\mathbf{e}_1 - Q_{k+1}\underline{H}_k\mathbf{y}_k\| \\ &= \|\|\mathbf{r}_0\|\mathbf{e}_1 - \underline{H}_k\mathbf{y}_k\| \end{aligned}$$

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.
- Try to reduce effects of restarting by including more information from previous cycle (GCRO, deflated restarting).

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.
- Try to reduce effects of restarting by including more information from previous cycle (GCRO, deflated restarting).
  Tries to reduce effects of a bad idea.

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.
- Try to reduce effects of restarting by including more information from previous cycle (GCRO, deflated restarting).
  Tries to reduce effects of a bad idea.
- Nested GMRES methods (GMRESR and FGMRES).
  Do inner iterations to try to avoid restarting the outer iteration.

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.
- Try to reduce effects of restarting by including more information from previous cycle (GCRO, deflated restarting).
  Tries to reduce effects of a bad idea.
- Nested GMRES methods (GMRESR and FGMRES).
  Do inner iterations to try to avoid restarting the outer iteration.
- Use very good preconditioner, so only few GMRES iterations are needed.

GMRES variants

Since GMRES is a long recurrence method, it becomes too expensive if many iterations are needed. Cures for this are:

- Restart after m iterations (standard technique).
  Bad idea. Destroys super-linear convergence, finite termination.
- Try to reduce effects of restarting by including more information from previous cycle (GCRO, deflated restarting).
  Tries to reduce effects of a bad idea.
- Nested GMRES methods (GMRESR and FGMRES).
  Do inner iterations to try to avoid restarting the outer iteration.
- Use very good preconditioner, so only few GMRES iterations are needed.
  Very good preconditioners are not easy to parallelise.

BiCG and Bi-Lanczos

BiCG and Bi-Lanczos

- The Bi-Lanczos method constructs bases for a left and a right Krylov subspace:

$$\text{span}\{\mathbf{v}_1 \cdots \mathbf{v}_k\} \text{ is a basis for } K^k(A; \mathbf{v}_1)$$

$$\text{span}\{\mathbf{w}_1 \cdots \mathbf{w}_k\} \text{ is a basis for } K^k(A^T; \mathbf{w}_1)$$

With

$$V_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ldots \mathbf{v}_k] \quad W_k = [\mathbf{w}_1 \ \mathbf{w}_2 \ldots \mathbf{w}_k] \ .$$

the Bi-Lanczos process is described by

$$AV_k = V_k T_k + \delta_{k+1} \mathbf{v}_{k+1} \mathbf{e}_k^T$$

$$A^T W_k = W_k T_k^T + \beta_{k+1} \mathbf{w}_{k+1} \mathbf{e}_k^T .$$

T_k is tridiagonal and such that $W_k^T V_k = I$.

BiCG and Bi-Lanczos

- The Bi-Lanczos method constructs bases for a left and a right Krylov subspace:

  $$\text{span}\{\mathbf{v}_1 \cdots \mathbf{v}_k\} \text{ is a basis for } K^k(A;\mathbf{v}_1)$$

  $$\text{span}\{\mathbf{w}_1 \cdots \mathbf{w}_k\} \text{ is a basis for } K^k(A^T;\mathbf{w}_1)$$

  With

  $$V_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ldots \mathbf{v}_k] \quad W_k = [\mathbf{w}_1 \ \mathbf{w}_2 \ldots \mathbf{w}_k] \ .$$

  the Bi-Lanczos process is described by

  $$AV_k = V_k T_k + \delta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^T$$

  $$A^T W_k = W_k T_k^T + \beta_{k+1}\mathbf{w}_{k+1}\mathbf{e}_k^T .$$

  T_k is tridiagonal and such that $W_k^T V_k = I$.
- BiCG takes residuals $\mathbf{r}_k = \mathbf{v}_{k+1}$. These are orthogonal to the vectors $\mathbf{w}_k$.

The CG algorithm

Regular steps in CG algorithm:

$$\begin{aligned}
&\rho_n = r_n^T r_n,\ \beta_n = \rho_n/\rho_{n-1}\\
&p_n = r_n + \beta_n p_{n-1};\\
&\sigma_n = p_n^T A p_n,\ \alpha_n = \rho_n/\sigma_n\\
&r_{n+1} = r_n - \alpha_n A p_n;\\
&x_{n+1} = x_n + \alpha_n p_n
\end{aligned}$$

The Bi-CG algorithm

Regular steps in Bi-CG algorithm:

$$
\begin{aligned}
&\rho_n = \tilde{r}_n^T r_n,\ \beta_n = \rho_n/\rho_{n-1} \\
&p_n = r_n + \beta_n p_{n-1}, \\
&\tilde{p}_n = \tilde{r}_n + \beta_n \tilde{p}_{n-1}, \\
&\sigma_n = \tilde{p}_n^T A p_n,\ \alpha_n = \rho_n/\sigma_n \\
&r_{n+1} = r_n - \alpha_n A p_n; \\
&\tilde{r}_{n+1} = \tilde{r}_n - \alpha_n A^T \tilde{p}_n \\
&x_{n+1} = x_n + \alpha_n p_n
\end{aligned}
$$

Bi-CG wastes time

Bi-CG constructs its approximations as a linear combination of the search directions. The only reason why we construct shadow vectors is to calculate the parameters α_n and β_n.

Another disadvantage of Bi-CG is that operations with A^T have to be performed, and this may be difficult in matrix-free computations.

Next we look at the following questions:

- Can the computational work of Bi-CG be better exploited?
- Can operations with A^T be avoided?

Krylov methods as polynomial methods

Krylov methods as polynomial methods

- Every vector in a Krylov subspace can be written as a polynomial in the matrix A times the initial vector.

Krylov methods as polynomial methods

- Every vector in a Krylov subspace can be written as a polynomial in the matrix A times the initial vector.
- So we can write for the BiCG residuals

$$\mathbf{r}_k^{BiCG} = \Psi_k(A)\mathbf{r}_0$$

For BiCG, $\Psi(A)$ is the CG-polynomial (good!).

Krylov methods as polynomial methods

- Every vector in a Krylov subspace can be written as a polynomial in the matrix A times the initial vector.
- So we can write for the BiCG residuals

$$\mathbf{r}_k^{BiCG} = \Psi_k(A)\mathbf{r}_0$$

  For BiCG, $\Psi(A)$ is the CG-polynomial (good!).
- In BiCG we have to make the residual $\mathbf{r}_k^{BiCG}$ orthogonal to $\mathbf{w}_k \in K^k(A^T;\mathbf{w}_1)$, for this we have to compute an inner product $\mathbf{w}_k^T\mathbf{r}_k^{BiCG}$

Krylov methods as polynomial methods

- Every vector in a Krylov subspace can be written as a polynomial in the matrix A times the initial vector.
- So we can write for the BiCG residuals

  $$\mathbf{r}_k^{BiCG} = \Psi_k(A)\mathbf{r}_0$$

  For BiCG, $\Psi(A)$ is the CG-polynomial (good!).
- In BiCG we have to make the residual $\mathbf{r}_k^{BiCG}$ orthogonal to $\mathbf{w}_k \in K^k(A^T; \mathbf{w}_1)$, for this we have to compute an inner product $\mathbf{w}_k^T \mathbf{r}_k^{BiCG}$
- Since $\mathbf{w}_k \in K^k(A^T; \mathbf{w}_1)$ we can write this vector as $\mathbf{w}_k = \Phi_k(A^T)\mathbf{w}_1$. In standard BiCG we use the CG-polynomial ($\Phi_k(t) = \Psi_k(t)$) to generate the left Krylov vectors.

Krylov methods as polynomial methods

- Every vector in a Krylov subspace can be written as a polynomial in the matrix A times the initial vector.
- So we can write for the BiCG residuals

$$\mathbf{r}_k^{BiCG} = \Psi_k(A)\mathbf{r}_0$$

  For BiCG, $\Psi(A)$ is the CG-polynomial (good!).
- In BiCG we have to make the residual $\mathbf{r}_k^{BiCG}$ orthogonal to $\mathbf{w}_k \in K^k(A^T; \mathbf{w}_1)$, for this we have to compute an inner product $\mathbf{w}_k^T \mathbf{r}_k^{BiCG}$
- Since $\mathbf{w}_k \in K^k(A^T; \mathbf{w}_1)$ we can write this vector as $\mathbf{w}_k = \Phi_k(A^T)\mathbf{w}_1$. In standard BiCG we use the CG-polynomial ($\Phi_k(t) = \Psi_k(t)$) to generate the left Krylov vectors.
- But this is not essential, the $\mathbf{w}_k$ only have to form a basis for $K^k(A^T; \mathbf{w}_1)$.

A briljant idea

A briljant idea

- Remember we have to calculate inner products like

$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = (\Phi_k(A^T)\mathbf{w}_1)^T \Psi_k(A)\mathbf{r}_0$$

A briljant idea

- Remember we have to calculate inner products like

$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = (\Phi_k(A^T)\mathbf{w}_1)^T \Psi_k(A)\mathbf{r}_0$$

- But this can also be writen as

$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = \mathbf{w}_1^T (\Phi_k(A)\Psi_k(A))\mathbf{r}_0$$

A briljant idea

- Remember we have to calculate inner products like
$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = (\Phi_k(A^T)\mathbf{w}_1)^T \Psi_k(A)\mathbf{r}_0$$
- But this can also be writen as
$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = \mathbf{w}_1^T (\Phi_k(A)\Psi_k(A))\mathbf{r}_0$$
- Idea: choose $\Phi_k(A)$ such that it reduces the norm of $\Psi(A)\mathbf{r}_0$, an extra reduction for free! For this you have to make an algorithm that computes residual vectors $\mathbf{r}_{2k} = (\Phi_k(A)\Psi_k(A))\mathbf{r}_0$.

A briljant idea

- Remember we have to calculate inner products like
$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = (\Phi_k(A^T)\mathbf{w}_1)^T \Psi_k(A)\mathbf{r}_0$$
- But this can also be writen as
$$\mathbf{w}_k^T \mathbf{r}_k^{BiCG} = \mathbf{w}_1^T (\Phi_k(A)\Psi_k(A))\mathbf{r}_0$$
- Idea: choose $\Phi_k(A)$ such that it reduces the norm of $\Psi(A)\mathbf{r}_0$, an extra reduction for free! For this you have to make an algorithm that computes residual vectors $\mathbf{r}_{2k} = (\Phi_k(A)\Psi_k(A))\mathbf{r}_0$.
- This idea revolutionized the Krylov methods, many new methods have been proposed using different choices for $\Phi_k(A)$.

Hybrid BiCG methods

- CGS (Sonneveld, 1989)
- BiCGSTAB (Van der Vorst, 1992)
- BiCGstab(ℓ) (Sleijpen and Fokkema, 1994)
- GPBiCG (Zhang, 1997)
- Many other variants

Peter Sonneveld

CGS

CGS

- CGS applies the CG polynomial twice:

$$\mathbf{r}_{2k} = \Psi_k^2(A)\mathbf{r}_0$$

CGS

- CGS applies the CG polynomial twice:

$$\mathbf{r}_{2k} = \Psi_k^2(A)\mathbf{r}_0$$

- For many problems CGS converges considerably faster than Bi-CG, sometimes twice as fast.

  However, convergence is more erratic. Bi-CG shows peaks in the convergence curves. These peaks are squared by CGS.

  The peaks can destroy the accuracy of the solution and also convergence of the method. The search for a more smoothly converging method has led to the development of Bi-CGSTAB by Van der Vorst.

BiCGSTAB

BiCGSTAB

- BiCGSTAB combines BiCG iterations with linear minimal residual steps:

$$\mathbf{r}_{2k} = \Omega_k(A)\Psi_k(A)\mathbf{r}_0$$

with

$$\Omega_k(t) = (1-\omega_k t)(1-\omega_{k-1}t)\cdots(1-\omega_1 t),\ \Omega_0(t) \equiv 1.$$

BiCGSTAB

- BiCGSTAB combines BiCG iterations with linear minimal residual steps:

$$\mathbf{r}_{2k} = \Omega_k(A)\Psi_k(A)\mathbf{r}_0$$

with

$$\Omega_k(t) = (1-\omega_k t)(1-\omega_{k-1}t)\cdots(1-\omega_1 t),\ \Omega_0(t) \equiv 1.$$

- Is this a good idea?

BiCGSTAB

- BiCGSTAB combines BiCG iterations with linear minimal residual steps:

$$\mathbf{r}_{2k} = \Omega_k(A)\Psi_k(A)\mathbf{r}_0$$

with

$$\Omega_k(t) = (1-\omega_k t)(1-\omega_{k-1}t)\cdots(1-\omega_1 t),\ \Omega_0(t) \equiv 1.$$

- Is this a good idea?
- That depends if you can find good parameters ω. If the problem is easy (real eigenvalues in the right half plane, well conditioned) this works very well. If the problem is indefinite, the polynomial $\Omega_k(t)$ will not be residual reducing, and BiCGSTAB may converge *slower* than BiCG and become unstable.

BiCGSTAB

- BiCGSTAB combines BiCG iterations with linear minimal residual steps:

$$\mathbf{r}_{2k} = \Omega_k(A)\Psi_k(A)\mathbf{r}_0$$

with

$$\Omega_k(t) = (1 - \omega_k t)(1 - \omega_{k-1} t) \cdots (1 - \omega_1 t),\ \Omega_0(t) \equiv 1.$$

- Is this a good idea?
- That depends if you can find good parameters ω. If the problem is easy (real eigenvalues in the right half plane, well conditioned) this works very well. If the problem is indefinite, the polynomial $\Omega_k(t)$ will not be residual reducing, and BiCGSTAB may converge *slower* than BiCG and become unstable.
- BiCGSTAB is, of the short recurrence methods, by far the most popular.

Variants, extensions

We mention a few variants and extension that can be made to any Krylov method:

Variants, extensions

We mention a few variants and extension that can be made to any Krylov method:

- Quasi-minimization: igore that the basis of the Krylov subspace generated by Bi-Lanczos is nonorthogonal, and then "quasi" minimize. The BiCG variant of this is called QMR.

Variants, extensions

We mention a few variants and extension that can be made to any Krylov method:

- Quasi-minimization: igore that the basis of the Krylov subspace generated by Bi-Lanczos is nonorthogonal, and then "quasi" minimize. The BiCG variant of this is called QMR.
- Flexible Krylov methods: they allow a variable preconditioner, for example another Krylov method. Examples: FGMRES and GMRESR.

Variants, extensions

We mention a few variants and extension that can be made to any Krylov method:

- Quasi-minimization: igore that the basis of the Krylov subspace generated by Bi-Lanczos is nonorthogonal, and then "quasi" minimize. The BiCG variant of this is called QMR.
- Flexible Krylov methods: they allow a variable preconditioner, for example another Krylov method. Examples: FGMRES and GMRESR.
- Multi-shift methods: Can solve shifted systems simultaneously by making use of shift invariance of Krylov subspaces.

Multi-shift Krylov methods (1)

Several applications require the solution of sequences of shifted systems

$$(A - s_i I)x_i = b \quad i = 1, \cdots, n_s$$

Examples are

- Model Order Reduction
- Helmholtz problems
- Quantum Chromo-Dynamics (QCD)
- Tykhonov Regularization
- PageRank

Note that only the shift is different from system to system.

Multi-shift Krylov methods (2)

The vector b is the same for the all the systems. Since

$$K_m(A, b) := \text{span}\{b, Ab, \cdots, A^{m-1}b\} = K_m((A - sI), b)$$

we can use the same basis for $K_m(A, b)$ to solve a sequence of shifted systems.

Multi-shift Krylov methods (2)

The vector b is the same for the all the systems. Since

$$K_m(A,b) := \text{span}\{b, Ab, \cdots, A^{m-1}b\} = K_m((A - sI), b)$$

we can use the same basis for $K_m(A,b)$ to solve a sequence of shifted systems.

We illustrate this for GMRES. Use the Arnoldi relation

$$AV_m = V_{m+1}\underline{H}_m \ , \ \ V_m^T V_m = I_m \ , \ \ V_m e_1 = \beta b$$

to build a basis for $K_m(A,b)$.

Multi-shift Krylov methods (2)

The vector b is the same for the all the systems. Since

$$K_m(A,b) := \text{span}\{b, Ab, \cdots, A^{m-1}b\} = K_m((A-sI),b)$$

we can use the same basis for $K_m(A,b)$ to solve a sequence of shifted systems.

We illustrate this for GMRES. Use the Arnoldi relation

$$AV_m = V_{m+1}\underline{H}_m \;, \;\; V_m^T V_m = I_m \;, \;\; V_m e_1 = \beta b$$

to build a basis for $K_m(A,b)$. Substitute a solution $x_{s,m} = V_m y_s$ and use the Arnoldi relation. This gives

$$\min_{x_{s,m}} \|b - (A-sI)x_{s,m}\| \Rightarrow \min_{y_s} \|\beta e_1 - (\underline{H}_m - s\underline{I}_m)y_s\|$$

in which $\underline{I}$ is the identity with a zero row appended.

IDR(s).

IDR(s).

- The hybrid Krylov methods like BiCGSTAB are mathematically quite elegant and often effective for not-too-bad problems (read: well preconditioned). But does the story end here?

IDR(s).

- The hybrid Krylov methods like BiCGSTAB are mathematically quite elegant and often effective for not-too-bad problems (read: well preconditioned). But does the story end here?
- In 2007 Peter and I developed IDR(s). This method showed to be more robust and often much more efficient for difficult problems.

IDR(s).

- The hybrid Krylov methods like BiCGSTAB are mathematically quite elegant and often effective for not-too-bad problems (read: well preconditioned). But does the story end here?
- In 2007 Peter and I developed IDR(s). This method showed to be more robust and often much more efficient for difficult problems.
- The idea was to construct residuals in a sequence of nested subspaces of shrinking dimension. Here I will explain the method in a more traditional setting using Krylov subspaces.

The left Krylov subspace.

The left Krylov subspace.

- In BiCG you create residuals that are orthogonal to the left Krylov subspace. The computations for computing the basis for the left Krylov subspace do not contribute to the reduction of the residual.

The left Krylov subspace.

- In BiCG you create residuals that are orthogonal to the left Krylov subspace. The computations for computing the basis for the left Krylov subspace do not contribute to the reduction of the residual.
- Idea: create a block Krylov subpace $K^k(A^T; P)$:

$$W_{js} = [\mathbf{p}_1 \cdots \mathbf{p}_s; (I - \omega_j A^T)\mathbf{p}_1 \cdots (I - \omega_j A^T)\mathbf{p}_s; \cdots]$$

The left Krylov subspace.

- In BiCG you create residuals that are orthogonal to the left Krylov subspace. The computations for computing the basis for the left Krylov subspace do not contribute to the reduction of the residual.
- Idea: create a block Krylov subpace $K^k(A^T;P)$:

$$W_{js} = [\mathbf{p}_1 \cdots \mathbf{p}_s; (I-\omega_j A^T)\mathbf{p}_1 \cdots (I-\omega_j A^T)\mathbf{p}_s; \cdots]$$

- Again we have to compute inner products $\mathbf{w}_{js+i}^T \mathbf{r}_{js+i}, i = 1, \cdots, s$:

$$\mathbf{w}_{js+i}^T \mathbf{r}_{js+i} = ((I-\omega_j A^T)^j \mathbf{p}_i)^T \mathbf{r}_{js+i}, i = 1, \cdots s$$

What is IDR(s) and why is it better than BiCGSTAB?

What is IDR(s) and why is it better than BiCGSTAB?

- Now we can do the inner-product trick:

$$\mathbf{w}_{js+i}^T \mathbf{r}_{js+i} = \mathbf{p}_i^T (I - \omega_j A)^j \mathbf{r}_{js+i}, i = 1, \cdots s$$

What is IDR(s) and why is it better than BiCGSTAB?

- Now we can do the inner-product trick:

$$\mathbf{w}_{js+i}^T \mathbf{r}_{js+i} = \mathbf{p}_i^T (I - \omega_j A)^j \mathbf{r}_{js+i}, i = 1, \cdots s$$

- So we can increase the dimension of the left (block) Krylov subspace by blocksize s at the cost of one multiplication with A!!

What is IDR(s) and why is it better than BiCGSTAB?

- Now we can do the inner-product trick:

$$\mathbf{w}_{js+i}^T \mathbf{r}_{js+i} = \mathbf{p}_i^T (I - \omega_j A)^j \mathbf{r}_{js+i}, i = 1, \cdots s$$

- So we can increase the dimension of the left (block) Krylov subspace by blocksize s at the cost of one multiplication with A!!
- For the IDR(s) residuals we get:

$$r_{k+j} = \Omega_j(A)\Psi_k(A) r_0$$

The degree of the CG polynomial grows much faster in dimension than the (poorer) local minimal residual polynomial Ω.

What is IDR(s) and why is it better than BiCGSTAB?

What is IDR(s) and why is it better than BiCGSTAB?

- This makes the algorithm more efficient (number of matvecs is often reduced by factor 2 for modest s), and more robust because the effect of a polynomial $\Omega_j(A)$ is much smaller.

What is IDR(s) and why is it better than BiCGSTAB?

- This makes the algorithm more efficient (number of matvecs is often reduced by factor 2 for modest s), and more robust because the effect of a polynomial $\Omega_j(A)$ is much smaller.
- Some properties:
  - Finite termination in $N + N/s$ iterations ($=$ MATVECS)
  - Recursion of length $s + 2$.
  - Memory requirement: $3s + 4$ vectors.
  - Underlying Hessenberg matrix has upper-bandwidth s.

Software package (1)

Four main routines:

- `idrs`: Highly optimised
  - `idr(1)` with proper parameters equivalent to `bicgstab`
  - Includes recycling and extraction of spectral information
- `qmridr`: Robust, flexible
  - Equivalent to flexible `GMRES` in first s steps.
  - Can use `idrs` as inner iteration method.
- `msidrs`: Highly optimised, solves the multishift problem.
- `msqmridr`: Robust, flexible, solves the multishift problem.
  - Can use `msidrs` as inner iteration method.

Software package (2)

Software package (2)

- (Modern) Fortran and Matlab versions

Software package (2)

- (Modern) Fortran and Matlab versions
- Comes with many examples:

Software package (2)

- (Modern) Fortran and Matlab versions
- Comes with many examples:
  - Matrix Market interface to test your own favourite examples.

Software package (2)

- (Modern) Fortran and Matlab versions
- Comes with many examples:
  - Matrix Market interface to test your own favourite examples.
  - Dense matrix (PageRank) example

Software package (2)

- (Modern) Fortran and Matlab versions
- Comes with many examples:
  - Matrix Market interface to test your own favourite examples.
  - Dense matrix (PageRank) example
  - Matrix free (convection-diffusion-reaction) example

Software package (2)

- (Modern) Fortran and Matlab versions
- Comes with many examples:
  - Matrix Market interface to test your own favourite examples.
  - Dense matrix (PageRank) example
  - Matrix free (convection-diffusion-reaction) example
- *https://github.com/mbvangijzen/IDRS-package*

Software package (3)

Details Fortran implementation:

Software package (3)

Details Fortran implementation:

- Users needs to supply modules for matvec and preconditioning operations. These operations should be overloaded to $y = A * x$ and $y = x/M$, respectively. Several examples are supplied.

Software package (3)

Details Fortran implementation:

- Users needs to supply modules for matvec and preconditioning operations. These operations should be overloaded to $y = A * x$ and $y = x/M$, respectively. Several examples are supplied.
- Routines work for complex and real (due to overloading). Precision for complex and real arithmatic can be set by two parameters `cp` and `rp`.

Software package (3)

Details Fortran implementation:

- Users needs to supply modules for matvec and preconditioning operations. These operations should be overloaded to $y = A * x$ and $y = x/M$, respectively. Several examples are supplied.
- Routines work for complex and real (due to overloading). Precision for complex and real arithmatic can be set by two parameters `cp` and `rp`.
- Communication avoiding, one global synchronisation point per iteration.

Software package (3)

Details Fortran implementation:

- Users needs to supply modules for matvec and preconditioning operations. These operations should be overloaded to $y = A * x$ and $y = x/M$, respectively. Several examples are supplied.
- Routines work for complex and real (due to overloading). Precision for complex and real arithmatic can be set by two parameters `cp` and `rp`.
- Communication avoiding, one global synchronisation point per iteration.
- Parallelisation using co-array Fortran

Polynomial preconditioners

Polynomial preconditioners

- The idea of a polynomial preconditioner is to construct a polynomial of degree n, $P_n(A)$ such that

$$P_n(A) \approx A^{-1}$$

Polynomial preconditioners

- The idea of a polynomial preconditioner is to construct a polynomial of degree n, $P_n(A)$ such that

$$P_n(A) \approx A^{-1}$$

- A simple idea is to use the geometric series

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$$

  Using for x the matrix $I - A$ gives A^{-1} (if the series converges).

Polynomial preconditioners

- The idea of a polynomial preconditioner is to construct a polynomial of degree n, $P_n(A)$ such that

$$P_n(A) \approx A^{-1}$$

- A simple idea is to use the geometric series

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$$

  Using for x the matrix $I - A$ gives A^{-1} (if the series converges).
- Truncating the series after n terms gives a preconditioner.

Polynomial preconditioners

- The idea of a polynomial preconditioner is to construct a polynomial of degree n, $P_n(A)$ such that

$$P_n(A) \approx A^{-1}$$

- A simple idea is to use the geometric series

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$$

  Using for x the matrix $I - A$ gives A^{-1} (if the series converges).
- Truncating the series after n terms gives a preconditioner.
- This preconditioner is applied by repeated matrix-vector multiplications, powers of $(I - A)$ are not calculated.

Other polynomial preconditioners

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.
- Many other polynomial preconditioners exist. They are usually generated as follows:

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.
- Many other polynomial preconditioners exist. They are usually generated as follows:
  - Generate a residual polynomial $r(A) = 1 - Ap(A)$ that is small on the spectrum of A. Examples:

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.
- Many other polynomial preconditioners exist. They are usually generated as follows:
  - Generate a residual polynomial $r(A) = 1 - Ap(A)$ that is small on the spectrum of A. Examples:
    - Chebyshev polynomials scaled to an interval that contains the spectrum has minimal maximum on that interval.

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.
- Many other polynomial preconditioners exist. They are usually generated as follows:
  - Generate a residual polynomial $r(A) = 1 - Ap(A)$ that is small on the spectrum of A. Examples:
    - Chebyshev polynomials scaled to an interval that contains the spectrum has minimal maximum on that interval.
    - GMRES generates residuals that minimize the residual norm. By extracting this residual polynomial one can construct a polynomial preconditioner.

Other polynomial preconditioners

- The Neumann preconditioner discussed before is very simple, but only works well if the geometric series converges. A simple modification is to generate a geometric series for $I - \omega A$.
- Many other polynomial preconditioners exist. They are usually generated as follows:
  - Generate a residual polynomial $r(A) = 1 - Ap(A)$ that is small on the spectrum of A. Examples:
    - Chebyshev polynomials scaled to an interval that contains the spectrum has minimal maximum on that interval.
    - GMRES generates residuals that minimize the residual norm. By extracting this residual polynomial one can construct a polynomial preconditioner.
  - Take $p(A) = A^{-1}(1 - r(A)), p(A)$ is then an approximation for A^{-1}.

Why polynomial preconditioners?

Why polynomial preconditioners?

- Many people are skeptical about polynomial preconditioner: Krylov methods generate (near) optimal residual polynomials, so why combine with another polynomial?

Why polynomial preconditioners?

- Many people are skeptical about polynomial preconditioner: Krylov methods generate (near) optimal residual polynomials, so why combine with another polynomial?
- Advantages of polynomial preconditioners:

Why polynomial preconditioners?

- Many people are skeptical about polynomial preconditioner: Krylov methods generate (near) optimal residual polynomials, so why combine with another polynomial?
- Advantages of polynomial preconditioners:
  - Maybe more matvecs, but less vector operations. Advantage for example for GMRES where vector overhead is big.

Why polynomial preconditioners?

- Many people are skeptical about polynomial preconditioner: Krylov methods generate (near) optimal residual polynomials, so why combine with another polynomial?
- Advantages of polynomial preconditioners:
  - Maybe more matvecs, but less vector operations. Advantage for example for GMRES where vector overhead is big.
  - Matvecs have often better parallel performance than inner products. Matvecs only require communication with neighbours, inner products with all processors.

Why polynomial preconditioners?

- Many people are skeptical about polynomial preconditioner: Krylov methods generate (near) optimal residual polynomials, so why combine with another polynomial?
- Advantages of polynomial preconditioners:
  - Maybe more matvecs, but less vector operations. Advantage for example for GMRES where vector overhead is big.
  - Matvecs have often better parallel performance than inner products. Matvecs only require communication with neighbours, inner products with all processors.
  - Basically it is the only preconditioner that works for solving shifted systems.

Software package (4)

Software package (4)

- IDRS-package includes a Fortran module for polynomial preconditioning

Software package (4)

- IDRS-package includes a Fortran module for polynomial preconditioning
- Includes Chebyshev and Neumann preconditioners

Software package (4)

- IDRS-package includes a Fortran module for polynomial preconditioning
- Includes Chebyshev and Neumann preconditioners
- Works with all IDRS methods, including for shifted problems
- Users only need to supply matrix-vector routine.