

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

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 if A is a complex matrix, then the MATLAB notation A' means the *conjugate transpose* A^H . Hence B is hermitian symmetric.

(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 $[U, D] = \text{eig}(B)$ 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).

(i) Show by matrix algebra (without numerical calculation) that $C^*C = C^*C'$. What does this tell you about the eigenvectors of C ?

(ii) Are the eigenvectors of B also eigenvectors of C ?

(iii) What is the relation between the eigenvalues of B and the eigenvalues of C ?

Now use MATLAB to calculate

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

Then 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 section 2: *Finite Fourier Transform* of the supplementary class notes (posted on the course web page).

(a) Fourier Matrix: The k th column of the $n \times n$ Fourier matrix F_n consist of the consecutive powers of ω^{k-1} (from 0 to $n-1$), where $\omega = e^{-2\pi i/n}$. If $\mathbf{y} \in \mathbf{C}^n$ is a column vector, then the *Finite Fourier Transform* of \mathbf{y} is the vector $Y = F_n \mathbf{y}$. In particular, taking $\mathbf{y} = \mathbf{e}_k$ as the k th standard basis vector, we obtain the vector $\mathbf{u}_k = F_n \mathbf{e}_k$. The vectors $\{\mathbf{u}_1, \dots, \mathbf{u}_n\}$ are the *Fourier basis* for \mathbf{C}^n .

The built-in MATLAB function `fft` takes the Discrete Fourier transform of each column of a matrix argument. 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). 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 Class Notes, pages 6 and 7.

Note: In Strang, page 193, the *Fourier matrix* is defined using $w = e^{2\pi i/n}$, which is the complex conjugate of ω . Hence Strang's matrix F is the complex conjugate \overline{F}_n . When multiplied by a factor of $1/n$, Strang's Fourier matrix becomes the inverse of the matrix F_n defined here.

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

```
u2 = [1 ; w ; w^2 ; w^3 ; w^4 ; w^5 ; w^6 ; w^7]
```

Show that `u2` is the same as the second column of `F8` by calculating `norm(u2 - F8(:,2))`. Use MATLAB to 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) Fourier Basis: Open a web browser window, go to the Math 550A home page, click on the *Fast Fourier Transform* link, and at the end of the *Explanatory Material* part of the page click on *FFT Laboratory*. The complete web address is

```
http://sepwww.stanford.edu/oldsep/hale/FftLab.html
```

The FFT Laboratory shows four windows: The top two indicate the real and imaginary parts of a function $f(x)$, and the bottom two indicate the real and imaginary parts of the discrete Fourier transform $F(k)$ of $f(x)$. Here $f(x)$ is sampled at 16, 32, or 64 points; you choose **Length** (the number of sample points) by clicking a button at the bottom of the page. The real and imaginary parts of the components of $\mathbf{f}(\mathbf{x})$ and $\mathbf{F}(\mathbf{k})$ are indicated by *lollipops* which you adjust using the mouse. The components of $\mathbf{f}(\mathbf{x})$ are relative to the standard basis for \mathbf{C}^n , while the components of $\mathbf{F}(\mathbf{k})$ are relative to the Fourier basis of \mathbf{C}^n (the columns of the Fourier matrix).

Leave the **Origin Centered** box unchecked, set **Length** to 16, and **Editing** to **Draw**. Click the **Zero All** button. Now use the mouse to lift the first (open) circle in the upper-left (**Real $\mathbf{f}(\mathbf{x})$**) graph above the axis as much as possible to make a lollipop. Then $\mathbf{f}(\mathbf{x})$ is a signal with value 1 at time zero, and zero values elsewhere ($\mathbf{f}(\mathbf{x})$ corresponds to the first standard basis vector \mathbf{e}_1 for \mathbf{C}^{16}). Notice that the Fourier transform of this signal has constant real part and zero imaginary part.

Now click on **Origin Centered**, which moves the time-zero position to the middle of the axis. Change the **Editing** box to **shift** and use the mouse to move the lollipop to the right, one position at a time.

(i) What symmetries around zero (the midpoint) do the graphs of **Real $\mathbf{F}(\mathbf{k})$** and **Imaginary $\mathbf{F}(\mathbf{k})$** have?

Notice that the **Real** and **Imaginary** parts of $\mathbf{F}(\mathbf{k})$ look like cosine and sine waves when the lollipop is near the origin (the open circle), but they seem much more erratic as you move the lollipop to the right end of the **Real x** axis. This is due to *undersampling*. Change the **Length** to 32 (which doubles the sampling rate) and move the lollipop back to the origin. Now move the lollipop to the right and left of the origin a few positions and observe the sampled cosine/sine wave pattern.

Now click the **Zero All** button, remove the check from the **Origin Centered** box, set the **Length** back to 16, and change the editing button back to **draw**. Using the mouse, create a signal $\mathbf{f}(\mathbf{x})$ corresponding to the vector

$$\mathbf{u} = \mathbf{e}_2 - \mathbf{e}_4 + \frac{1}{2}\mathbf{e}_8 \in \mathbf{R}^{16}$$

by raising or lowering the dots in the appropriate positions (the open dot on the left is position 1 in this indexing, so you should make a lollipop of maximum height above the second dot for the coefficient of \mathbf{e}_2 , and so on). Save and print the graph. Label the graph (by hand) **Fig. 1: Signal of Length 16**. Then change the **Length** to 32 (doubling the sampling rate) and redraw the signal \mathbf{u} (now \mathbf{u} is viewed as a vector in \mathbf{R}^{32}). Notice how the higher sampling rate shows the waveform pattern more clearly. Save and print the graph. Label the graph (by hand) **Fig. 2: Signal of Length 32**.

Question 3. Diagonalization of Circulant Matrices

For this question, read section 3 of the supplementary 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 9 of the notes).

Generate a random integer vector $\mathbf{v} \in \mathbf{R}^4$ by `v = rvect(4)`. Calculate $S*\mathbf{v}$ to see that S shifts the components of \mathbf{v} cyclically. Check by MATLAB that $(1/4)*\mathbf{F4}'*S*\mathbf{F4}$ is a diagonal matrix (where $\mathbf{F4}$ 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 \mathbf{v} from part (a) and powers of the matrix \mathbf{S} to construct a 4×4 *circulant matrix* \mathbf{C} whose first column is \mathbf{v} (see Theorem 3.2 in the Notes). Check by MATLAB that $(1/4)*\mathbf{F}_4' * \mathbf{C} * \mathbf{F}_4$ is a diagonal matrix.

(i) What are the eigenvalues of \mathbf{C} ? Why?

(ii) What are the eigenvectors of \mathbf{C} ? Why?

Check your answers by using MATLAB to calculate $[\mathbf{U}, \mathbf{D}] = \text{eig}(\mathbf{C})$.

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}_{even} and \mathbf{y}_{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 $\mathbf{y} = \text{rvect}(8)$ and the 8×8 downsampling matrix $\mathbf{P} = \text{downsamp}(4)$. Calculate $\mathbf{P} * \mathbf{y}$ and check that the entries in positions 1, 3, 5, and 7 of \mathbf{y} are the first four entries in $\mathbf{P} * \mathbf{y}$, and the entries in positions 2, 4, 6, and 8 of \mathbf{y} are the last four entries in $\mathbf{P} * \mathbf{y}$.

The block form in the FFT uses a diagonal matrix D_n (see equation (33) 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^(j-1) ;
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 Fourier matrix F_4 from the Fourier matrix F_2 , and then the Fourier matrix F_8 from F_4 . Type

```
P4 = downsamp(2); D2 = fftdiag(2); F2 = fft(eye(2));
F4 = [F2, D2*F2; F2, -D2*F2]*P4
```

Compare F_4 with the Fourier matrix `fft(eye(4))`. Are they the same?

Now repeat the process to generate the 8×8 Fourier matrix. First generate $\mathbf{P}_8 = \text{downsamp}(4)$ and $\mathbf{D}_4 = \text{fftdiag}(4)$. Use these and the matrix \mathbf{F}_4 you just generated to obtain \mathbf{F}_8 (see equation (33) in the Notes). Compare your \mathbf{F}_8 with the MATLAB-generated Fourier matrix by calculating `norm(F8 - fft(eye(8)))`. This should be of size 10^{-15} , thus zero to the limits of machine computation.

(b) **Speed Comparison:** Generate the $2^{10} \times 2^{10}$ Fourier matrix and a random vector $\mathbf{x} \in \mathbf{R}^{1024}$ (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(1024)); x = rand(1024,1);
```

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; toc
tic, fft(x); toc
```

It is quite likely that you will get zero for the elapsed time for the `fft` calculation. For the theoretical speed advantage of the FFT over the direct matrix multiplication when $n = 2^{10}$, see Section 4 of the Notes.

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 $A = \text{gallery}(3)$ from the MATLAB files (this matrix has eigenvalues 1, 2, and 3). Set $n = \text{size}(A,1)$, $I = \text{eye}(n,n)$ 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 block matrices that the diagonal entries of the final 2×2 upper-triangular matrix are eigenvalues of the original matrix A .

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.