Final Project
MA/CS 375, Spring 2002
Due 12:00 noon on Monday, May 6
Turn in to me at my office, Hum 432

TURN IN TESTS OF ALL FUNCTIONS YOU WRITE!
Usual rules apply; format and appearance is important!

The project is to implement and test a program for computing polynomial interpolants on Chebyshev nodes using the Fast Fourier Transform (FFT). The key to this is the representation of the interpolant, $P(x)$, in the Chebyshev basis:

$$P(x) = \sum_{j=0}^{n} a_j T_j(x), \tag{1}$$

where the jth Chebyshev polynomial is given by

$$T_j(x) = \cos(j \arccos x)\ , \quad \text{or} \quad T_j(\cos t) = \cos(jt). \tag{2}$$

Using trigonometric identitites, it can also be shown that:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x), \ j \geq 2. \tag{3}$$

(You may remember using this recursion relation in homework 1.)

(NOTE: **The discrete cosine transform is its own inverse!**
The following formulas can be useful for validating your results!

$$f_k = \sum_{m=0}^{M} F_m \cos\frac{\pi k m}{M},$$

$$F_n = \frac{2c_n}{M} \sum_{k=0}^{M} c_k f_k \cos\frac{\pi k n}{M}.$$

where $f_k = f(x_k)$ are the grid values of the function $f(x)$, F_n are the expansion coeficients and the quantities c_k are defined as $c_k = 1$ for $k = 1 : M-1$ while $c_0 = c_M = 1/2$.)

(i.) Argue that each member of the canonical basis for the polynomials, x^n, can be expressed in terms of the Chebyshev polynomials:

$$x^n = \sum_{j=0}^{n} d_j T_j(x). \tag{4}$$

You need not find the coefficients d_j explicitly, but simply show that they can be constructed. Note that $T_j(x)$ is a degree j polynomial which by the recursion relation (3) may be written in the form:

$$T_j(x) = 2^{j-1}x^j + \text{ lower order terms.} \tag{5}$$

Explain why this implies that any polynomial of degree n can be expressed in the form (1).

(ii.) Consider the polynomial interpolation problem using Chebyshev nodes and the Chebyshev expansion (1):

$$P(x_i) = \sum_{j=0}^{n} a_j T_j(x_i) = y_i, \quad i = 0, \ldots, n, \tag{6}$$

$$x_i = \cos\left(\frac{i}{n}\pi\right). \tag{7}$$

(Note these are a slightly different set of Chebyshev nodes than used in class which include the endpoints. They also lead to stable interpolation.) Making the change of variables:

$$x = \cos t, \tag{8}$$

show that the interpolation problem (6)-(7) is equivalent to the trigonometric interpolation problem:

$$a_0 + \sum_{j=1}^{n-1} a_j \cos jt_i + a_n \cos nt_i = y_i, \quad t_i = \frac{i}{n}\pi, \quad i = 0, \ldots, n. \tag{9}$$

By equivalent we mean that the same coefficients, a_j, solve each problem.

(iii.) Problem (9) differs from the standard interpolation problem of Section 2.4.4 by the absence of the $\sin jt$ terms. Consider the even extension of the data, adding nodes in $(\pi, 2\pi)$ by the prescription:

$$y_i = y_{2n-i}, \quad t_i = \frac{i}{n}\pi, \quad i = n+1, \ldots, 2n. \tag{10}$$

We now have 2π-periodic data. We can thus compute a full trigonometric interpolant, involving both $\cos jt$ and $\sin jt$, using the FFT-based function you wrote for problem 5.4.2. However, due to the fact that:

$$\cos(jt_i) = \cos(jt_{2n-i}), \quad \sin(jt_i) = -\sin(jt_{2n-i}), \quad i = n+1, \ldots, 2n, \tag{11}$$

the coefficients of the sine term vanish (in exact arithmetic). Therefore, the output of your FFT-based trigonometric interpolant, given the extended dataset (10), are the coefficents a_j in (9) and, hence, (6). Modify your function from 5.4.2 to solve (9) via FFTs applied to the extended dataset. Verify that the sine coefficients are negligibly small.

(iv.) Use the function written in (iii.) to solve the polynomial interpol ation problem on Chebyshev points, (6).

(v.) For the Chebyshev series to be useful, we must be able to evaluate it. Write a function which takes as input the coefficients, a_j, computed by the function written for part (iv) and a set of points, z, and returns:

$$\sum_{j=0}^{n} a_j T_j(z). \tag{12}$$

To do this efficiently, i.e. with $O(n)$ flops, use the recursion (3).

(vi.) Demonstrate the accuracy and efficiency of your code by carrying out at least the following experiments:

1. For n chosen so that FFTs of vectors of length 8, 16, 32, 64 and 128 are required, i.e. $n = 4, 8, 16, 32, 64$, verify the accuracy of your functions by interpolating random data and computing the norm of the relative residual of the interpolation equations. That is, evaluate the interpolant at the nodes and find the maximum error normalized by the maximum data value. Comment on the results.
2. For the same choices of n, find the number of flops required to compute the coefficients and verify that it grows like $n \ln n$. Also record the number of flops required to evaluate the interpolant at a single point and verify that it grows like n.
3. Use your functions to interpolate:

$$\cos(kx)e^{-2x^2}, \quad k = 1, 10, 30. \tag{13}$$

   Try the values of n above, and find the maximum error on 100 equally spaced nodes. What is the smallest value that produces an error less than .001? For each value of k plot the function along with representative interpolants. Don't bother to increase n once the error tolerance of .001 is attained.