DN2222
Applied Numerical Methods
- part 2:
Numerical Linear Algebra

Lecture 6
Eigenvalues (cont)

2012-11-19

Transformation algorithms

- For matrices of moderate size, the standard method to compute eigenvalues is through similarity transformations.
- If there are n linearly independent eigenvectors. Let them build the non-singular X and then

  $AX = XD$, giving $A = XDX^{-1}$, where $D = diag(\lambda_k)$

  in which case we say A is diagonalizable.
- Not all matrices are diagonalizable, but one can transform any square matrix into *triangular* form by a unitary (orthogonal) similarity transformation. That is what the *Schur theorem* says.
- Any practical transformation algorithm is divided into two phases: an initial reduction (into *Hessenberg form*, by $n-2$ elementary transformations), followed by an iterative phase where the remaining sub-diagonal elements are shrunk (usually by the *QR algorithm*)

Final algorithm - QR

- Given the transformation into Hessenberg form. Let $A_1 = H$ (the Hessenberg matrix) and $U_1 = W$ (the transformation matrix).
  - for k=1,2,...
  - Factorize $A_k = Q_k R_k$ with Q_K orthogonal and R_k upper triangular.
  - Multiply $A_{k+1} = R_k Q_k$
  - Accumulate $U_{k+1} = U_k Q_k$
- Then $A_{k+1} = U_k^T A U_k$ is (almost) upper triangular.
- Convergence for the QR algorithm without shifts is slow.
- If the matrix A_k is singular, ie has a zero eigenvalue, one diagonal element of R_k will be zero, usually the last. Then the remaining $(n-1) \times (n-1)$ sub-matrix contains the other eigenvalues. Shrinking the size of the problem like this is called *deflation*.

Final algorithm - QR with shifts

- Given the transformation to Hessenberg form, $A = WHW^T$. Let $A_1 = H$ (the Hessenberg matrix) and $U_1 = W$ (the transformation matrix).

  - for k=1,2,...
  - Choose shift σ_k
  - Factorize $A_k = Q_k R_k$
  - Multiply $A_{k+1} = R_k Q_k$
  - Accumulate $U_{k+1} = U_k Q_k$

- The new A_{k+1} is still an orthogonal similarity transformation of A_k:

$$A_{k+1} = R_k Q_k + \sigma_k I = (Q_k^T (A_k - \sigma_k I) Q_k + \sigma_k I = Q_k^T A_k Q_k$$

- Common choices for shifts are:
  - *Newton shift:* The last diagonal element (is the eigenvalues of the last 1×1 block).
  - *Wilkinson shift:* An eigenvalue of the last 2×2 sub-matrix. (An advantage with this method is that it can give complex shifts, even with a real matrix).

Iterative eigenvalue algorithms

- The power method $x_k = Ax_{k-1}$ has slow convergence if the largest eigenvalue is not well isolated. It is also not efficient - it throws away all computed x_j.

- If we save all the vectors we get the *Krylov subspace*

$$K_k(A, x_0) = \{x_0, Ax_0, A^2 x_0, \ldots, A^{k-1} x_0\}$$

- The vectors in the Krylov space will be less and less independent. The *Arnoldi algorithm* will make an orthonormal basis from them.

- Start with the first vector. From the next, remove all dependency on the previous (one step from the Gram-Schmidt algorithm). Then normalize the remaining vector and put it as the next basis vector.

Arnoldi algorithm

- Start with $q_1 = x/||x||_2$
  - for k=1,2,...
  - $u = Aq_k$
  - for j=1,2,...,k.
    $h_{j,k} = q_j^H u$
    $u = u - q_j h_{j,k}$
  - $h_{k+1,k} = ||u||_2$
  - $q_{k+1} = u/h_{k+1,k}$

Lanczos algorithm

- Start with $q_1 = x/||x||_2$

- for k=1,2,...
- $u = Aq_k - q_{k-1}\beta_{k-1}$
- $\alpha_k = q_k^H u$
- $u = u - q_k\alpha_k$
- $\beta_k = ||u||_2$
- $q_{k+1} = u/\beta_k$

- The Lanczos uses much less operations per iteration than Arnoldi.
- Lanczos only saves the last two vectors.
- Thus Lanczos manages much bigger problems.
- However, orthogonality gets lost after a while. Re-orthogonalization is as costly as Arnoldi.

Spectral transformation

- A standard practice to find eigenvalue to a large matrix is to apply a Krylov space algorithm, Lanczos or Arnoldi, to a shift invert spectral transformation:

$$C = (A - \sigma B)^{-1}B, \quad \text{with eigenvalues} \quad \theta_j = 1/(\lambda_j - \sigma)$$