Revised 9/11/15

LAB 4: Unitary Diagonalization of Matrices, QR Algorithm, Finite Fourier Transform, and Fast Fourier Transform

In this lab you will use MATLAB to study the following topics:

- Diagonalization of hermitian and normal matrices by unitary matrices
- The Fourier matrix and Fourier basis for $\mathbf{C}^n$
- Diagonalization of circulant matrices by the Fourier matrix
- The fast Fourier transform
- The QR algorithm for fast computation of eigenvalues

MATLAB *Preliminaries*

Lab Write-up: After opening your diary file, type the comment line

```
% Math 550 MATLAB Lab Assignment #4
```

at the MATLAB prompt. Type `format compact` so that your diary file will not have unnecessary spaces. Put labels to mark the beginning of your work on each part of each question, so that your edited lab write-up has the format

```
% Question 1 (a) ...
      ⋮
% Question 1 (b) ...
```

and so on. Be sure to answer all the questions in the lab assignment. Insert comments in your diary file as you work through the assignment.

Random Seed: When you start your MATLAB session, initialize the random number generator by typing

```
rand('seed', abcd)
```

where *abcd* are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

Question 1. Unitary Diagonalization of Matrices

Generate random 4×4 complex matrices A and B by

```
A = rand(4) + i*rand(4);   B = A + A'
```

(here $i = \sqrt{-1}$). Note that the real and imaginary parts of A will be different random matrices (each call to the random matrix generator produces a different output). Also, when A is a complex matrix, then the MATLAB notation `A'` means the *conjugate transpose* A^{H}. Hence $B = B^{\mathrm{H}}$ is a hermitian symmetric matrix.

(a) Use MATLAB to compute the matrix S of normalized eigenvectors of A and the diagonal eigenvalue matrix L of A by

```
[S, L] = eig(A)
```

Since A is a random matrix, it should have 4 distinct eigenvalues. Verify by MATLAB that $S^{-1}AS = L$. Then use MATLAB calculations to answer the following questions:

(i) Are the columns of S mutually orthogonal?

(ii) Is $S' * A * S = L$? (**Note:** Here S' means S^{H}.)

(iii) Is $A * A' = A' * A$?

Explain the relations among these questions using the eigenvalue/eigenvector properties of normal matrices.

(b) Let U be the matrix of normalized eigenvectors of B and let D be the diagonal eigenvalue matrix of B. Before making any computations answer the following questions using the theory of hermitian matrices:

(i) What can you predict about the eigenvalues of B?

(ii) Are the columns of U mutually orthogonal? Why?

(iii) Is $U' * B * U = D$? Why?

Now use MATLAB to calculate `[U, D] = eig(B)` and confirm your answers to *(i)*, *(ii)*, and *(iii)*.

(c) Let C be the complex matrix generated by the following commands:

```
s = rand(1), t = rand(1)
C = s*B^2 + i*t*B^3
```

(notice that each call to `rand(1)` produces a different random number). Answer the following questions using general matrix algebra results (no numerical calculation):

(i) Show that `C'*C = C*C'`. What does this tell you about the eigenvectors of C?

(ii) Show that every eigenvector of B is also an eigenvector of C.

(iii) Show that you can determine the eigenvalues of C from the eigenvalues of B.

Now use MATLAB to calculate

```
E = U'*C*U
```

Then use the results *(ii)* and *(iii)* to answer the following questions:

(iv) Why is the matrix E diagonal?

(v) What is the relation between D and E?

Check your answer to *(v)* by MATLAB.

Question 2. The Fourier Matrix and Fourier Basis

For this question, read §3.5 of Strang's book and Section 2: *Finite Fourier Transform* of the supplementary class notes (posted on the course web page).

(a) Fourier Matrix: The kth column of the $n \times n$ Fourier matrix F_n consist of the consecutive powers of w^{k-1} (from 0 to $n-1$), where $w = \mathrm{e}^{2\pi\,\mathrm{i}\,/n}$. These columns all have squared lenght N. In particular, taking $\mathbf{y} = \mathbf{e}_k$ as the kth standard basis vector and putting in a normalization factor, we obtain the vectors $\mathbf{u}_k = (1/\sqrt{N})F_n\mathbf{e}_k$. The vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ are the *Fourier orthonormal basis* for $\mathbf{C}^n$.

If $\mathbf{y} \in \mathbf{C}^n$ is a column vector, then the *Finite Fourier Transform* of $\mathbf{y}$ is the vector $Y = F_n^{\mathrm{H}}\mathbf{y}$. If A is an $n \times m$ matrix, the MATLAB command `fft(A)` takes the Discrete Fourier transform $F_n^{\mathrm{H}}A$ of each column of A. Hence `Fn = fft(eye(n))'` generates the matrix F_n, since `eye(n)` generates the $n \times n$ identity matrix (whose columns are the vectors $\mathbf{e}_k$), and the prime `'` in MATLAB gives the Hermitian transpose. Check this by typing

```
F2 = fft(eye(2))'
F4 = fft(eye(4))'
F8 = fft(eye(8))'
```

and compare the matrices `F2` and `F4` with the examples of Fourier matrices in the supplementary class notes, page 4.

Use MATLAB to calculate `w = exp(2*pi*i/8)` and the column vectors

```
u1 = (1/sqrt(8))*ones(8,1)
u2 = (1/sqrt(8))*[1 ; w ; w^2 ; w^3 ; w^4; w^5 ; w^6 ; w^7]
```

Show that `u1` and `u2` are mutually perpendicular by calculating `u1'*u2`. Then show that the matrix $(1/\sqrt{8})F_8$ is unitary by calculating `norm((1/8)*F8*F8' - eye(8))` . In both cases you should get numbers on the order of 10^{-15} (considered as zero for numerical purposes).

(b) Sampling and the Nyquist Point To carry it out this Matlab exploration, you will need the m-file `fftgui.m` (*Finite Fourier transform graphic user interface*). This file can be downloaded from the Numerical Computing in Matlab (NCM) directory on the Math 642:550 course page or from

`www.mathworks.com/moler/ncmfilelist.html`

If `y` is a column vector with complex entries, then the MATLAB command `fftgui(y)` generates a window with four subplots: The top two plots indicate the real and imaginary parts of `y`, and the bottom two indicate the real and imaginary parts of the discrete Fourier transform `Y = fft(y)`, all plotted as stem graphs (lollipops). Try this with a vector of length 16 with a single 1 in the first entry and the other entries zero:

```
y = zeros(16, 1); y(2) = 1; fftgui(y)
```

. Here the notation `y(k)` means that we are following the MATLAB indexing convention, numbering the entries from 1 to 16.

(i) Describe the plots of `fft(y)` in each case in terms of sampled sine and cosine waves.

(see Section 1 of the supplementary class notes).

Use the uparrow to repeat this, taking `y(2) = 1, y(3) = 1` and the other entries zero. In these plots consider the points on the horizontal axis numbered from 0 to $N-1$, where N is the length of `y`. The point numbered $N/2$ on the frequency axis is called the *Nyquist point*. Hence when $N = 16$, then the Nyquist point is the 9th dot from the left.

(ii) What symmetries around the Nyquist point do the graphs of `real(fft(y))` and `imag(fft(y))` have for each of your plots?

(See Remark 2.8 in the supplementary class notes; view the plots as periodic funtions of period 16.)

Now close the `fftgui` figure window and enter the command `fftgui` at the MATLAB prompt. The figure window will reappear with all plots initialized to `y = zeros(32, 1)`. You can use the mouse to move any of the points in the upper (or lower) two plot windows, Use the mouse in the `real(y)` plot window to draw a random waveform with large and small oscillations. Since the period is now $N = 32$, the Nyquist point is the 17th dot from the left. Check that the graphs of `real(fft(y))` and `imag(fft(y))` have the correct symmetries around this point.

Question 3. Diagonalization of Circulant Matrices

For this question, read section 3 of the Finite Fourier Transform class notes.

(a) Shift Operator Create the following MATLAB function m-file that generates the $n \times n$ *shift matrix* S:

```
function S = shift(n)
S = zeros(n,n); S(1,n) = 1;
for k = 1:n-1
    S(k+1,k) = 1;
end
```

Save this file as `shift.m`. Then test it by typing `S = shift(4)`. You should get the 4×4 shift matrix (the 3×3 shift matrix is shown on page 6 of the notes).

Generate a random integer vector $\mathbf{v} \in \mathbf{R}^4$ by `v = rvect(4)`. Calculate `S*v` to see that S shifts the components of $\mathbf{v}$ cyclically. Check by MATLAB that `(1/4)*F4'*S*F4` is a diagonal matrix (where `F4` is the 4×4 Fourier matrix).

(i) What are the eigenvalues of `S`? Why?

(ii) What are the eigenvectors of `S`? Why?

(b) Circulant Matrices Use the random vector `v` from part **(a)** and powers of the matrix `S` to construct a 4×4 *circulant matrix* `C` whose first column is `v` (see Theorem 3.2 in the Notes). Then use the general theory of circulant matrices to answer the following:

(i) What are the eigenvalues of `C`? How are they related to the eigenvalues of S?

(ii) What are the eigenvectors of `C`? How are they related to the eigenvectors of S?

Check your answers by using MATLAB to calculate that `(1/4)*F4'*C*F4` is a diagonal matrix.

Question 4. The Fast Fourier Transform

Before working on this question, open a web browser window, go to the Math 550A home page, click on the *Fast Fourier Transform* links, and in the *Explanatory Material* part of the page click on *Graphical interpretation of DFT and FFT by Steve Mann.* The complete web address is

`http://wearcam.org/ece431/course_material/fourierop_and_dit.htm`

This page gives a very clear description of the DFT and FFT with graphics and flowcharts for the 8×8 Fourier matrix. Also review Section 4 of the Notes.

(a) Block Decomposition of the Fourier Matrix: The first step in the FFT is to split a signal vector $\mathbf{y}$ (of length $2n$) into two vectors $\mathbf{y}_{\text{even}}$ and $\mathbf{y}_{\text{odd}}$, each of length n. This step is called *downsampling.* The following function m-file generates the permutation matrix P_{2n} that does this:

```
function P = downsamp(n)
P = zeros(2*n, 2*n);          % Create 2n by 2n matrix of zeros
for j = 1:n
    P(j, 2*j - 1) = 1;        % Sort entries 1, 3, ..., 2n - 1 into rows 1, ..., n
    P(n + j, 2*j) = 1;        % Sort entries 2, 4, ..., 2n into rows n + 1, ..., 2n
end
```

Create and save this m-file under the name `downsamp.m`. Test it by creating a random vector `y = rvect(8)` and the 8×8 downsampling matrix `P = downsamp(4)`. Calculate `P*y` and check that the entries in positions 1, 3, 5, and 7 of `y` are the first four entries in `P*y`, and the entries in positions 2, 4, 6, and 8 of `y` are the last four entries in `P*y`.

The block form in the FFT uses a diagonal matrix D_n (see equation (28) in the Notes), which can be generated by the following function m-file:

```
function D = fftdiag(n)
D = zeros(n,n);
w = exp(pi*i/n);
for j = 1:n
  D(j, j) = w^(1-j) ;
end
```

Create and save this m-file under the name `fftdiag.m`.

With the two function m-files you have made you can now create the matrix F_4^{H} from the matrix F_2^{H}, and then the matrix F_8^{H} from F_4^{H}. Type

```
P4 = downsamp(2); D2 = fftdiag(2); F2 = fft(eye(2));
F4H = [F2,  D2*F2; F2,  -D2*F2]*P4
```

Compare `F4H` with the matrix `fft(eye(4))'`. Are they the same?

Now repeat the process to generate the 8×8 matrix F_8^{H}. First generate `P8 = downsamp(4)` and `D4 = fftdiag(4)`. Use these, the matrix `F4H` you just generated, and equation (29) in the supplementary notes to obtain a matrix `F8H`. Compare your `F8H` with the MATLAB-generated matrix F_8^{H} by calculating `norm(F8H - fft(eye(8))')`. This should be of size 10^{-15}, thus zero to the limits of machine computation.

(b) Speed Comparison: Generate the $2^{12} \times 2^{12}$ Fourier matrix and a random vector $\mathbf{x} \in \mathbf{R}^{4096}$ (**be sure to include the ; at the end of the line so that the matrix will not be displayed or written in your diary file**):

```
F = fft(eye(4096))'; x = rand(4096,1);
```

Now calculate the computation time to obtain Fourier transform of $\mathbf{x}$ in two ways: first by direct matrix multiplication and then by the fast Fourier transform (**be sure to type the semicolons as indicated**):

```
tic; F'*x; matrixtime = toc
tic, fft(x); ffttime = toc
speedup = floor(matrixtime/ffttime)
```

Now compare your observed speedup ratio using the FFT with the theoretical prediction. If $N = 2^k$, then computing the matrix $\times$ vector `F'*x` requires $N^2 = 2^{2k}$ scalar multiplications. Computing `fft(x)` needs at most $k2^{k-1}$ scalar multiplications by Equation (30) in the supplementary class notes (also see Strang pp. 193-4). The theoretical speedup ratio for the FFT over the plain (matrix $\times$ vector) multiplication, in terms of the scalar multiplications required, is thus

$$2^{2k}/[k2^{k-1}] = 2^{k+1}/k\ . \qquad (\star)$$

A similar speed advantage holds for the scalar additions required. Compute the theoretical speed advantage $(\star)$ for the value $k = 12$ that you have used, and compare it with the observed value `speedup`. Scalar multiplications require much more computing time than scalar additions, so it is the reduction in the number of multiplications that is crucial for the speedup given by the FFT.

Question 5. The QR Eigenvalue Algorithm

"The QR algorithm is one of the most important, widely used, and successful tools we have in technical computation. Several variants of it are in the mathematical core of MATLAB. They compute the eigenvalues of real symmetric matrices, real nonsymmetric matrices, and pairs of complex matrices, and the singular values of general matrices." (from *Numerical Computing with* MATLAB, by Cleve Moler, the original author of MATLAB).

(a) Single-shift QR algorithm : This algorithm is described on page 364 of Strang (equation (6)). To illustrate this algorithm, use the integer matrix `A = gallery(3)` from the MATLAB files (this matrix has eigenvalues 1, 2, and 3). Set `n = size(A,1), I = eye(n,n)` and then iterate the following MATLAB line using the up-arrow key:

```
s = A(n,n);   [Q, R] = qr(A - s*I);   A = R*Q + s*I
```

(i) Show by a hand calculation that the new matrix A produced by this line is orthogonally similar to the old matrix A.

Hence the new A has the same eigenvalues as the old A (although the eigenvectors will be different). Since the similarity is by an orthogonal transformation, the procedure is numerically stable.

Continue forming a new A from the old A until the last row of A becomes `[0  0  2.000]` (this should require about ten iterations).

(b) Deflation: Now replace the matrix A by the 2×2 matrix `A = A(1:2,1:2)`. Update the values of `n` and `I` and repeat the QR iteration from part **(a)** until this 2×2 matrix is upper triangular (this should require only two or three iterations).

(i) Prove by a hand calculation using orthogonal similarity of block matrices that the diagonal entries of the final 2×2 upper-triangular matrix are eigenvalues of the original matrix A. (Use the same method as in the inductive proof of the Schur triangular form.)

Use the MATLAB `eig` function on the original matrix `gallery(3)` to check that its eigenvalues are the same as the ones you have just calculated.

Final Editing of Lab Write-up:

After you have worked through all the parts of the lab assignment, edit your diary file. Include the MATLAB calculations, but remove errors that you made in entering commands and remove other material that is not directly related to the questions.