A short course on: Preconditioned Krylov subspace methods

Yousef Saad
University of Minnesota
Dept. of Computer Science and Engineering

Universite du Littoral, Jan 19-30, 2005

Outline

Part 1

- Introd., discretization of PDEs
- Sparse matrices and sparsity
- Basic iterative methods (Relaxation..)

Part 2

- Projection methods
- Krylov subspace methods

Part 3

- Preconditioned iterations
- Preconditioning techniques
- Parallel implementations

Part 4

- Eigenvalue problems
- Applications –

PROJECTION METHODS FOR LINEAR SYSTEMS

THE PROBLEM

We consider the linear system

$$Ax = b$$

where A is $N \times N$ and can be

- Real symmetric positive definite
- Real nonsymmetric
- Complex

Focus: A is large and sparse, possibly with an irregular structure

PROJECTION METHODS

Initial Problem: $b - Ax = 0$

Given two subspaces K and L of $\mathbb{R}^N$ de£ne the approximate problem:

Find $\tilde{x} \in K$ such that $b - A\tilde{x} \perp L$

➨ **Leads to a small linear system ('projected problems') This is a basic projection step. Typically: sequence of such steps are applied**

➨ **With a nonzero initial guess x_0, the approximate problem is**

Find $\tilde{x} \in x_0 + K$ such that $b - A\tilde{x} \perp L$

Write $\tilde{x} = x_0 + \delta$ and $r_0 = b - Ax_0$. Leads to a system for δ:

Find $\delta \in K$ such that $r_0 - A\delta \perp L$

Matrix representation:

Let

- $V = [v_1, \ldots, v_m]$ **a basis of** K **&**
- $W = [w_1, \ldots, w_m]$ **a basis of** L

Then letting x **be the approximate solution** $\tilde{x} = x_0 + \delta \equiv x_0 + Vy$ **where** y **is a vector of** $\mathbb{R}^m$**, the Petrov-Galerkin condition yields,**

$$W^T(r_0 - AVy) = 0$$

and therefore

$$\tilde{x} = x_0 + V[W^T AV]^{-1} W^T r_0$$

Remark: In practice $W^T AV$ **is known from algorithm and has a simple structure [tridiagonal, Hessenberg,..]**

PROTOTYPE PROJECTION METHOD

Until Convergence Do:

1. **Select a pair of subspaces K, and L;**
2. **Choose bases $V = [v_1, \ldots, v_m]$ for K and $W = [w_1, \ldots, w_m]$ for L.**
3. **Compute**

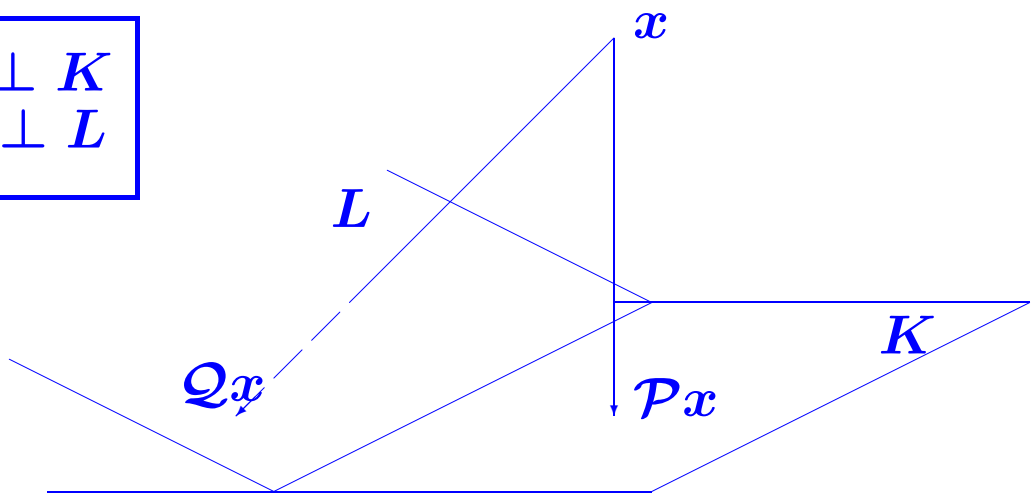
$$r \leftarrow b - Ax,$$
$$y \leftarrow (W^T A V)^{-1} W^T r,$$
$$x \leftarrow x + V y.$$

OPERATOR FORM REPRESENTATION

Let $\mathcal{P}$ be the orthogonal projector onto K and

$\mathcal{Q}$ the (oblique) projector onto K and orthogonally to L.

$$\begin{aligned} \mathcal{P}x &\in K, \quad x - \mathcal{P}x \perp K \\ \mathcal{Q}x &\in K, \quad x - \mathcal{Q}x \perp L \end{aligned}$$



The $\mathcal{P}$ and $\mathcal{Q}$ projectors

Approximate problem amounts to solving

$$\mathcal{Q}(b - Ax) = 0, \quad x \quad \in K$$

or in operator form

$$\mathcal{Q}(b - A\mathcal{P}x) = 0$$

Question: what accuracy can one expect?

Let x^* be the exact solution. Then

1) we cannot get better accuracy than $\|(I - \mathcal{P})x^*\|_2$, i.e.,

$$\|\tilde{x} - x^*\|_2 \geq \|(I - \mathcal{P})x^*\|_2$$

2) the residual of the *exact solution* for the *approximate problem* satisfies:

$$\|b - \mathcal{Q}A\mathcal{P}x^*\|_2 \leq \|\mathcal{Q}A(I - \mathcal{P})\|_2 \|(I - \mathcal{P})x^*\|_2$$

Two Important Particular Cases.

1. $\boxed{L = AK}$. Then $\|b - A\tilde{x}\|_2 = \min_{z \in K} \|b - Az\|_2$

   $\rightarrow$ Class of Minimal Residual Methods: CR, GCR, ORTHOMIN, GMRES, CGNR, ...

2. $\boxed{L = K}$ $\rightarrow$ Class of Galerkin or Orthogonal projection methods.

   When A is SPD then $\|x^* - \tilde{x}\|_A = \min_{z \in K} \|x^* - z\|_A$.

One-dimensional projection processes

$$K = span\{d\}$$
and
$$L = span\{e\}$$

Then $\tilde{x} \leftarrow x + \alpha d$ and Petrov-Galerkin condition $r - A\delta \perp e$ yields

$$\boxed{\alpha = \frac{(r,e)}{(Ad,e)}}$$

Three popular choices:

(I) Steepest descent. A is SPD. Take at each step $d = r$ and $e = r$.

Iteration:

$$\boxed{\begin{array}{l} r \leftarrow b - Ax, \\ \alpha \leftarrow (r,r)/(Ar,r) \\ x \leftarrow x + \alpha r \end{array}}$$

➨ **Each step minimizes $f(x) = \|x - x^*\|_A^2 = (A(x - x^*), (x - x^*))$ in direction $-\nabla f$. Convergence guaranteed if A is SPD.**

(II) Residual norm steepest descent . A is arbitrary (nonsingular). Take at each step $d = A^T r$ and $e = Ad$.

Iteration:

$$
\begin{aligned}
&r \leftarrow b - Ax, d = A^T r \\
&\alpha \leftarrow \|d\|_2^2 / \|Ad\|_2^2 \\
&x \leftarrow x + \alpha d
\end{aligned}
$$

- **Each step minimizes $f(x) = \|b - Ax\|_2^2$ in direction $-\nabla f$.**
- **Important Note: equivalent to usual steepest descent applied to normal equations $A^T Ax = A^T b$.**
- **Converges under the condition that A is nonsingular.**

(III) Minimal residual iteration. A positive definite ($A + A^T$ is SPD).

Take at each step $d = r$ and $e = Ar$.

Iteration:

$$
\begin{aligned}
&r \leftarrow b - Ax, \\
&\alpha \leftarrow (Ar, r)/(Ar, Ar) \\
&x \leftarrow x + \alpha r
\end{aligned}
$$

- **Each step minimizes $f(x) = \|b - Ax\|_2^2$ in direction r.**
- **Converges under the condition that $A + A^T$ is SPD.**

KRYLOV SUBSPACE METHODS

Principle: Projection methods on Krylov subspaces, i.e., on

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$$

- probably the most important class of iterative methods.
- many variants exist depending on the subspace L.

Simple properties of K_m. Let $\mu =$ deg. of minimal polynomial of v

- $K_m = \{p(A)v | p =$ polynomial of degree $\leq m-1\}$
- $K_m = K_\mu$ for all $m \geq \mu$. Moreover, K_μ is invariant under A.
- $dim(K_m) = m$ iff $\mu \geq m$.

A little review: Gram-Schmidt process

→ **Goal: given $X = [x_1, \ldots, x_m]$ compute an orthonormal set $Q = [q_1, \ldots, q_m]$ which spans the same susbpace.**

ALGORITHM : 1. ***Classical Gram-Schmidt***

1. ***For*** $j = 1, ..., m$ ***Do:***
2. ***Compute*** $r_{ij} = (x_j, q_i)$ ***for*** $i = 1, \ldots, j-1$
3. ***Compute*** $\hat{q}_j = x_j - \sum_{i=1}^{j-1} r_{ij} q_i$
4. $r_{jj} = \|\hat{q}_j\|_2$ ***If*** $r_{jj} == 0$ ***exit***
5. $q_j = \hat{q}_j / r_{jj}$
6. ***EndDo***

ALGORITHM : 2. *Modified Gram-Schmidt*

1. **For** $j = 1, ..., m$ **Do:**
2. $\hat{q}_j := x_j$
3. **For** $i = 1, \ldots, j-1$ **Do**
4. $r_{ij} = (\hat{q}_j, q_i)$
5. $\hat{q}_j := \hat{q}_j - r_{ij} q_i$
6. **EndDo**
7. $r_{jj} = \|\hat{q}_j\|_2$. **If** $r_{jj} == 0$ **exit**
8. $q_j := \hat{q}_j / r_{jj}$
9. **EndDo**

Let:

$X = [x_1, \ldots, x_m]$ ($n \times m$ matrix)

$Q = [q_1, \ldots, q_m]$ ($n \times m$ matrix)

$R = \{r_{ij}\}$ ($m \times m$ upper triangular matrix)

➨ **At each step,**

$$x_j = \sum_{i=1}^{j} r_{ij} q_i$$

Result:

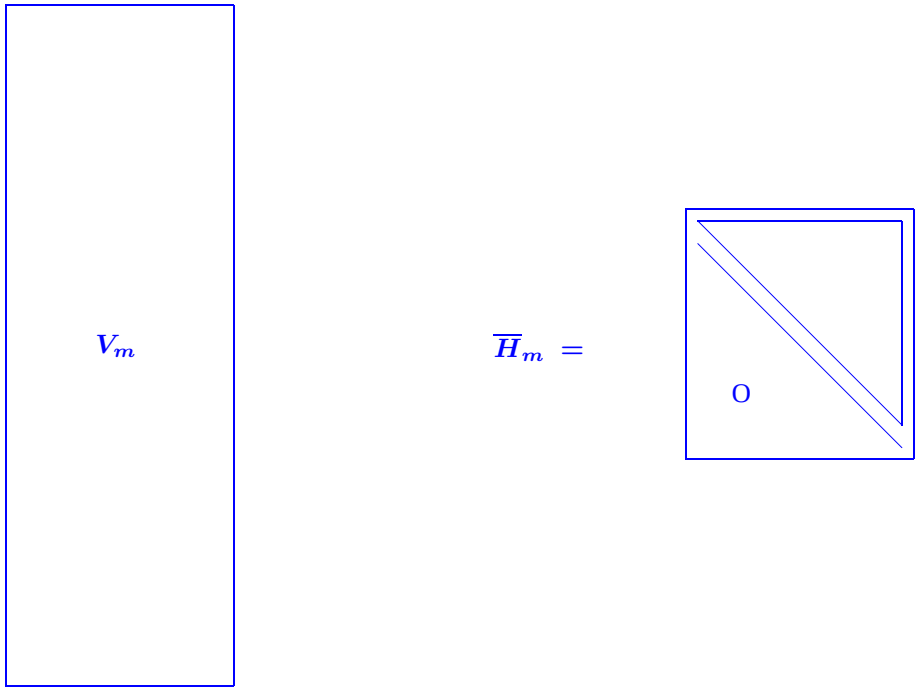
$$X = QR$$

ARNOLDI'S ALGORITHM

- **Goal: to compute an orthogonal basis of K_m.**
- **Input: Initial vector v_1, with $\|v_1\|_2 = 1$ and m.**

For $j = 1, ..., m$ do

- **Compute $w := Av_j$**
- **for $i = 1, \ldots, j$, do** $\begin{cases} h_{i,j} := (w, v_i) \\ w := w - h_{i,j} v_i \end{cases}$
- **$h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$**



Result of orthogonalization process (Arnoldi's algorithm:)

1. $V_m = [v_1, v_2, ..., v_m]$ **orthonormal basis of** K_m.

2. $AV_m = V_{m+1}\overline{H}_m$

3. $V_m^T A V_m = H_m \equiv \overline{H}_m -$ **last row.**

Arnoldi's Method ($L_m = K_m$)

From Petrov-Galerkin condition when $L_m = K_m$, we get

$$x_m = x_0 + V_m H_m^{-1} V_m^T r_0$$

If, in addition we choose $v_1 = r_0/\|r_0\|_2 \equiv r_0/\beta$ in Arnoldi's algorithm, then

$$x_m = x_0 + \beta V_m H_m^{-1} e_1$$

Several algorithms mathematically equivalent to this approach:

* FOM [Saad, 1981] (above formulation)

* Young and Jea's ORTHORES [1982].

* Axelsson's projection method [1981].

Minimal residual methods ($L_m = AK_m$)

When $L_m = AK_m$, we let $W_m \equiv AV_m$ and obtain relation

$$x_m = x_0 + V_m[W_m^T A V_m]^{-1} W_m^T r_0 = x_0 + V_m[(AV_m)^T AV_m]^{-1}(AV_m)^T r_0.$$

Use again $v_1 := r_0/(\beta := \|r_0\|_2)$ and the relation $\boxed{AV_m = V_{m+1}\bar{H}_m}$:

$$x_m = x_0 + V_m[\bar{H}_m^T \bar{H}_m]^{-1} \bar{H}_m^T \beta e_1 = x_0 + V_m y_m$$

where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ over $y \in \mathbb{R}^m$. Therefore, (Generalized Minimal Residual method (GMRES) [Saad-Schultz, 1983]):

$$x_m = x_0 + V_m y_m \quad \textbf{where} \quad y_m : \min_y \|\beta e_1 - \bar{H}_m y\|_2$$

Equivalent methods:

- **Axelsson's CGLS**
- **Orthomin (1980)**
- **Orthodir**
- **GCR**

Restarting and Truncating

Difficulty: As m increases, storage and work per step increase fast.

First remedy: Restarting. Fix the dimension m of the subspace

ALGORITHM : 3. ***Restarted GMRES (resp. Arnoldi)***

1. **Start/Restart:** ***Compute*** $r_0 = b - Ax_0$, ***and*** $v_1 = r_0/(\beta := \|r_0\|_2)$.
2. **Arnoldi Process:** ***generate*** $\bar{H}_m$ ***and*** V_m.
3. ***Compute*** $y_m = H_m^{-1}\beta e_1$ ***(FOM), or***

   $y_m = argmin\|\beta e_1 - \bar{H}_m y\|_2$ ***(GMRES)***
4. $x_m = x_0 + V_m y_m$
5. ***If*** $\|r_m\|_2 \leq \epsilon\|r_0\|_2$ ***stop else set*** $x_0 := x_m$ ***and go to 1.***

Second remedy: Truncate the orthogonalization

The formula for v_{j+1} is replaced by

$$h_{j+1,j} v_{j+1} = A v_j - \sum_{i=j-k+1}^{j} h_{ij} v_i$$

$\longrightarrow$ each v_j is made orthogonal to the previous k v_i's.

$\longrightarrow$ x_m still computed as $x_m = x_0 + V_m H_m^{-1} \beta e_1$.

$\longrightarrow$ It can be shown that this is again an oblique projection process.

IOM (Incomplete Orthogonalization Method) = replace orthogonalization in FOM, by the above truncated (or 'incomplete') orthogonalization.

The direct version of IOM [DIOM]:

Writing the LU decomposition of H_m as $H_m = L_m U_m$ we get

$$x_m = x_0 + V_m U_m^{-1} \; L_m^{-1} \beta e_1 \equiv x_0 + P_m z_m$$

➨ **Structure of L_m, U_m when $k = 3$**

$$L_m = \begin{pmatrix} 1 & & & & & & \\ x & 1 & & & & & \\ & x & 1 & & & & \\ & & x & 1 & & & \\ & & & x & 1 & & \\ & & & & x & 1 & \\ & & & & & x & 1 \end{pmatrix} \qquad U_m = \begin{pmatrix} x & x & x & & & & \\ & x & x & x & & & \\ & & x & x & x & & \\ & & & x & x & x & \\ & & & & x & x & x \\ & & & & & x & x \\ & & & & & & x \end{pmatrix}$$

$$p_m = u_{mm}^{-1}[v_m - \Sigma_{i=m-k+1}^{m-1} u_{im} p_i] \qquad z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix}$$

- Can update x_m at each step:

$$x_m = x_{m-1} + \zeta_m p_m$$

Note: Several existing pairs of methods have a similar link: they are based on the LU, or other, factorizations of the H_m matrix

- CG-like formulation of IOM called DIOM [Saad, 1982]
- ORTHORES(k) [Young & Jea '82] equivalent to DIOM(k)
- SYMMLQ [Paige and Saunders, '77] uses LQ factorization of H_m.
- Can incorporate partial pivoting in LU factorization of H_m

Some implementation details: GMRES

- **Issue 1 : how to solve least-squares problem ?**
- **Issue 2: How to compute residual norm (without computing solution at each step)?**
- **Several solutions to both issues. Simplest: use Givens rotations.**
- **Recall: we want to solve least-squares problem**

$$\min_y \|\beta e_1 - \overline{H}_m y\|_2$$

- **Transform the problem into upper triangular one.**

➤ **Rotation matrices of dimension $m+1$. Define (with $s_i^2 + c_i^2 = 1$):**

$$\Omega_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \\ & & & -s_i & c_i & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{row } i \\ \leftarrow \text{row } i+1 \\ \\ \\ \\ \end{matrix}$$

➤ **Multiply $\bar{H}_m$ and right-hand side $\bar{g}_0 \equiv \beta e_1$ by a sequence of such matrices from the left.** ➤ **s_i, c_i selected to eliminate $h_{i+1,i}$**

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- **1-st Rotation**

$$\Omega_1 = \begin{pmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$$

with

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}, \quad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}}$$

$$\bar{H}_m^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

➨ **repeat with $\Omega_2, \ldots, \Omega_i$. Result:**

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix}.$$

Define

$$\begin{aligned} Q_m &= \Omega_m \Omega_{m-1} \dots \Omega_1 \\ \bar{R}_m &= \bar{H}_m^{(m)} = Q_m \bar{H}_m, \\ \bar{g}_m &= Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T. \end{aligned}$$

- **Since Q_m is unitary,**

$$\min \|\beta e_1 - \bar{H}_m y\|_2 = \min \|\bar{g}_m - \bar{R}_m y\|_2.$$

- **Delete last row and solve resulting triangular system.**

$$R_m y_m = g_m$$

PROPOSITION:

1. The rank of AV_m is equal to the rank of R_m. In particular, if $r_{mm} = 0$ then A must be singular.

2. The vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ is given by

$$y_m = R_m^{-1} g_m.$$

3. The residual vector at step m satisfies

$$b - Ax_m = V_{m+1}\left(\beta e_1 - \bar{H}_m y_m\right) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$$

and, as a result,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}|.$$

THE SYMMETRIC CASE: Observation

Observe: When A is real symmetric then in Arnoldi's method:

$$\boxed{H_m = V_m^T A V_m}$$

must be symmetric. Therefore

Theorem. When Arnoldi's algorithm is applied to a (real) symmetric matrix then the matrix H_m is symmetric tridiagonal:

$h_{ij} = 0 \quad 1 \leq i < j-1;$ and $h_{j,j+1} = h_{j+1,j}, \quad j = 1, \ldots, m$

- **We can write**

$$H_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & . & . & . & \\ & & & . & . & . \\ & & & & \beta_m & \alpha_m \end{pmatrix} \tag{1}$$

The v_i's satisfy a three-term recurrence [Lanczos Algorithm]:

$$\boxed{\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}}$$

$\longrightarrow$ simplified version of Arnoldi's algorithm for sym. systems.

Symmetric matrix + Arnoldi $\longrightarrow$ Symmetric Lanczos

The Lanczos algorithm

ALGORITHM : 4. **Lanczos**

1. **Choose an initial vector v_1 of norm unity. Set $\beta_1 \equiv 0, v_0 \equiv 0$**
2. **For $j = 1, 2, \ldots, m$ Do:**
3. $w_j := Av_j - \beta_j v_{j-1}$
4. $\alpha_j := (w_j, v_j)$
5. $w_j := w_j - \alpha_j v_j$
6. $\beta_{j+1} := \|w_j\|_2$. **If $\beta_{j+1} = 0$ then Stop**
7. $v_{j+1} := w_j / \beta_{j+1}$
8. **EndDo**

Lanczos algorithm for linear systems

- **Usual orthogonal projection method setting:**

  - $L_m = K_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$
  - **Basis** $V_m = [v_1, \ldots, v_m]$ **of** K_m **generated by the Lanczos algorithm**

- **Three different possible implementations.**

(1) Arnoldi-like; (2) Exploit tridigonal nature of H_m (DIOM); (3) Conjugate gradient.

ALGORITHM : 5. *Lanczos Method for Linear Systems*

1. **Compute** $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, **and** $v_1 := r_0/\beta$
2. **For** $j = 1, 2, \ldots, m$ **Do:**
3. $w_j = Av_j - \beta_j v_{j-1}$ **(If** $j = 1$ **set** $\beta_1 v_0 \equiv 0$**)**
4. $\alpha_j = (w_j, v_j)$
5. $w_j := w_j - \alpha_j v_j$
6. $\beta_{j+1} = \|w_j\|_2$**. If** $\beta_{j+1} = 0$ **set** $m := j$ **and go to 9**
7. $v_{j+1} = w_j/\beta_{j+1}$
8. **EndDo**
9. **Set** $T_m = tridiag(\beta_i, \alpha_i, \beta_{i+1})$, **and** $V_m = [v_1, \ldots, v_m]$.
10. **Compute** $y_m = T_m^{-1}(\beta e_1)$ **and** $x_m = x_0 + V_m y_m$

ALGORITHM : 6. *D-Lanczos*

1. **Compute** $r_0 = b - Ax_0$, $\zeta_1 := \beta := \|r_0\|_2$, **and** $v_1 := r_0/\beta$
2. **Set** $\lambda_1 = \beta_1 = 0$, $p_0 = 0$
3. **For** $m = 1, 2, \ldots$, **until convergence Do:**
4. **Compute** $w := Av_m - \beta_m v_{m-1}$ **and** $\alpha_m = (w, v_m)$
5. **If** $m > 1$ **then compute** $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ **and** $\zeta_m = -\lambda_m \zeta_{m-1}$
6. $\eta_m = \alpha_m - \lambda_m \beta_m$
7. $p_m = \eta_m^{-1} (v_m - \beta_m p_{m-1})$
8. $x_m = x_{m-1} + \zeta_m p_m$
9. **If** x_m **has converged then Stop**
10. $w := w - \alpha_m v_m$
11. $\beta_{m+1} = \|w\|_2$, $v_{m+1} = w/\beta_{m+1}$
12. **EndDo**

The Conjugate Gradient Algorithm (A S.P.D.)

- Note: the p_i's are A-orthogonal
- The r_i''s are orthogonal.
- And we have $x_m = x_{m-1} + \xi_m p_m$

So there must be an update of the form:

1. $p_m = r_{m-1} + \beta_m p_{m-1}$
2. $x_m = x_{m-1} + \xi_m p_m$
3. $r_m = r_{m-1} - \xi_m A p_m$

The Conjugate Gradient Algorithm (A S.P.D.)

1. Start: $r_0 := b - Ax_0$, $p_0 := r_0$.

2. Iterate: **Until convergence do,**

(a) $\alpha_j := (r_j, r_j)/(Ap_j, p_j)$

(b) $x_{j+1} := x_j + \alpha_j p_j$

(c) $r_{j+1} := r_j - \alpha_j A p_j$

(d) $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$

(e) $p_{j+1} := r_{j+1} + \beta_j p_j$

- $r_j = scaling \times v_{j+1}$. The r_j's are orthogonal.
- The p_j's are A-conjugate, i.e., $(Ap_i, p_j) = 0$ for $i \neq j$.

METHODS BASED ON LANCZOS BIORTHOGONALIZATION

ALGORITHM : 7. *The Lanczos Bi-Orthogonalization Procedure*

1. *Choose two vectors* v_1, w_1 *such that* $(v_1, w_1) = 1$.
2. *Set* $\beta_1 = \delta_1 \equiv 0$, $w_0 = v_0 \equiv 0$
3. *For* $j = 1, 2, \ldots, m$ *Do:*
4. $\alpha_j = (Av_j, w_j)$
5. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}$. *If* $\delta_{j+1} = 0$ *Stop*
8. $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$
9. $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
10. $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
11. *EndDo*

- **Extension of the symmetric Lanczos algorithm**

- **Builds a pair of biorthogonal bases for the two subspaces**

$$\mathcal{K}_m(A, v_1) \quad \text{and} \quad \mathcal{K}_m(A^T, w_1)$$

- **Different ways to choose $\delta_{j+1}, \beta_{j+1}$ in lines 7 and 8.**

Let

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & . & . & . & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix}.$$

- $v_i \in \mathcal{K}_m(A, v_1)$ **and** $w_j \in \mathcal{K}_m(A^T, w_1)$**.**

If the algorithm does not break down before step m, then the vectors $v_i, i = 1, \ldots, m$, and $w_j, j = 1, \ldots, m$, are biorthogonal, i.e.,

$$(v_j, w_i) = \delta_{ij} \quad 1 \leq i,\ j \leq m\ .$$

Moreover, $\{v_i\}_{i=1,2,\ldots,m}$ is a basis of $\mathcal{K}_m(A, v_1)$ and $\{w_i\}_{i=1,2,\ldots,m}$ is a basis of $\mathcal{K}_m(A^T, w_1)$ and

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T,$$
$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T,$$
$$W_m^T A V_m = T_m\ .$$

The Lanczos Algorithm for Linear Systems

ALGORITHM : 8. **Lanczos Algorithm for Linear Systems**

1. **Compute** $r_0 = b - Ax_0$ **and** $\beta := \|r_0\|_2$
2. **Run** m **steps of the nonsymmetric Lanczos Algorithm i.e.,**
3. **Start with** $v_1 := r_0/\beta$**, and any** w_1 **such that** $(v_1, w_1) = 1$
4. **Generate the Lanczos vectors** $v_1, \ldots, v_m$**,** $w_1, \ldots, w_m$
5. **and the tridiagonal matrix** T_m **from Algorithm ??.**
6. **Compute** $y_m = T_m^{-1}(\beta e_1)$ **and** $x_m := x_0 + V_m y_m$**.**

➨ **BCG can be derived from the Lanczos Algorithm similarly to CG in symmetric case.**

The BCG and QMR Algorithms

- Let $\boxed{T_m = L_m U_m}$ (LU factorization of T_m). Define $\boxed{P_m = V_m U_m^{-1}}$

Then, solution is

$$x_m = x_0 + V_m T_m^{-1}(\beta e_1) = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) = x_0 + P_m L_m^{-1}(\beta e_1)$$

- x_m is updatable from x_{m-1} similar to the CG algorithm.
- r_j and r_j^* are in the same direction as v_{j+1} and w_{j+1} respectively.
- they form a biorthogonal sequence.
- The p_i^*'s p_i's are are A-conjugate.
- Utilizing this information, a CG-like algorithm can be easily derived from the Lanczos procedure.

ALGORITHM : 9. *BiConjugate Gradient (BCG)*

1. *Compute* $r_0 := b - Ax_0$. *Choose* r_0^* *such that* $(r_0, r_0^*) \neq 0$.
2. *Set,* $p_0 := r_0$, $p_0^* := r_0^*$
3. *For* $j = 0, 1, \ldots$, *until convergence Do:,*
4. $\alpha_j := (r_j, r_j^*)/(Ap_j, p_j^*)$
5. $x_{j+1} := x_j + \alpha_j p_j$
6. $r_{j+1} := r_j - \alpha_j A p_j$
7. $r_{j+1}^* := r_j^* - \alpha_j A^T p_j^*$
8. $\beta_j := (r_{j+1}, r_{j+1}^*)/(r_j, r_j^*)$
9. $p_{j+1} := r_{j+1} + \beta_j p_j$
10. $p_{j+1}^* := r_{j+1}^* + \beta_j p_j^*$
11. *EndDo*

Quasi-Minimal Residual Algorithm

- The Lanczos algorithm gives the relations $AV_m = V_{m+1}\bar{T}_m$ with $\bar{T}_m$ = $(m+1) \times m$ tridiagonal matrix $\bar{T}_m = \begin{pmatrix} T_m \\ \delta_{m+1} e_m^T \end{pmatrix}$.
- Let $v_1 \equiv \beta r_0$ and $x = x_0 + V_m y$. Residual norm $\|b - Ax\|_2$ is

$$\|r_0 - AV_m y\|_2 = \|\beta v_1 - V_{m+1}\bar{T}_m y\|_2 = \|V_{m+1}\left(\beta e_1 - \bar{T}_m y\right)\|_2$$

- Column-vectors of V_{m+1} are not orthonormal ($\neq$ GMRES).
- But: reasonable idea to minimize the function $J(y) \equiv \|\beta e_1 - \bar{T}_m y\|_2$
- Quasi-Minimal Residual Algorithm (Freund, 1990).

ALGORITHM : 10. QMR

1. Compute $r_0 = b - Ax_0$ and $\gamma_0 := \|r_0\|_2$, $w_1 := v_1 := r_0/\gamma_1$
2. For $m = 1, 2, \ldots,$ until convergence Do:
3. Compute α_m, δ_{m+1} and v_{m+1}, w_{m+1} as in Lanczos Algor. [alg. ??]
4. Update the QR factorization of $\bar{T}_m$, i.e.,
5. Apply Ω_i, $i = m-2, m-1$ to the m-th column of $\bar{T}_m$
6. Compute the rotation coefficients c_m, s_m
7. Apply rotation Ω_m, to $\bar{T}_m$ and $\bar{g}_m$, i.e., compute:
8. $\gamma_{m+1} := -s_m\gamma_m$; $\gamma_m := c_m\gamma_m$; and $\alpha_m := c_m\alpha_m + s_m\delta_{m+1}$
9. $p_m = \left(v_m - \sum_{i=m-2}^{m-1} t_{im}p_i\right)/t_{mm}$
10. $x_m = x_{m-1} + \gamma_m p_m$
11. If $|\gamma_{m+1}|$ is small enough Stop
12. EndDo

Transpose-Free Variants

- BCG and QMR require a matrix-by-vector product with A and A^T at each step. The products with A^T do not contribute directly to x_m.
- They allow to determine the scalars (α_j and β_j in BCG).
- QUESTION: is it possible to bypass the use of A^T?

- Motivation: in nonlinear equations, A is often not available explicitly but via the Frechet derivative:

$$J(u_k)v = \frac{F(u_k + \epsilon v) - F(u_k)}{\epsilon} .$$

Conjugate Gradient Squared

*** Clever variant of BCG which avoids using A^T [Sonneveld, 1984].**

In BCG:

$$r_i = \rho_i(A) r_0$$

where $\rho_i =$ polynomial of degree i.

In CGS:

$$r_i = \rho_i^2(A) r_0$$

- **Define** $r_j = \phi_j(A)r_0, \quad p_j = \pi_j(A)r_0, \quad r_j^* = \phi_j(A^T)r_0^*, \quad p_j^* = \pi_j(A^T)r_0^*.$

Scalar α_j in BCG is given by

$$\alpha_j = \frac{(\phi_j(A)r_0, \phi_j(A^T)r_0^*)}{(A\pi_j(A)r_0, \pi_j(A^T)r_0^*)} = \frac{(\phi_j^2(A)r_0, r_0^*)}{(A\pi_j^2(A)r_0, r_0^*)}$$

- **Possible to get a recursion for the $\phi_j^2(A)r_0$ and $\pi_j^2(A)r_0$?**

$$\phi_{j+1}(t) = \phi_j(t) - \alpha_j t\pi_j(t),$$
$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j\pi_j(t)$$

Square the equalities

$$\phi_{j+1}^2(t) = \phi_j^2(t) - 2\alpha_j t\pi_j(t)\phi_j(t) + \alpha_j^2 t^2\pi_j^2(t),$$
$$\pi_{j+1}^2(t) = \phi_{j+1}^2(t) + 2\beta_j\phi_{j+1}(t)\pi_j(t) + \beta_j^2\pi_j(t)^2.$$

- **Problem: Cross terms**

- **Solution: Let $\phi_{j+1}(t)\pi_j(t)$, be a third member of the recurrence.**

For $\pi_j(t)\phi_j(t)$, note:

$$\phi_j(t)\pi_j(t) = \phi_j(t)\left(\phi_j(t) + \beta_{j-1}\pi_{j-1}(t)\right) = \phi_j^2(t) + \beta_{j-1}\phi_j(t)\pi_{j-1}(t).$$

- **Result:**

$$\begin{aligned}
\phi_{j+1}^2 &= \phi_j^2 - \alpha_j t\left(2\phi_j^2 + 2\beta_{j-1}\phi_j\pi_{j-1} - \alpha_j t\,\pi_j^2\right) \\
\phi_{j+1}\pi_j &= \phi_j^2 + \beta_{j-1}\phi_j\pi_{j-1} - \alpha_j t\,\pi_j^2 \\
\pi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j\phi_{j+1}\pi_j + \beta_j^2\pi_j^2.
\end{aligned}$$

- **Define:**

$$r_j = \phi_j^2(A)r_0, \quad p_j = \pi_j^2(A)r_0, \quad q_j = \phi_{j+1}(A)\pi_j(A)r_0$$

Recurrences become:

$$
\begin{aligned}
r_{j+1} &= r_j - \alpha_j A\left(2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A\, p_j\right), \\
q_j &= r_j + \beta_{j-1}q_{j-1} - \alpha_j A\, p_j, \\
p_{j+1} &= r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j.
\end{aligned}
$$

Define auxiliary vector $d_j = 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A p_j$

➨ **Sequence of operations to compute the approximate solution, starting with** $r_0 := b - Ax_0$, $p_0 := r_0$, $q_0 := 0$, $\beta_0 := 0$.

1. $\alpha_j = (r_j, r_0^*)/(Ap_j, r_0^*)$	**5.** $r_{j+1} = r_j - \alpha_j A d_j$
2. $d_j = 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A p_j$	**6.** $\beta_j = (r_{j+1}, r_0^*)/(r_j, r_0^*)$
3. $q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j A p_j$	**7.** $p_{j+1} = r_{j+1} + \beta_j(2q_j + \beta_j p_j)$.
4. $x_{j+1} = x_j + \alpha_j d_j$	

- one more auxiliary vector, $u_j = r_j + \beta_{j-1} q_{j-1}$. So

$$\begin{aligned} d_j &= u_j + q_j, \\ q_j &= u_j - \alpha_j A p_j, \\ p_{j+1} &= u_{j+1} + \beta_j (q_j + \beta_j p_j), \end{aligned}$$

- vector d_j is no longer needed.

ALGORITHM : 11. *Conjugate Gradient Squared*

1. **Compute** $r_0 := b - Ax_0$; r_0^* **arbitrary.**
2. **Set** $p_0 := u_0 := r_0$.
3. **For** $j = 0, 1, 2 \ldots,$ **until convergence Do:**
4. $\alpha_j = (r_j, r_0^*)/(Ap_j, r_0^*)$
5. $q_j = u_j - \alpha_j A p_j$
6. $x_{j+1} = x_j + \alpha_j(u_j + q_j)$
7. $r_{j+1} = r_j - \alpha_j A(u_j + q_j)$
8. $\beta_j = (r_{j+1}, r_0^*)/(r_j, r_0^*)$
9. $u_{j+1} = r_{j+1} + \beta_j q_j$
10. $p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j)$
11. **EndDo**

➨ **Note: no matrix-by-vector products with A^T but two matrix-by-vector products with A, at each step.**

Vector: $\longleftrightarrow$ **Polynomial in BCG :**

$$q_i \longleftrightarrow \bar{r}_i(t)\bar{p}_{i-1}(t)$$

$$u_i \longleftrightarrow \bar{p}_i^2(t)$$

$$r_i \longleftrightarrow \bar{r}_i^2(t)$$

where $\bar{r}_i(t)$ = residual polynomial at step i for BCG, .i.e., $r_i = \bar{r}_i(A)r_0$, and $\bar{p}_i(t)$ = conjugate direction polynomial at step i, i.e., $p_i = \bar{p}_i(A)r_0$.

BCGSTAB (van der Vorst, 1992)

- In CGS: residual polynomial of BCG is squared. ➨ bad behavior in case of irregular convergence.

- Bi-Conjugate Gradient Stabilized (BCGSTAB) = a variation of CGS which avoids this difficulty. ➨ Derivation similar to CGS.

- Residuals in BCGSTAB are of the form, $\boxed{r'_j = \psi_j(A)\phi_j(A)r_0}$ in which, $\phi_j(t)$ = BCG residual polynomial, and ..

- .. $\psi_j(t)$ = a new polynomial defined recursively as

$$\psi_{j+1}(t) = (1 - \omega_j t)\psi_j(t)$$

ω_i chosen to 'smooth' convergence [steepest descent step]

ALGORITHM : 12 . BCGSTAB

1. **Compute** $r_0 := b - Ax_0$; r_0^* **arbitrary;**
2. $p_0 := r_0$.
3. **For** $j = 0, 1, \ldots,$ **until convergence Do:**
4. $\alpha_j := (r_j, r_0^*)/(Ap_j, r_0^*)$
5. $s_j := r_j - \alpha_j A p_j$
6. $\omega_j := (As_j, s_j)/(As_j, As_j)$
7. $x_{j+1} := x_j + \alpha_j p_j + \omega_j s_j$
8. $r_{j+1} := s_j - \omega_j A s_j$
9. $\beta_j := \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}$
10. $p_{j+1} := r_{j+1} + \beta_j (p_j - \omega_j A p_j)$
11. **EndDo**

THEORY

Convergence Theory for CG

- Approximation of the form $x = x_0 + p_{m-1}(A)r_0$. with $x_0 =$ initial guess, $r_0 = b - Ax_0$;

- Optimality property:

x_m minimizes $\|x - x_*\|_A$ over $x_0 + K_m$

- **Consequence:** Standard result

Let x_m = m-th CG iterate, x_* = exact solution and

$$\eta = \frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}$$

Then: $$\|x_* - x_m\|_A \leq \frac{\|x_* - x_0\|_A}{T_m(1 + 2\eta)}$$

where T_m = Chebyshev polynomial of degree m.

THEORY FOR NONHERMITIAN CASE

- **Much more difficult!**
- **No convincing results on** *'global convergence'* **for many algorithms (bi-CG, FOM, etc..)**
- **Can get a general a-priori – a-posteriori error bound**

Convergence results for nonsymmetric case

- **Methods based on minimum residual better understood.**

- **If $(A + A^T)$ is positive definite ($(Ax, x) > 0 \; \forall x \neq 0$), all minimum residual-type methods (ORTHOMIN, ORTHODIR, GCR, GMRES,...), + their restarted and truncated versions, converge.**

- **Convergence results based on comparison with steepest descent [Eisenstat, Elman, Schultz 1982] $\longrightarrow$ not sharp.**

Minimum residual methods: if $A = X\Lambda X^{-1}$, Λ diagonal, then

$$\|b - Ax_m\|_2 \leq \mathbf{Cond}_2(X) \min_{p \in \mathcal{P}_{m-1}, p(0)=1} \quad max_{\lambda \in \Lambda(A)} |p(\lambda)|$$

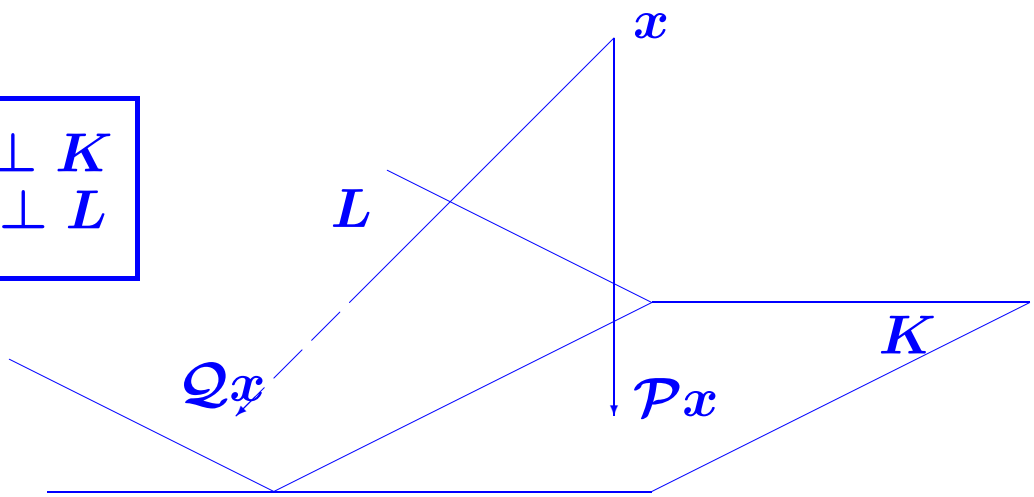
($\mathcal{P}_{m-1} \equiv$ set of polynomials of degree $\leq m-1$, $\Lambda(A) \equiv$ spectrum of A)

Two useful projectors

Let $\mathcal{P}$ be the orthogonal projector onto K and

$\mathcal{Q}$ be the (oblique) projector onto K and orthogonally to L.

$$\begin{array}{l} \mathcal{P}x \in K, \quad x - \mathcal{P}x \perp K \\ \mathcal{Q}x \in K, \quad x - \mathcal{Q}x \perp L \end{array}$$



The approximate problem in terms of $\mathcal{P}$ and $\mathcal{Q}$

- **Approximate problem amounts to solving**

$$\mathcal{Q}(b - Ax) = 0, \quad x \quad \in K$$

or in operator form

$$\mathcal{Q}(b - A\mathcal{P}x) = 0$$

Question: what accuracy can one expect?

- **If x^* is the exact solution, then we cannot get better accuracy than $\|(I - \mathcal{P})x^*\|_2$, i.e.,**

$$\boxed{\|\tilde{x} - x^*\|_2 \geq \|(I - \mathcal{P})x^*\|_2}$$

THEOREM. Let $\gamma = \|\mathcal{Q}A(I-\mathcal{P})\|_2$ and assume that b belongs to K. Then the residual norm of the exact solution x^* for the (approximate) linear operator A_m satisfies the inequality,

$$\boxed{\|b - A_m x^*\|_2 \leq \gamma \|(I-\mathcal{P})x^*\|_2}$$

➨ In other words "if approximate problem is not poorly conditioned and if $\|(I-\mathcal{P})x^*\|_2$ is small then we will obtain a good approximate solution".

Methods based on the normal equations

It is possible to obtain the solution of $Ax = b$ from the equivalent system:

$$\boxed{A^T A x = A^T b}$$

or

$$\boxed{A A^T y = b \, , x = A^T y}$$

➨ Methods based on these approaches are usually slower than previous ones. (Condition number of system is squared)

➨ Exception: when A is strongly indefinite (extreme case: A is orthogonal, $A^T A = I \rightarrow$ convergence in 1 step).

CGNR and CGNE

Can use CG to solve normal equations. Two well-known options.

(1) CGNR: Conjugate Gradient method on

$$\boxed{A^T A x = A^T b}$$

(2) CGNE: Let $x = A^T y$ and use conjugate gradient method on

$$\boxed{A A^T y = b}$$

- Different optimality properties
- Various 'efficient' formulations in both cases

ALGORITHM : 13. CGNR

1. **Compute** $r_0 = b - Ax_0$, $z_0 = A^T r_0$, $p_0 = z_0$.
2. **For** $i = 0, \ldots,$ **until convergence Do:**
3. $w_i = Ap_i$
4. $\alpha_i = \|z_i\|^2 / \|w_i\|_2^2$
5. $x_{i+1} = x_i + \alpha_i p_i$
6. $r_{i+1} = r_i - \alpha_i w_i$
7. $z_{i+1} = A^T r_{i+1}$
8. $\beta_i = \|z_{i+1}\|_2^2 / \|z_i\|_2^2$,
9. $p_{i+1} = z_{i+1} + \beta_i p_i$
10. **EndDo**

CGNR: The approximation x_m minimizes the residual norm $\|b - Ax\|_2$ over the affine Krylov subspace,

$$x_0 + \text{span}\{A^T r_0, (A^T A)A^T r_0, \ldots, (A^T A)^{m-1} A^T r_0\},$$

where $r_0 \equiv b - Ax_0$.

- **The difference with GMRES is the subspace in which the residual norm is minimized. For GMRES the subspace is $x_0 + \mathcal{K}_m(A, r_0)$.**

ALGORITHM : 14. *CGNE (Craig's Method)*

1. **Compute** $r_0 = b - Ax_0$, $p_0 = A^T r_0$.
2. **For** $i = 0, 1, \ldots,$ **until convergence Do:**
3. $\alpha_i = (r_i, r_i)/(p_i, p_i)$
4. $x_{i+1} = x_i + \alpha_i p_i$
5. $r_{i+1} = r_i - \alpha_i A p_i$
6. $\beta_i = (r_{i+1}, r_{i+1})/(r_i, r_i)$
7. $p_{i+1} = A^T r_{i+1} + \beta_i p_i$
8. **EndDo**

CGNE produces the approximate solution x in the subspace

$$\boxed{x_0 + A^T\mathcal{K}_m(AA^T, r_0) = x_0 + \mathcal{K}_m(A^TA, A^Tr_0)}$$

which minimizes $x_* - x$, where $x_* = A^{-1}b$, $r_0 = b - Ax_0$.

- **Note: Same subspace as CGNR!**

Block GMRES and Block Krylov Methods

Main Motivation: To solve linear systems with several right-hand sides

$$Ax^{(i)} = b^{(i)}, \quad i = 1, \ldots, p$$

or, in matrix form,

$$AX = B$$

- Sometimes Block methods are used as a strategy for enhancing convergence even for the case $p = 1$.

Let

$$R_0 \equiv [r_0^{(1)}, r_0^{(2)}, \ldots, r_0^{(p)}] \; .$$

each column is $r_0^{(i)} = b^{(i)} - Ax_0^{(i)}$.

Krylov methods find an approximation to X from the subspace

$$K_m(A, R_0) = \text{span}\{R_0, AR_0, \ldots, A^{m-1}R_0\}$$

- For example Block-GMRES (BGMRES) £nds X to

$$\text{minimize } \|B - AX\|_F \quad \text{for} \quad X \in X_0 + K_m(A, R_0)$$

- Various implementations of BGMRES exist

- Simplest one is based on Ruhe's variant of the Block Arnoldi procedure.

ALGORITHM : 15. **Block Arnoldi–Ruhe's variant**

1. **Choose p initial orthonormal vectors $\{v_i\}_{i=1,\ldots,p}$.**
2. **For $j = p, p+1, \ldots, m$ Do:**
3. **Set $k := j - p + 1$;**
4. **Compute $w := Av_k$;**
5. **For $i = 1, 2, \ldots, j$ Do:**
6. $h_{i,k} := (w, v_i)$
7. $w := w - h_{i,k} v_i$
8. **EndDo**
9. **Compute $h_{j+1,k} := \|w\|_2$ and $v_{j+1} := w/h_{j+1,k}$.**
10. **EndDo**

- $p = 1$ coincides with standard Arnoldi process.

- Interesting feature: dimension of the subspace need not be a multiple of the block-size p.

At the end of the algorithm, we have the relation

$$AV_m = V_{m+p}\bar{H}_m.$$

- The matrix $\bar{H}_m$ is now of size $(m + p) \times m$.

- Each approximate solution has the form

$$x^{(i)} = x_0^{(i)} + V_m y^{(i)},$$

where $y^{(i)}$ must minimize the norm $\|b^{(i)} - Ax^{(i)}\|_2$.

- Plane rotations can be used for this purpose as in the standard GMRES [p rotations are needed for each step.]