Krylov subspace methods (Continued)

- Practical variants: restarting and truncating
- Symmetric case: The link with the Lanczos algorithm
- The Conjugate Gradient algorithm
- See Chapter 6 of text for details.

**Difficulty:** As \( m \) increases, storage and work per step increase fast.

**First remedy:** Restart. Fix \( m \) (dim. of subspace)

**ALGORITHM : 1.** Restarted GMRES (resp. Arnoldi)

1. (Re)-Start: Compute \( r_0 = b - Ax_0 \),  
   \( v_1 = r_0 / (\beta := \|r_0\|_2) \).
2. Arnoldi Process: generate \( \bar{H}_m \) and \( V_m \).
3. Compute \( y_m = H^{-1}m \beta e_1 \) (FOM), or \( y_m = \text{argmin} \|\beta e_1 - \bar{H}_m y\|_2 \) (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
\[
h_{j+1,j} v_{j+1} = Av_{j} - \sum_{i=j-k+1}^{j} h_{ij} v_{i}
\]

- Each \( v_j \) is made orthogonal to the previous \( k \) \( v_i \)'s.
- \( x_m \) still computed as \( x_m = x_0 + V_m H_m^{-1} \beta e_1 \).
- It can be shown that this is 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]:**

- Write the LU decomposition of \( H_m \) as \( H_m = L_m U_m \)
- \( x_m = x_0 + V_m U_m^{-1} L_m^{-1} \beta e_1 \equiv x_0 + P_m z_m \)

**Structure of \( L_m, U_m \) when \( k = 3 \)**

\[
L_m = \begin{bmatrix} 1 & x & x \\ x & 1 & x \\ x & x & 1 \end{bmatrix}, \quad U_m = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}
\]

- \( p_m = u_{mm}^{-1} [v_m - \sum_{i=m-k+1}^{m-1} u_{im} p_i] \)
- \( z_m = [z_{m-1} \zeta_m] \)
Can update $x_m$ at each step:

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

**Algorithm:**

Until convergence do:

1. Update LU factorization of $H_m \rightarrow H_m = L_m U_m$
2. $p_m = u_m^{-1} [v_m - \sum_{i=m-k+1}^{m-1} u_m p_i]$
3. $x_m = x_{m-1} + \zeta_m p_m$
4. $h_{m+1,m} v_{m+1} = A v_m - \sum_{i=m-k+1}^{m} h_{im} v_i$ (Arnoldi step)

Enddo

- Requires $2k + 1$ vectors [in addition to solution]

**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 [YS, 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 $H_m$

**The symmetric case:** Observation

**Observe:** When $A$ is real symmetric then in Arnoldi’s method:

$$H_m = V_m^T A V_m$$

must be symmetric. Therefore

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

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

- We can write

$$H_m = \begin{bmatrix} \alpha_1 & \beta_2 & \beta_3 & \beta_4 & \ldots \\ \beta_2 & \alpha_2 & \beta_3 & \beta_4 & \ldots \\ \beta_3 & \beta_2 & \alpha_3 & \beta_4 & \ldots \\ \vdots & \vdots & \vdots & \ddots & \beta_m \\ \beta_m & \beta_{m-1} & \beta_{m-2} & \ldots & \alpha_m \end{bmatrix}$$

(1)

The $v_i$’s satisfy a 3-term recurrence [Lanczos Algorithm]:

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

- Simplified version of Arnoldi’s algorithm for sym. systems.

**Symmetric matrix + Arnoldi \rightarrow Symmetric Lanczos**
The Lanczos algorithm

**ALGORITHM : 2. Lanczos**

1. Choose an initial vector \( v_1 \), s.t. \( \|v_1\|_2 = 1 \)
   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

**ALGORITHM : 3. Lanczos Method for Linear Systems**

1. Compute \( r_0 = b - Ax_0, \beta_1 := \|r_0\|_2, \) 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 \( m > 1 \) 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

Lanczos algorithm for linear systems

**ALGORITHM : 4. 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 \( m > 1 \) 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

Usual orthogonal projection method setting:

- \( L_m = K_m = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \)
- 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 tridiagonal nature of \( H_m \) (DIOM);
3. Conjugate gradient - derived from (2)
In D-Lanczos, \( r_m = \text{scalar} \times v_{m-1} \) and \( p_m = \text{scalar} \times [v_m - \beta_m p_{m-1}] \).

And we have \( x_m = x_{m-1} + \xi_m p_m \).

So there must exist an update of the form:

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

\( \text{Note: } p_m \text{ is scaled differently and } \beta_m \text{ is not the same} \)

\( \text{.. In CG, index of } p_m \text{ aligned with that of } r_m \text{ – so } p_j \text{ replaced by } p_{j-1}. \)

\( \text{Note: the } p_i \text{'s are } A \text{-orthogonal} \)

\( \text{The } r_i \text{'s are orthogonal.} \)

Question: How to apply preconditioning?