

## CLS361 Programming assignment 4:

1. Develop a **Matlab** program for the **QR** algorithm based on *Householder reflections* for real  $m \times n$  matrices. Carry out the **QR** decomposition for random matrices (order of  $30 \times 30$ ) and compare your results with the native *qr* available in **Matlab**.
2. Develop a function for reducing a real square symmetric matrix to a tridiagonal form using *Householder* based similarity transformations.
3. Suppose that the

$$\text{function : } \mathbf{v} = \text{house}(\mathbf{x})$$

discussed in the class returns the approximation  $\hat{\mathbf{v}}$  for the *Householder vector*  $\mathbf{v}$  after round-offs. If

$$\hat{\mathbf{P}} = \mathbf{I} - 2 \frac{\hat{\mathbf{v}}\hat{\mathbf{v}}^T}{\hat{\mathbf{v}}^T\hat{\mathbf{v}}}$$

then show/verify the following:

- (a)  $\|\mathbf{P} - \hat{\mathbf{P}}\|_2 = O(\mu)$
  - (b)  $fl(\hat{\mathbf{P}}\mathbf{A}) = \mathbf{P}(\mathbf{A} + \mathbf{E})$ , where  $\|\mathbf{E}\|_2 = O(\mu\|\mathbf{A}\|_2)$
  - (c)  $fl(\mathbf{A}\hat{\mathbf{P}}) = (\mathbf{A} + \mathbf{E})\mathbf{P}$ , where  $\|\mathbf{E}\|_2 = O(\mu\|\mathbf{A}\|_2)$
4. Program the following **QR** iteration (read the note at the end of this assignment) for computing the eigenvalues and eigenvectors of a general real matrix of size  $n \times n$ . For symmetric matrices, first carry out a tridiagonalization. Show/verify that the **QR** iteration preserves the tridiagonal structure for a symmetric matrix. Test your results on randomly generated matrices by comparing with standard **Matlab** functions.
  5. Use your eigen-computation routine to compute the *SVD* of an  $m \times n$  matrix  $\mathbf{A}$ . Note that if the *SVD* of  $\mathbf{A}$  is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$$

then

$$\mathbf{V}^T(\mathbf{A}^T \mathbf{A})\mathbf{V} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) \in \mathbb{R}^{n \times n}$$

and

$$\mathbf{U}^T(\mathbf{A}\mathbf{A}^T)\mathbf{U} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2, 0, \dots, 0) \in \mathbb{R}^{m \times m}$$

Moreover, if

$$\mathbf{U} = [ \mathbf{U}_1 \quad \mathbf{U}_2 ]$$

where  $\mathbf{U}_1$  is the matrix of the first  $n$  columns of  $\mathbf{U}$  and  $\mathbf{U}_2$  is the matrix of the last  $m - n$  columns of  $\mathbf{U}$ , and we define

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{V} & \mathbf{V} & \mathbf{0} \\ \mathbf{U}_1 & -\mathbf{U}_1 & \sqrt{2}\mathbf{U}_2 \end{bmatrix}$$

then

$$\mathbf{Q}^T \begin{bmatrix} \mathbf{0} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \mathbf{Q} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n, -\sigma_1, -\sigma_2, \dots, -\sigma_n, \underbrace{0, \dots, 0}_{m-n})$$

Compare your *SVD* computation with that of the standard **Matlab** function for moderate size matrices.

6. Consider a straight line  $ax + by - c = 0$  for any choice of parameters  $a, b, c$ . Randomly select 100 points on the straight line and shift these by adding 2-dimensional isotropic Gaussian random noise (you can add zero mean Gaussian noise of variance  $\sigma$  independently to  $x$  and  $y$ ). Vary  $\sigma$  and obtain least square line fits using your *SVD* routine. Add comments/analysis in a separate file.

**Note:** The **QR** iteration consists of a sequence of orthogonal transformations

$$\begin{aligned} \mathbf{A}_k &= \mathbf{Q}_k \mathbf{R}_k \\ \mathbf{A}_{k+1} &= \mathbf{R}_k \mathbf{Q}_k \quad (= \mathbf{Q}_k^T \mathbf{R}_k \mathbf{Q}_k) \end{aligned}$$

The following (non-obvious) theorem is the basis of the algorithm for a real matrix  $\mathbf{A}$ :

1. If  $\mathbf{A}$  has eigenvalues of distinct absolute value  $|\lambda_i|$ , then  $\mathbf{A}_k \rightarrow$  [upper-triangular form] as  $k \rightarrow \infty$ . The eigenvalues appear on the diagonal in increasing order of absolute magnitude.

2. If  $\mathbf{A}$  has eigenvalue  $|\lambda_i|$  of multiplicity  $p$ , then  $\mathbf{A}_k \rightarrow$  [upper-triangular form] as  $k \rightarrow \infty$ , except for a diagonal block matrix of order  $p$ , whose eigenvalues  $\rightarrow \lambda_i$ .

If  $\mathbf{A}$  is real  $n \times n$  then there exists an orthogonal  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  such that

$$\mathbf{Q}^T \mathbf{A} \mathbf{Q} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1m} \\ \mathbf{0} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2m} \\ \vdots & \vdots & \ddots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{mm} \end{bmatrix}$$

where each  $\mathbf{R}_{ij}$  is either a  $1 \times 1$  matrix or a  $2 \times 2$  matrix having complex conjugate eigenvalues.

The above result is called the **Real Schur decomposition theorem**.

The **QR** iteration for general real matrices converges to a **Real Schur** form. It is customary to start with  $\mathbf{H}_0 = \mathbf{U}_0^T \mathbf{A} \mathbf{U}_0$  such that  $\mathbf{H}_0$  is in upper *Hessenberg* form ( $h_{ij} = 0, i > j + 1$ ). A real square matrix can be reduced to the *Hessenberg form* using *Householder* operations. The *Hessenberg form* is preserved by the **QR** iteration.

For a real symmetric matrix all eigenvalues are real. If  $\lambda_i$  has a multiplicity of  $p$  then the matrix can be split in to sub-matrices which can be diagonalized separately.

In the proof of the theorem quoted above, one finds that in general a sub-diagonal element converges to 0 like

$$a_{ij}^{(k)} \sim \left( \frac{\lambda_i}{\lambda_j} \right)^k$$

Although  $\lambda_i < \lambda_j$ , convergence can be slow if they are close. Convergence can be accelerated by a technique called *shifting*: if  $s$  is any constant, then  $\mathbf{A} - s\mathbf{I}$  has eigenvalues  $\lambda_i - s$ . If we decompose

$$\begin{aligned} \mathbf{A}_k - s_k \mathbf{I} &= \mathbf{Q}_k \mathbf{R}_k \\ \mathbf{A}_{k+1} &= \mathbf{R}_k \mathbf{Q}_k + s_k \mathbf{I} \\ &= \mathbf{Q}_k^T \mathbf{R}_k \mathbf{Q}_k \end{aligned}$$

then, the convergence is determined by the ratio

$$\frac{\lambda_i - s_k}{\lambda_j - s_k}$$

The idea is to choose the shift  $s_k$  at each stage to maximize the rate of convergence. A good choice for the shift initially would be  $s_k$  close to  $\lambda_1$ ,

the smallest eigenvalue. Then the first row of off-diagonal elements would tend rapidly to zero. However  $\lambda_1$  is not known apriori. A very effective strategy is to compute the eigenvalues of the leading  $2 \times 2$  sub-matrix of  $\mathbf{A}_k$  at each stage and set  $s_k$  to the smaller. One can show that the convergence of the algorithm with this strategy is generally cubic (and at worst quadratic) for degenerate eigenvalues.

Good luck.