Math 566 **Homework 2** **Due Feb. 17, 2015**
Grady Wright

1. *Trigonometric interpolation.* For this problem you will be using the MATLAB function for trigonometric interpolation below. Please note that we have sacraficed efficiency for readability with the implementation of trigonometric interpolation given below. As opposed to copying this function, you may simply download it from the course website.

```
% Computes the trigonometric interpolant based on the equally spaced nodes
% in x and function values in y, and evaluates the interpolant at all
% points in xi.
% Note:  This is not the most optimal implementation, especially the
%        evaluation.
function p = trigInterp(x,y,xi)
N = length(x);
h = x(2)-x(1);
% Store the size of xi
[mxi,nxi] = size(xi);
% Force x, y, and xi are row vectors.
x = x(:).'; y = y(:).'; xi = xi(:).';
% Compute the interpolation coefficients.  Note, the indicies are arranged as
% 0,1,...,(N-1)/2,-(N-1)/2,...,-2,-1 for N odd, and
% 0,1,...,N/2-1,N/2,-N/2+1,...,-2,-1 for N even.
c = (1/N)*fft(y);
% Change variables to t.
t = (xi-x(1))/h;
% If N is odd, no modification is needed.
if ( mod(N,2) == 1 )
   % The wave numbers k arranged according to the output from MATLAB's FFT.
   k = [0:((N-1)/2) -(N-1)/2:-1].';
   % Compute the  interpolant as a row-column vector product instead
   % of the actually summing the result.
   p = c*(exp(2*pi*1i*k*t./N));
else
   % The wave numbers k (without the N/2 term) arranged according to the
   % output from MATLAB's FFT.
   k = [0:(N/2)-1 -N/2+1:-1]';
   % Compute the interpolant with the N/2 term (using a row-column vector
   % product) then add that term.
   p = c([1:N/2 N/2+2:N])*(exp(2*pi*1i*k*t./N)) + c(N/2+1)*cos(pi*t);
end
% The real part of p.  Note the p should only contain real values. However,
% since c contained complex values, MATLAB assumes p should also contain
% complex values and thus appends a 0+i onto the final result.
p = real(p);
% Make p the same shape as xi.
p = reshape(p,[mxi nxi]);
end
```

(a) Use `trigInterp` function to plot the trigonometric interpolant to the periodic square-wave function

$$f(x) = \begin{cases} 1 & |x| \le 1/2 \\ 0 & -1 \le x < -1/2 \text{ or } 1/2 < x < 1 \end{cases}$$

over $[-1,1]$. For the interpolation points (i.e. the nodes), use the values $x_j = -1 + j(0.1)$ for $j = 0, \ldots, 19$. The plot that you should see is an example of the Gibbs' phenomenon. It arises whenever a discontinuous function is interpolated (or expanded) with smooth functions. In MATLAB , $f(x)$, for $x \in [-1, 1)$, can be implemented with the code:

```
x = -1 + (0:19)*0.1;
y = abs(x) <= 0.5;
```

Note: What we now call Gibbs' phenomenon was first noted by Wilbraham (1848). Unaware of this, Michelson and Stratten (1898) found traces of these overshoots in the output plots form a mechanical Fourier analyzer they had constructed. This observation by Michelson (who is probably best know for his ether experiment with Morely) prompted him to write a letter to *Nature* inquiring about the convergence properties of a Fourier Series for a discontinuous function. In reply, J. Gibbs (an eminent chemist) provided first a flawed and then a satisfactory answer. (Extracted from *A Practical Guide to Pseudospectral Methods*, Fornberg (1995))

(b) Use the `trigInterp` function to compute the maximum magnitude of the error in the trigonometric interpolant to the function $f(x) = \exp(\sin(4\pi x))$ for equally spaced node sets over $[-1, 1)$ of sizes $N = 10, 30, 50, \ldots, 150$. Plot the error verses the number of points N using a `loglog` plot and a `semilogy` plot. What type of accuracy does the interpolant exhibit for this function (you should be able to tell based on which plot shows the error decreasing linearly).

(c) If $f(x) = \sin(\pi x)$, what should the error in the trigonometric interpolant to this function be if three or more interpolation points are used? Justify your answer (this should require one sentence).

2. The discrete convolution of two vectors $\mathbf{a} = \{a_j\}_{j=0}^{N-1}$ and $\mathbf{x} = \{x_j\}_{j=0}^{N-1}$ is defined as a new vector $\mathbf{b}$ with entries

$$b_m = \sum_{j=0}^{N-1} a_{m-j} x_j, \quad m = 0, \ldots, N-1, \tag{1}$$

or in matrix-vector form as

$$\underbrace{\begin{bmatrix} a_0 & a_{N-1} & a_{N-2} & \cdots & a_1 \\ a_1 & a_0 & a_{N-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & a_0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{N-1} \end{bmatrix}}_{\mathbf{b}} \tag{2}$$

The structure exhibited by A gets a special name in linear algebra: *circulant*. This can be viewed as a discretization of the continuous circular convolution (up to scaling) of two T-periodic functions $a(t)$ and $x(t)$ on the interval $[t_0, t_0 + T]$:

$$(a * x)(t) = \int_{t_0}^{t_0+T} a(t-\tau)x(\tau)d\tau,$$

where $a_{m-j} \approx a(t_m - t_j)$, $x_j \approx x(t_j)$, and $t_\ell = t_0 + (T/N)\ell$. Circular convolutions and their discrete versions play a fundamental role in signal processing (your cell phones would not work without them!).

One of the most important results of the discrete Fourier transform (DFT) is that it can be used relate the discrete convolution of two vectors to the DFT of the individual vectors in a beautiful way. The result is known as the discrete (or circular) convolution theorem and says the following. Let $\hat{\mathbf{a}}$ and $\hat{\mathbf{x}}$ denote the DFTs of $\mathbf{a}$ and $\mathbf{x}$ in (2), respectively. Then the DFT of the discrete convolution of $\mathbf{a}$ and $\mathbf{x}$ is given by

$$\hat{b}_m = N\hat{a}_m\hat{x}_m, \quad m = 0, \ldots, N-1, \tag{3}$$

or in matrix-vector form:

$$\underbrace{\begin{bmatrix} \hat{a}_0 & 0 & 0 & \cdots & 0 \\ 0 & \hat{a}_1 & 0 & \cdots & 0 \\ 0 & 0 & \hat{a}_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \hat{a}_{N-1} \end{bmatrix}}_{\hat{A} := \text{diag}(\hat{\mathbf{a}})} \underbrace{\begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_{N-1} \end{bmatrix}}_{\hat{\mathbf{x}}} = \frac{1}{N} \underbrace{\begin{bmatrix} \hat{b}_1 \\ \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_{N-1} \end{bmatrix}}_{\hat{\mathbf{b}}} \tag{4}$$

This theorem means that we can recover the result of the discrete convolution of **a** and **x** by the following procedure:

i Compute the DFTs of **a** and **x**.

ii Obtain $\hat{\mathbf{b}}$ according to (3).

iii Compute the inverse DFT of $\hat{\mathbf{b}}$ to obtain **b** in (1).

By using the FFT to compute the DFT, this procedure requires $O(N \log N)$ operations instead of $O(N^2)$ as formula (2) might suggest.

Problem: Verify that the discrete convolution theorem is correct.

Hint:

- Let U be the DFT matrix ($U_{j,k} = \omega_N^{jk}$, with $\omega_N = e^{2\pi i/N}$) and $U^{-1} = \frac{1}{N}U^*$ be the inverse DFT matrix. To verify the theorem, it is sufficient to show

$$\frac{1}{N}U^*AU = N\hat{A},$$

where A is defined in (2). **Answer why this sufficient?**. It may be useful to first show that entry (m, n) in the matrix product AU is given as

$$(AU)_{m,n} = \sum_{k=0}^{N-1} a_k \omega_N^{(m-k)n}, \; m, n = 0, \ldots, N-1.$$

Here we are assuming the indexing of the rows and columns of the product start at $n = m = 0$.

- As an alternative to a fully general proof, you may analytically verify the theorem for the $N = 4$ case in such a way that it becomes obvious how the same procedure would carry through to any larger size as well.

3. *Fast Toeplitz matrix-vector multiplication.* Toeplitz matrices occur surprising often in many areas of pure and applied mathematics (e.g. time-series analysis, the numerical solution of certain partial differential equations, approximation of functions, combinatorics, algebra, etc.) A matrix is called Toeplitz if it has constant entries down its diagonals. For example

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & a_{-3} & \cdots & a_{-(N-1)} \\ a_1 & a_0 & a_{-1} & a_{-2} & \ddots & \vdots \\ a_2 & a_1 & a_0 & a_{-1} & \ddots & a_{-3} \\ a_3 & a_2 & a_1 & a_0 & \ddots & a_{-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & a_{-1} \\ a_{(N-1)} & \cdots & a_3 & a_2 & a_1 & a_0 \end{bmatrix}.$$

Note that A only depends on $a_0, a_1, \ldots, a_{N-1}, a_{-(N-1)}, a_{-(N-2)}, \ldots, a_{-1}, a_0$. Interestingly, the discrete convolution theorem and the FFT can be used to multiply a Toeplitz matrix by a vector of length N in $O(N \log N)$ operations. The idea is to embed the Toeplitz matrix into a *circulant* matrix, which can then be multiplied by a vector using the FFT and inverse FFT as discussed in problem 2.

The trick for the circulant embedding can be illustrated with the following example. Suppose we wish to compute the 3−by−3 Toeplitz matrix-vector product:

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} \\ a_1 & a_0 & a_{-1} \\ a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}. \tag{5}$$

This is equivalent to the 5−by−5 circulant matrix-vector product

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & a_2 & a_1 \\ a_1 & a_0 & a_{-1} & a_{-2} & a_2 \\ a_2 & a_1 & a_0 & a_{-1} & a_{-2} \\ a_{-2} & a_2 & a_1 & a_0 & a_{-1} \\ a_{-1} & a_{-2} & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ 0 \\ 0 \end{bmatrix}, \tag{6}$$

the right hand side vector $\mathbf{b}$ in (5) is given by the first three entries in the resulting vector from (6). Note that (6) is just a discrete convolution of the vector $\mathbf{a} = \begin{bmatrix} a_0 & a_1 & a_2 & a_{-2} & a_{-1} \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_0 & x_1 & x_2 & 0 & 0 \end{bmatrix}$.

(a) Write a MATLAB function, `function y=toepmult(A,x)`, for computing the Toeplitz matrix-vector product Ax using the circulant embedding and the FFT and inverse FFT.

(b) Use your function to compute matrix-vector products with the Toeplitz matrix

$$A = \begin{bmatrix} 1 & 1/2 & 1/3 & \cdots & 1/N \\ -1/2 & 1 & 1/2 & \ddots & \vdots \\ -1/3 & -1/2 & 1 & \ddots & 1/3 \\ \vdots & \ddots & \ddots & \ddots & 1/2 \\ -1/N & \cdots & -1/3 & -1/2 & 1 \end{bmatrix}$$

and various random vectors x. Report the max. norm difference in a nice table between your functions multiplication vs. straightforward matrix-vector multiplication in MATLAB for $N = 10, 100, 500, 1000, 5000$ and various random vectors x.

Note: The A matrix can be created very simply in MATLAB using the command: `A = toeplitz([1 -1./(2:N)],1./(1:N))`, where N has already been defined. A random vector can be generated using the command `x = rand(N,1)`. Regular matrix-vector multiplication in MATLAB can be done using the command `A*x`.

4. Consider the tridiagonal linear system of equations

$$\begin{bmatrix} b & a & 0 & \cdots & \cdots & \cdots & 0 \\ a & b & a & \ddots & & & \vdots \\ 0 & a & b & a & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & a & b & a \\ 0 & \cdots & \cdots & \cdots & 0 & a & b \end{bmatrix} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{bmatrix}}_{\mathbf{f}}, \tag{7}$$

and suppose $u_0 = u_N = f_0 = f_N = 0$, and $b \neq 0$. This system can be solved efficiently using the Crout (or Thomas) algorithm for linear systems. However, in this problem you will show how to solve the system using the Discrete Sine Transform (DST).

(a) Let $\hat{\mathbf{u}}$ and $\hat{\mathbf{f}}$ denote the DST of the vectors $\mathbf{u}$ and $\mathbf{f}$ in (7), i.e.

$$\hat{u}_k = \frac{1}{N}\sum_{j=1}^{N-1} u_j \sin\left(\frac{\pi j k}{N}\right),$$
$$\hat{f}_k = \frac{1}{N}\sum_{j=1}^{N-1} f_j \sin\left(\frac{\pi j k}{N}\right),$$

for $k = 1, \ldots, N-1$. Now, we can express the entries of $\mathbf{u}$ and $\mathbf{f}$ as

$$u_j = 2\sum_{k=1}^{N-1} \hat{u}_k \sin\left(\frac{\pi j k}{N}\right),$$
$$f_j = 2\sum_{k=1}^{N-1} \hat{f}_k \sin\left(\frac{\pi j k}{N}\right).$$

Substitute these expressions into the linear system (7) to show that row j of this systems simplifies to

$$\sum_{k=1}^{N-1} \hat{u}_k \left(2a\cos\left(\frac{\pi k}{N}\right) + b\right)\sin\left(\frac{\pi j k}{N}\right) = \sum_{k=1}^{N-1} \hat{f}_k \sin\left(\frac{\pi j k}{N}\right),$$

in the case of $j = 1$ and $j = N-1$ use the fact that $u_0 = u_N = 0$.

(b) Use the results from part (a) to show that the DST of the solution to (7) is given by

$$\hat{u}_k = \frac{\hat{f}_k}{2a\cos\left(\frac{\pi k}{N}\right) + b}, \quad k = 1, \ldots, N-1.$$

(c) Put parts (a) and (b) together to explain how to obtain the solution to (7).

(d) Download the MATLAB code `dst.m` on the course webpage for computing the DST (this code uses the FFT to do the computation and is thus 'fast', although not as fast as it could be). Use this code and the procedure outlined in (a)–(c) to solve the linear system (7) for $N = 100$, $a = 1$, $b = -2$, and $f_j = (\pi/N)^2 \tanh(4\sin(\pi j/N))$, for $j = 1, \ldots, N-1$.