AMS213A Project 3

Sutirtha Sengupta (Student id#:1566652)

February 27, 2017

Problem 1

For a given $m \times n$ matrix A, the QRalgo subroutine (in the module LinAl.f90) first constructs the QR decomposition of A given by

$$A = QR \tag{1}$$

using the QRdec subroutine (from PROJECT 3) which constructs the QR decomposition of the matrix A and it returns the upper triangular matrix R in the upper triangle of A. It then iteratively computes the steps

$$A = RQ \tag{2}$$

$$V = VQ \tag{3}$$

which reduces A into diagonal form with the eigenvalues of the original matrix A as the diagonal entries of A and the corresponding eigenvectors as the columns of the matrix V. The iteration is terminated if the norm of residual vector r given by

$$r_i = (a_{ii} - \lambda_i)/a_{ii} \tag{4}$$

is within machine accuracy (i.e. 10^{-8} using single precision for floating point arithmetic).

The Fortran program myprog.f90 computes the eigenvalues and eigenvectors of the matrix stored in Amat.dat given by

$$A = \begin{pmatrix} 5 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \tag{5}$$

using the QR algo routine which returns A in diagonal form (outputted onto screen as the D_lambda matrix) containing the eigenvalues along the diagonal. For each eigenvalue (λ), the error in the solution is computed as

$$E = ||Av - \lambda v|| \tag{6}$$

where v is the eigenvector of A corresponding to the eigenvalue λ which are obtained as column vectors of the matrix V returned by the QRalgo subroutine.
Analytically, solving for the characteristic equation for A, we have

$$\begin{vmatrix} 5-\lambda & 1 & 1 \\ 1 & 3-\lambda & 1 \\ 1 & 1 & 3-\lambda \end{vmatrix} = 0 \tag{7}$$

$$\implies \lambda = 6, 3, 2 \tag{8}$$

For $\lambda = 6$, the eigenvector is given by

$$\frac{1}{\sqrt{6}}(2,1,1)^T = \begin{pmatrix} 0.816496581 \\ 0.40824829 \\ 0.40824829 \end{pmatrix} \tag{9}$$

, for $\lambda = 3$, the eigenvector is given by

$$\frac{1}{\sqrt{3}}(-1,1,1)^T = \begin{pmatrix} -0.577350269 \\ 0.577350269 \\ 0.577350269 \end{pmatrix} \tag{10}$$

and for $\lambda = 2$, the eigenvector is given by

$$\frac{1}{\sqrt{2}}(0,-1,1)^T = \begin{pmatrix} 0 \\ -0.707106781 \\ 0.707106781 \end{pmatrix} \tag{11}$$

In comparison, the code gives the eigenvalues as

$$\lambda = 6.00000143,\ 2.99999690,\ 1.99999809 \tag{12}$$

which are all accurate to within machine precision compared to the analytical values given by Eq. 8. The errors in the solution, computed with Eq. 6 are $5.22892788E-06$, $3.67053959E-04$, $3.67430650E-04$, corresponding to the three eigenvectors given by

$$\begin{pmatrix} -0.816497684 \\ -0.408247292 \\ -0.408247501 \end{pmatrix}, \begin{pmatrix} 0.577348828 \\ -0.577610135 \\ -0.577091098 \end{pmatrix}, \begin{pmatrix} -2.12191662E-04 \\ -0.706894577 \\ 0.707318783 \end{pmatrix} \tag{13}$$

The program when executed on the matrix in the given file Emat1.dat, returns the sum of its eigenvalues as 15.0000143, which is with machine accuracy of value of the trace

of the matrix (= 15). The error in the solution, computed with Eq. 6, for the different eigenvalues are given by

$$\begin{aligned}
\lambda = 7.39054871 &: \; E = 7.37172377E-06 \\
\lambda = 3.54039669 &: \; E = 3.13665427E-04 \\
\lambda = 2.43474150 &: \; E = 3.13650817E-04 \\
\lambda = 1.35663199 &: \; E = 5.27240081E-06 \\
\lambda = 0.277695864 &: \; E = 2.14503629E-07
\end{aligned} \tag{14}$$

The program when executed on the (non-symmetric) matrix in the given file Emat2.dat, which has the same trace as the one in Emat1.dat, also returns the sum of its its diagonal elements as 15.0000143, suggesting that the eigenvalues are indeed given by the diagonal elements of the matrix as follows:

$$\lambda = 6.95643425,\ 5.14561939,\ 1.89562488,\ 1.50989044,\ -0.507554591 \tag{15}$$

This is not a surprising finding that the QR algorithm gives the correct eigenvalues in the diagonal entries, because of the theorem (in Lecture 11) which states that the QR algorithm reduces any matrix A to its Schur form, which has the eigenvalues of A along its diagonal.

Problem 2

The Fortran program myprog.f90 computes the eigenvalues and eigenvectors of any square (symmetric or non-symmetric) matrix A given as argument by the user in the form of a file containing the size of the matrix and its elements. Firstly, it checks whether the matrix is symmetric or not and accordingly calls the relevant LAPACK routine (SSYEV for symmetric case and SGEEV for non-symmetric case) returning the eigenvalues and eigenvectors as well as the error in solution computed with Eq. 6.
For Emat.1.dat, the eigenvalues and errors in solution returned by the program are given by

$$\begin{aligned}
\lambda = 7.39054251 &: \; E = 2.27436840E-06 \\
\lambda = 3.54039454 &: \; E = 1.29700038E-06 \\
\lambda = 2.43473601 &: \; E = 8.01898466E-07 \\
\lambda = 1.35663283 &: \; E = 1.11877853E-06 \\
\lambda = 0.277696103 &: \; E = 3.14191311E-07
\end{aligned} \tag{16}$$

which on comparison with Eq. 14 shows that the LAPACK routine yields a more accurate solution compared to that obtained using the QR algorithm (PROBLEM 1).

For Emat.2.dat, the eigenvalues and errors in solution returned by the program are given by

$$\lambda = 6.95642996,\ 5.14562130,\ -0.507555306,\ 1.89562285,\ 1.50989044 \tag{17}$$

which on comparison with Eq. 15 shows that the eigenvalues computed using the QR algorithm agree with those obtained using the LAPACK routine to with machine accuracy.

Problem 3

The Fortran routine lehmer_mat constructs the Lehmer matrix L of given size m defined by

$$l_{ij} = \frac{min(i,j)}{max(i,j)} \tag{18}$$

The Fortran program myprog.f90 computes the eigenvalues of the Lehmer matrix L using the QR algorithm (PROBLEM 1) as well as using LAPACK routine (SSYEV) for different values of $m = 5, 10, 50, 100$. The eigenvalues computed with the QR algorithm (evalues_code.dat) are sorted using a standard heapsort algorithm (adapted from Numerical Recipes) in order to compare against those obtained from the LAPACK routine (evalues_LAPACK.dat) as shown the following figures (for different $m$values):

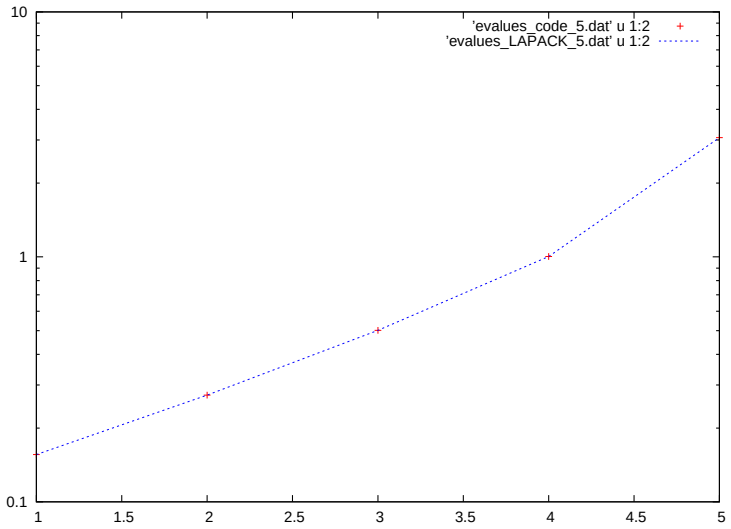

Figure 1: Plot of $|\lambda_i|$ vs i for $m = 5$

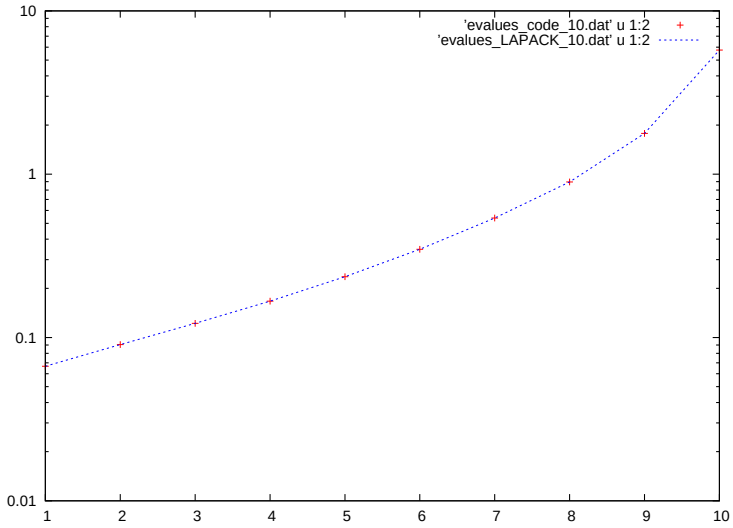

Figure 2: Plot of $|\lambda_i|$ vs i for $m = 10$

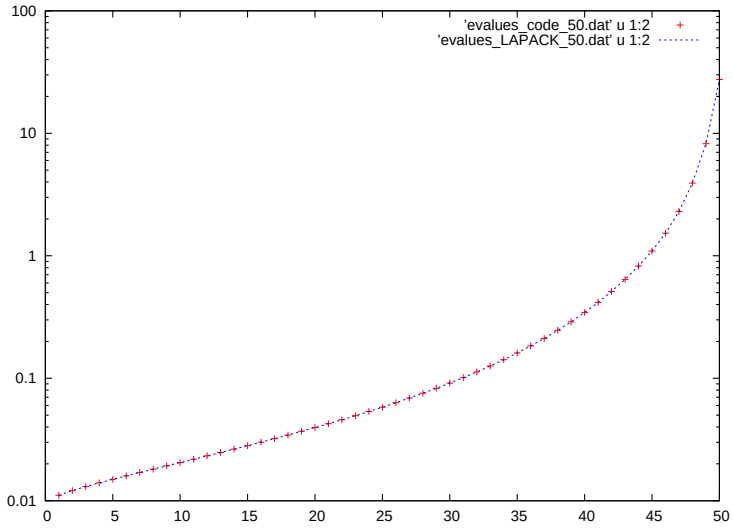

Figure 3: Plot of $|\lambda_i|$ vs i for $m = 50$

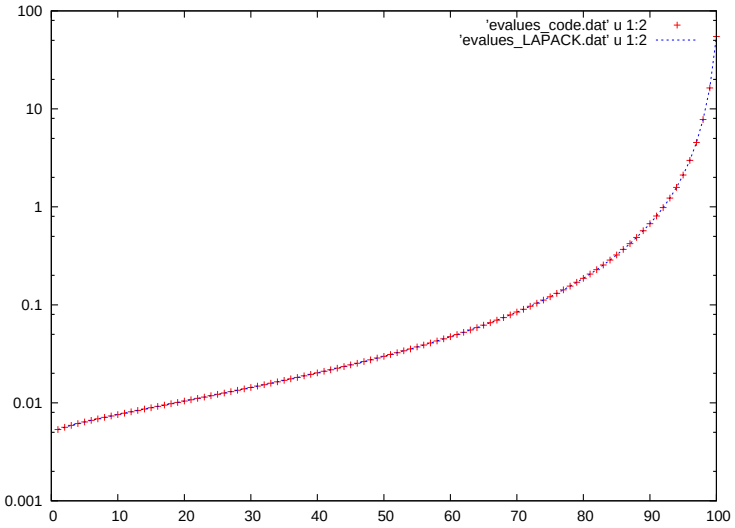

Figure 4: Plot of $|\lambda_i|$ vs i for $m = 100$

The compute-time for the QR method vs the LAPACK routine are shown Table 1, which shows that the compute-time for the QR-method increases rapidly for increasing m, whereas for LAPACK routine it scales much more smoothly.

m	QR	LAPACK
5	$2.88999872E-04$	$2.59999884E-04$
10	$6.61499985E-03$	$2.70999735E-04$
50	2.43463087	$5.45000192E-04$
100	51.6990929	$3.68700037E-03$

Table 1: Comparison of compute-time of QR algorithm vs LAPACK routine for finding eigenvalues of a Lehmer matrix.

The primary reason behind this difference is due to the fact that LAPACK uses the QR method with shift and deflation, thereby progressively reducing the size of the matrix to enable such acceleration in compute-time.