Numerical Mathematics: Homework 6

Due on T.B.D at 24:00pm

Problem 1

Let U denote a unitary matrix. By writing $U = A + iB$ show that the matrix:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix}$$

is an orthogonal matrix.

Solution. By definition of being unitary, we have:

$$I = U^*U = (A+iB)^*(A+iB) = (A^T - iB^T)(A+iB) = (A^TA + B^TB) + i(A^TB - B^TA)$$

Hence:

$$A^TA + B^TB = I, \quad A^TB - B^TA = 0.$$

and:

$$\begin{bmatrix} A^T & B^T \\ -B^T & A^T \end{bmatrix}\begin{bmatrix} A & -B \\ B & A \end{bmatrix} = \begin{bmatrix} A^TA - B^TB & -(A^TB - B^TA) \\ A^TB - B^TA & A^TA - B^TB \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

□

Problem 2

Prove that the DFT matrix F_n, as defined in the lecture notes, i.e.

$$F_n = \frac{1}{\sqrt{n}}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix}, \qquad \omega_n = e^{-\frac{2\pi i}{n}}$$

is indeed an unitary matrix. *Hint!* The geometric series:

$$\sum_{k=0}^{n-1} r^k = \frac{1-r^n}{1-r}, \quad r \neq 1$$

is your friend here!

Solution. For $k, s = 0, 1, \dots, n-1$ we must show that

$$\frac{1}{n}\sum_{l=0}^{n-1} \bar{\omega}_n^{kl}\omega_n^{sl} = \begin{cases} 1 & k = s \\ 0 & k \neq s \end{cases}$$

But observe that:

$$\frac{1}{n}\sum_{l=0}^{n-1} \bar{\omega}_n^{kl}\omega_n^{sl} = \frac{1}{n}\sum_{l=0}^{n-1} \omega_n^{(s-k)l}$$

If $s = k$, we have:

$$\frac{1}{n}\sum_{l=0}^{n-1} \omega_n^{(s-k)l} = \frac{1}{n}\sum_{l=0}^{n-1} 1 = 1$$

If $s \neq k$, we have:

$$\frac{1}{n}\sum_{l=0}^{n-1} \omega_n^{(s-k)l} = \frac{1}{n}\sum_{l=0}^{n-1} \left[\omega_n^{(s-k)}\right]^l = \frac{1}{n}\frac{\omega_n^{(s-k)n} - 1}{1 - \omega_n^{(s-k)}} = 0.$$

□

Problem 3

Let $f(x)$ be a real-valued continuous, periodic function on $\mathbb{R}$. For simplicity, let the period be equal to one. Suppose we would like to approximate this function by a trigonometric function of the kind:

$$f_m(x) = a_0 + \sum_{k=1}^{m} a_k \cos(2\pi kx) + b_k \sin(2\pi kx).$$

As you already know, there are several ways this can be done. One way is to do interpolation. There are in total $2m+1$ independent variables to be determined, so we need also $2m+1$ interpolation points. Let us break up $[0,1)$ into equal sized segments and choose $x_k = \frac{k}{2m+1}$ for $k = 0, 1, 2 \ldots, m$ as interpolation point with $f(x_k)$ as interpolation values. This would generate a $2m+1$ by $2m+1$ linear system which needs to be solved. Inverting a linear system the naive way would require $\mathcal{O}((2m+1)^3)$.

(a) Explain how the coefficients a_0, a_k, b_k can be obtained from applying a Discrete Fourier Transform in $\mathcal{O}((2m+1)\log(2m+1))$. In the explanation, please describe each step of the algorithm (but you don't need to explain the details of the FFT algorithm). *Hint:* use the fact that $\cos(2\pi x) = \frac{1}{2}(e^{2\pi ikx} + e^{-2\pi ikx})$ and $\sin(2\pi x) = \frac{1}{2i}(e^{2\pi ikx} - e^{-2\pi ikx})$ to rewrite $f_m(x)$ in terms of complex exponentials. Then set-up the linear system which needs to be solved.

Another way of approximating $f(x)$ is through least-squares. That is, for example through minimizing:

$$E = \int_0^1 w(x) \left(f(x) - a_0 + \sum_{k=1}^{m} a_k \cos(2\pi kx) + b_k \sin(2\pi kx) \right)^2 dx$$

where the weighting function is chosen to be $w(x) = 1$.

(b) Show that solving the above least-squares problem yields the Fourier series, where:

$$a_0 = \int_0^1 f(x)dx, \quad a_k = 2\int_0^1 f(x)\cos(2\pi kx)dx, \quad b_k = 2\int_0^1 f(x)\sin(2\pi kx)dx$$

for $k = 1, 2, \ldots, m$.

Solution. a) Observe that:

$$\begin{aligned}
f_m(x) &= a_0 + \sum_{k=1}^{m} a_k \cos(2\pi kx) + b_k \sin(2\pi kx) \\
&= a_0 + \sum_{k=1}^{m} a_k \frac{e^{2\pi ikx} + b_k e^{-2\pi ikx}}{2} + b_k \frac{e^{2\pi ikx} - e^{-2\pi ikx}}{2i} \\
&= a_0 + \sum_{k=1}^{m} a_k \frac{e^{2\pi ikx} + e^{-2\pi ikx}}{2} + b_k \frac{e^{2\pi ikx} - e^{-2\pi ikx}}{2i} \\
&= \frac{1}{\sqrt{2m+1}}(\sqrt{2m+1}a_0) + \sum_{k=1}^{m} \left((\tfrac{1}{2}\sqrt{2m+1})(a_k - b_k i)\right) \frac{1}{\sqrt{2m+1}} e^{2\pi ikx} \\
&\quad + \left((\tfrac{1}{2}\sqrt{2m+1})(a_k + b_k i)\right) \frac{1}{\sqrt{2m+1}} e^{-2\pi ikx}
\end{aligned}$$

Define:

$$c_0 = \sqrt{2m+1}a_0, \quad c_k = (\tfrac{1}{2}\sqrt{2m+1})(a_k - b_k i), \quad c_{-k} = (\tfrac{1}{2}\sqrt{2m+1})(a_k + b_k i) \quad k = 1, \ldots, m.$$

To yield:

$$f_m(x) = \frac{1}{\sqrt{2m+1}} \sum_{k=-m}^{m} c_k e^{2\pi i k x}.$$

The linear system which needs to be solved is:

$$\begin{bmatrix} f(0) \\ f\left(\frac{1}{2m+1}\right) \\ f\left(\frac{2}{2m+1}\right) \\ \vdots \\ f\left(\frac{2m}{2m+1}\right) \end{bmatrix} = \frac{1}{\sqrt{2m+1}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i \frac{1}{2m+1}} & e^{2\pi i 2 \frac{1}{2m+1}} & \dots & e^{2\pi i m \frac{1}{2m+1}} & e^{-2\pi i m \frac{1}{2m+1}} & e^{-2\pi i (m-1) \frac{1}{2m+1}} & \dots & e^{-2\pi i \frac{1}{2m+1}} \\ 1 & e^{2\pi i \frac{2}{2m+1}} & e^{2\pi i 2 \frac{2}{2m+1}} & \dots & e^{2\pi i m \frac{2}{2m+1}} & e^{-2\pi i m \frac{2}{2m+1}} & e^{-2\pi i (m-1) \frac{2}{2m+1}} & \dots & e^{-2\pi i \frac{2}{2m+1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi i \frac{2m}{2m+1}} & e^{2\pi i 2 \frac{2m}{2m+1}} & \dots & e^{2\pi i m \frac{2m}{2m+1}} & e^{-2\pi i m \frac{2m}{2m+1}} & e^{-2\pi i (m-1) \frac{2m}{2m+1}} & \dots & e^{-2\pi i \frac{2m}{2m+1}} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \\ c_{-m} \\ c_{-m+1} \\ \vdots \\ c_{-1} \end{bmatrix}$$

Notice the selected ordering of the columns used. The last m columns can be rewritten as:

$$\begin{bmatrix} f(0) \\ f\left(\frac{1}{2m+1}\right) \\ f\left(\frac{2}{2m+1}\right) \\ \vdots \\ f\left(\frac{2m}{2m+1}\right) \end{bmatrix} = \frac{1}{\sqrt{2m+1}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i \frac{1}{2m+1}} & e^{2\pi i 2 \frac{1}{2m+1}} & \dots & e^{2\pi i m \frac{1}{2m+1}} & e^{2\pi i (m+1) \frac{1}{2m+1}} & e^{2\pi i (m+2) \frac{1}{2m+1}} & \dots & e^{2\pi i (2m) \frac{1}{2m+1}} \\ 1 & e^{2\pi i \frac{2}{2m+1}} & e^{2\pi i 2 \frac{2}{2m+1}} & \dots & e^{2\pi i m \frac{2}{2m+1}} & e^{2\pi i (m+1) \frac{2}{2m+1}} & e^{2\pi i (m+2) \frac{2}{2m+1}} & \dots & e^{2\pi i (2m) \frac{2}{2m+1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi i \frac{2m}{2m+1}} & e^{2\pi i 2 \frac{2m}{2m+1}} & \dots & e^{2\pi i m \frac{2m}{2m+1}} & e^{2\pi i (m+1) \frac{2m}{2m+1}} & e^{2\pi i (m+1) \frac{2m}{2m+1}} & \dots & e^{2\pi i (2m) \frac{2m}{2m+1}} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \\ c_{-m} \\ c_{-m+1} \\ \vdots \\ c_{-1} \end{bmatrix}$$

Which is exactly the inverse DFT matrix! The steps to find a_0, a_k, b_k is now really straightforard. Apply the DFT to $\begin{bmatrix} f(0) & f\left(\frac{1}{2m+1}\right) & f\left(\frac{2}{2m+1}\right) & \cdots & f\left(\frac{2m}{2m+1}\right) \end{bmatrix}^T$ to find the coefficients c_k. Then:

$$a_0 = \frac{1}{\sqrt{2m+1}} c_0, \quad a_k = \frac{2}{\sqrt{2m+1}} \mathrm{Re}\left\{c_k\right\}, \quad b_k = \frac{2}{\sqrt{2m+1}} \mathrm{Im}\left\{c_k\right\}.$$

b) A careful evaluation of the normal equations should generate:

$$S = \begin{bmatrix} 1 & & & \\ & \frac{1}{2} & & \\ & & \ddots & \\ & & & \frac{1}{2} \end{bmatrix}$$

which should give the desired result.

□

Problem 4

Write a function called `[X] = fft_radix2(x)` which implements the base-2 Fast fourier Transform algorithm to compute the DFT of vector $x \in \mathbb{C}^n$. It is highly adviced that you use recursive programming techniques to keep your life is easy.

Proof. Shown below is the code.

```
function [ X ] = fft_radix2( x )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

X = (1/sqrt(length(x))) * fft_unnormalized(x);



function [X] = fft_unnormalized(x)

if length(x) == 1
      X = x;
else

    if rem(length(x),2) ≠ 0
       disp('Warning: length of vector is not a power of 2.')
    end

   yt = fft_unnormalized( x(1:2:length(x)) );
   yb = fft_unnormalized( x(2:2:length(x)) );

   d = transpose( exp( -2*pi*1i* (0:1:(length(yt)-1)) / length(x) ) );

   z = d .* yb;

   X = [yt+z;
        yt-z];

end
```

$\square$

Problem 5

Let Z_n be a n by n (permutation) matrix defined by:

$$Z_n = \begin{bmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \end{bmatrix}$$

(a) Explain what Z_n does to the entries of a vector under matrix vector multiplication.

(b) Show that the columns of the DFT matrix are the eigenvectors of Z_n. What are the corresponding eigenvalues? Write the matrix in the form $AF_n = F_n\Lambda$, where Λ is a diagonal matrix with eigenvalues on the main diagonal.

The following matrix:

$$C(a_0, a_2, \ldots a_{n-1}) = \begin{bmatrix} a_0 & a_{n-1} & \cdots & a_3 & a_2 \\ a_1 & a_0 & a_{n-1} & & a_3 \\ \vdots & a_1 & a_0 & \ddots & \vdots \\ a_{n-2} & & \ddots & \ddots & a_{n-1} \\ a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \end{bmatrix}$$

is called a Circulant matrix. The name is derived from the fact that each consecutive column is a cyclic permutation of the previous column. Notice also that the entries of the matrix are constant along the diagonal, i.e. it is a special case of a Toeplitz matrix.

(c) Show that the columns of the DFT matrix are also the eigenvectors of $C(a_0, a_2, \dots a_{n-1})$. Use the decomposition:

$$C(a_0, a_2, \dots a_{n-1}) = \sum_{k=0}^{n-1} a_k Z_n^k$$

Of course, you must also explain why the above decomposition is correct.

(d) Confirm that:

$$C(a_0, a_2, \dots a_{n-1}) = F_n D F_n^*, \qquad D = \mathrm{diag}(\sqrt{n} F_n^* a)$$

where $a = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \end{bmatrix}^T$.

Solution. a) It circularly permutes/rotates the entries of a vector.

b) Let $F_n = \begin{bmatrix} f_{n,0} & f_{n,1} & \cdots & f_{n,n-1} \end{bmatrix}$. Then it is easy to verify that:

$$Z_n f_{n,k} = e^{\frac{2\pi i}{n} k} f_{n,k} \qquad k = 0, 1, \dots, n-1.$$

c) This fact follows from:

$$C(a_0, a_2, \dots a_{n-1}) = \sum_{k=0}^{n-1} a_k Z_n^k = \sum_{k=0}^{n-1} a_k (F_n \Lambda F_n^*)^k = F_n (\sum_{k=0}^{n-1} a_k \Lambda^k) F_n^*$$

d) A closer look at:

$$D = \sum_{k=0}^{n-1} a_k \Lambda^k$$

should reveal this fact. □

Problem 6

Let $a, b : \mathbb{Z}_n \mapsto \mathbb{C}$ be two signals on the integers modulo n. That is,

$$a[k] = a_k, \quad \text{for } k = 0, 1, \dots, n-1 \qquad \text{and} \qquad a[k \mod n] = a[k].$$

and similarly for $b[k]$. Convolution $c = a * b$ is defined as the operation:

$$c[k] = \sum_{l=0}^{n-1} a[k-l] b[l]$$

where $c : \mathbb{Z}_n \mapsto \mathbb{C}$ is another signal defined on the integers modulo n.

(a) Show that the operation of convolution is commutative, i.e. $c = a * b = b * a$.

(b) Show that convolution of a n-periodic signal can be expressed as a multiplication with a circulant matrix. Given the commutativity of the operation, there are two ways to do this. Describe both.

Since convolution can be written as a matrix vector product, the naive approach would generally require $\mathcal{O}(n^2)$ operations to convolve two n-periodic signals.

(c) Given the results derived in the previous problem and your knowledge about FFT algorithms. Describe a $\mathcal{O}(n \log n)$ method that does the job more efficiently. Implement this algorithm. Call this function `[c] = convolve_periodic(a,b)`. For convenience, please use the built-in FFT code. Keep in mind that this version of the FFT is normalized differently.

Solution. a) A simple change of variables should do the trick.

b) Let $a = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \end{bmatrix}^T$ and $b = \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix}^T$. Then $c = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}^T$ is defined as:

$$c = C(a_0, a_2, \dots a_{n-1})b = C(b_0, b_2, \dots b_{n-1})a$$

c) This can simply done by evaluating a series of three FFTs. Let:

$$\hat{a} = \sqrt{n} F_n^* a, \quad \hat{b} = F_n^* b$$

Then define $\hat{c}$ as the product of $\hat{a}$ and $\hat{b}$, i.e. $c[k] = a[k]b[k]$. Then:

$$c = F_n \hat{c}.$$

Shown below is the code.

```
function [ c ] = convolve_periodic(a,b)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

n = length(a);
c = (1/sqrt(n))* fft(   (n* ifft(a) )   .*   ( sqrt(n)* ifft(b))   );

end
```

□

Problem 7

There are many applications to convolution, notably in signal processing and digital image processing. However, convolution (and in particular the efficient numerical implementation of convolution) has even applications in areas as fundamental as multiplying two large integers. Note that a n-digit (positive) integer can be decomposed as

$$a = \sum_{k=0}^{n-1} a_k \cdot 10^k, \qquad b = \sum_{k=0}^{n-1} b_k \cdot 10^k$$

We may write:

$$c = ab = \sum_{k-0}^{2n-1} c_k \cdot 10^k = \left(\sum_{k=0}^{n-1} a_k \cdot 10^k \right) \left(\sum_{k=0}^{n-1} b_k \cdot 10^k \right)$$

(a) Write the above as a matrix vector product. What is special about the entries of the matrix?

(b) Explain how one can be embed the matrix-vector mulplication of question (a) into a larger matrix-vector product involving a Circulant matrices.

(c) Discuss the savings that are made by using the FFT algorithm. What concerns should be taken into regard concerning floating point arithmetic?