

## Methods based on Lanczos biorthogonalization

---

- Biorthogonal systems and Lanczos biorthogonalization.
- The Bi-CG algorithm, the QMR algorithm
- Transpose-free variant - CGS and Bi-CGSTAB
- See Chapter 7 of text for details.

## Bi-Orthogonal systems and Bi-Orthogonal bases

▶▶ Two sets of vectors

$\{v_1, v_2, \dots, v_m\}$  and  $\{w_1, w_2, \dots, w_m\}$

are said to be **bi-orthogonal** when

$$(w_i, v_j) = 0 \text{ for } i \neq j$$

▶▶ The system is **bi-orthonormal** if  $(w_i, v_j) = \delta_{ij}$

▶▶ If  $V = [v_1, v_2, \dots, v_m]$ ,  $W = [w_1, w_2, \dots, w_m]$  then this is equivalent to  $W^T V = I$

Given a basis  $X$  of  $K$  and a basis  $Y$  of  $L$ , one can devise a Gram-Schmidt-like algorithm to build bases  $V$  of  $K$  and  $W$  of  $L$  that are bi-orthonormal

▶▶ The Lanczos bi-orthogonalization algorithm does this

# The Lanczos Bi-Orthogonalization Procedure

## ALGORITHM : 1. Lanczos bi-orthogonalization

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  $j = 1, 2, \dots, m$  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

## Properties

- ▶ 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

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

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

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

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

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

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

# The Lanczos Algorithm for Linear Systems

## ALGORITHM : 2. Bi-Lanczos for Linear Systems

1. Compute  $r_0 = b - Ax_0$  and  $\beta := \|r_0\|_2$
2. Run  $m$  steps of the nonsymmetric Lanczos Algorithm,
3. starting with  $v_1 := r_0/\beta$ , and  $w_1$  s. t.  $(v_1, w_1) = 1$
4. Result: Lanczos vectors  $v_1, \dots, v_m, w_1, \dots, w_m$
5. and tridiagonal matrix  $T_m$
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  $T_m = L_m U_m$  (LU factorization of  $T_m$ ).

Define  $P_m = V_m U_m^{-1}$  Then, solution is

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

▶  $x_m$  is updatable from  $x_{m-1}$ , similarly to CG

▶  $r_j$  and  $r_j^*$  are in the same direction as  $v_{j+1}$  and  $w_{j+1}$  respectively  $\rightarrow$  they form a biorthogonal sequence.

▶ The  $p_i^*$ 's and  $p_i$ 's are A-conjugate.

▶ Utilizing this information, a CG-like algorithm can be easily derived from the Lanczos procedure.

## ALGORITHM : 3. 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  $j = 0, 1, \dots$ , 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.      $r_{j+1} := r_j - \alpha_j Ap_j$
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

$$AV_m = V_{m+1}\bar{T}_m \quad \text{with} \quad \bar{T}_m = \begin{pmatrix} T_m \\ \delta_{m+1}e_m^T \end{pmatrix}$$

- ▶  $\bar{T}_m = (m + 1) \times m$  tridiagonal matrix

- ▶ Let  $v_1 \equiv \beta r_0$ ,  $x \equiv x_0 + V_m y$ . Residual norm:

$$\begin{aligned} \|b - Ax\|_2 &= \|r_0 - AV_m y\|_2 = \|\beta v_1 - V_{m+1}\bar{T}_m y\|_2 \\ &= \|V_{m+1}(\beta e_1 - \bar{T}_m y)\|_2 \end{aligned}$$

- ▶ Columns of  $V_{m+1}$  are not orthonormal ( $\neq$  GMRES)

- ▶ But still a good idea to minimize the function

$$J(y) \equiv \|\beta e_1 - \bar{T}_m y\|_2$$

- ▶ Quasi-Minimal Residual Algorithm (Freund, 1990).

## ALGORITHM : 4. QMR

---

1. Compute  $r_0 = b - Ax_0$ ,  $\gamma_0 := \|r_0\|_2$ ,  $w_1 := v_1 := r_0/\gamma_1$
2. For  $m = 1, 2, \dots$ , until convergence Do:
3.     Compute  $\alpha_m, \delta_{m+1}$  and  $v_{m+1}, w_{m+1}$  as in bi-Lanczos
4.     Update the QR factorization of  $\bar{T}_m$ , i.e.,
5.         Apply  $\Omega_{m-2}, \Omega_{m-1}$ , to  $T_{:,m}$
6.         Compute the rotation coefficients  $c_m, s_m$
7.     Apply rotation  $\Omega_m$ , 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 ( $\alpha_j$  and  $\beta_j$  in BCG).

▶▶ QUESTION: is it possible to bypass the use of  $A^T$ ?

▶▶ Motivation: In many applications,  $A$  is available only through matrix-vector products. In this case  $A^T$  is often not available.

▶▶ Example: In solving 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 (CGS)

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

▶▶ Let  $\rho_i$  be the  $i$ -th residual polynomial for BCG. By definition:

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

where  $\rho_i =$  polynomial of degree  $i$ .

▶▶ Then, in CGS:

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

▶ Define

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

Scalar  $\alpha_j$  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  $\pi_j^2(A)r_0$ ?

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

$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j \pi_j(t)$$

▶ Let us square these 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:

$$\begin{aligned} \phi_j(t)\pi_j(t) &= \phi_j(t) (\phi_j(t) + \beta_{j-1}\pi_{j-1}(t)) \\ &= \phi_j^2(t) + \beta_{j-1}\phi_j(t)\pi_{j-1}(t). \end{aligned}$$

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

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

$$d_j = u_j + q_j,$$

$$q_j = u_j - \alpha_j A p_j,$$

$$p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j),$$

▶▶ vector  $d_j$  is no longer needed.

## ALGORITHM : 5. Conjugate Gradient Squared

1. Compute  $r_0 := b - Ax_0$ ;  $r_0^*$  arbitrary.
2. Set  $p_0 := u_0 := r_0$ .
3. For  $j = 0, 1, 2, \dots$ , until convergence Do:
4.  $\alpha_j = (r_j, r_0^*) / (Ap_j, r_0^*)$
5.  $q_j = u_j - \alpha_j Ap_j$
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.  $p_{j+1} = u_{j+1} + \beta_j(p_j + \beta_j p_j)$
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 → poor 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,

$$r'_j = \psi_j(A)\phi_j(A)r_0$$

where  $\phi_j(t) =$  BCG residual polynomial, and ..

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

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

$\omega_i$  chosen to 'smooth' convergence [steepest descent step]

## ALGORITHM : 6. BCGSTAB

---

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