Homework/Programming 3. (Due Sept. 20)

Math. 609-600

This assignment is a programming problem designed to illustrates spectral convergence of the trigonometric interpolant. The solution to the exercise will involve a number of functions in matlab or python which will be put together with a driver computing the approximation and measuring the error.

We will solve the interpolation problem: Find

$$f_{2N}(x) = \sum_{j=-N+1}^{N} c_j \psi_j(x) \tag{0.1}$$

such that

$$f_{2N}(x_j) = f(x_j), \qquad j = 0, 1, ..., 2N-1, \tag{0.2}$$

where $\psi_j(x) = e^{ijx}$ and $x_j = jh$ with $h = \pi/N$. These are equations (9) and (10) of Lecture 9 in the 609d lecture notes. You should read Section 9.1 of the 609d lecture notes.

We use an even number of points here so that we may set $N = 2^k$ in this exercise. In this case, the FFT computes the N-point DFT's in $O(Nk)$ operations where

$$DFT_{\pm}(c)(j) = \sum_{l=0}^{2N-1} c_l E(\pm lj), \qquad j = 0, ..., 2N-1,$$

and

$$E(m) = e^{\frac{2\pi i m}{2N}}.$$

We shall start arrays with index 0 in this discussion. This will mean that all indices in MATLAB implementations will need to be shifted by one. This detail is left as part of the exercise if you choose to program in MATLAB.

Problem 1. Given $c_{-N+1}, c_{-N+2}, \ldots, c_N$, define a vector $d_0, d_1, \ldots, d_{2N-1}$ so that

$$g_m := \sum_{j=-N+1}^{N} c_j \exp(imx_j) = (DFT_+ d)(m).$$

Write a function with interface:

```
function [g]=TRIGEVAL(C,N);
```

This function should call the the appropriate FFT function in your language to implement DFT_+ above. I believe that the fft function in MATLAB computes DFT_- while the $ifft$ function computes its inverse (but you should check this to make sure).

Note that the above routine when applied to c computes

$$(f_{2N}(x_0), f_{2N}(x_1), \ldots, f_{2N}(x_{2N-1}))$$

where f_{2N} is defined by (0.1) above.

Problem 2. Write a routine which computes the coefficients

$$c = (c_{-N+1}, c_{-N+2}, \ldots, c_N)$$

of the interpolating polynomial f_{2n} (defined on the right hand side of (0.1)) satisfying

$$f_{2N}(x_i) = TF_i, \qquad \text{for } i = 0, 1, 2, \ldots 2N$$

by first computing

$$\tilde{d} = \frac{1}{2N} DFT_-(\tilde{f}).$$

Note that $\tilde{c}$ then is then the inverse of the shift used in Problem 1 applied to $\tilde{d}$.

The above routine should have interface:

```
function [C]=TRIGCOEF(TF,N);
```

The above routines are inverses of each other. You should check this by computing

$$D = TRIGGEN(TRIGEVAL(TC, N), N)$$

which should result in $D = TC$ for any vector TC (do this on a number of vectors, changing TC and N until you are convinced).

We are now ready to experiment with these codes. However, we have to be a bit careful here. To get an indicator of how well the interpolant is converging we need to look at points that are not interpolating nodes, i.e., outside the set $x_0, x_1, \ldots, x_{2N-1}$. Obviously, if you have coded the above functions correctly, the error at the interpolation points will be zero.

We will test the convergence by computing the error on a grid which has twice as many grid points, namely,

$$xp := (x_0, (x_0 + x_1)/2, x_1, (x_1 + x_2)/2, x_2, \ldots (x_{2N-1} + x_{2N})). \tag{2.1}$$

Note that these are just the nodes associated the grid with $4N$ points. We can use the TRIGEVAL routine to compute f_{2N} on the grid (2.1) by padding (by zeros) the solution coefficients c (for f_{2N}). We set

$$cp := (0, 0, \ldots, 0, c_{-N+1}, \ldots, c_N, 0, 0, \ldots, 0).$$

Here there are N zeros before c_{-N+1} and N zeros after c_N. Now if we use the right hand side of (0.1) with $4N$ and cp we obtain the identical trigonometric function, namely, f_{2N} so that applying

F4=TRIGEVAL(cp,2N)

produces f_{2N} but evaluated at the grid points in xp.

For the last problem, we consider four functions, namely

$$\begin{aligned} f_1(x) &= x^2(2\pi - x)^2, \\ f_2(x) &= x(2\pi - x), \\ f_3(x) &= 1 + x + x^2, \\ f_4(x) &= \exp(-(1/x + 1/(2\pi - x))). \end{aligned} \tag{2.2}$$

I did not mention (when we did 609d lecture 8) that Theorem 8.4 extends to non-integer values of n. The functions above are infinitely smooth so the only thing that limits n in Remark 8.2 is the boundary jump conditions. For the above problems:

(1) f_1, f_1' and f_1'' are periodic but f_1''' is not so that f_1 is in $\dot{H}^\alpha$ for $\alpha < 7/2$.
(2) f_2 and f_2' is periodic but f_2'' is not so that f_2 is in $\dot{H}^\alpha$ for $\alpha < 5/2$.
(3) f_3 is not periodic so that f_3 is in $\dot{H}^\alpha$ for $\alpha < 1/2$.
(4) f_4 and all of its derivatives are periodic so f_4 is in $\dot{H}^\alpha$ for any $\alpha > 0$.

The result of Theorem 8.4 can be further refined and one expects, for example,

$$\|f - f_{2N}\| \leq C \ln(N) N^{-3/2-\ell}$$

when $f^{(j)}$ is periodic for $j = 0, \ldots, \ell$ with $\ell = -1$ when f is not periodic.

Problem 3. For each $N = 2^k$ for $k = 3, 4, \ldots, 10$ and each of the 4 problems in (2.2) ($m = 1, 2, 3, 4$): (you should create a driver routine which loops over the 5 steps below)

(a) Set $TF_j = f_m((j-1)h)$ for $j = 0, \ldots, 2N-1$ with $h = \pi/N$.
(b) Compute the coefficients c of the interpolant using TRIGGEN and TF.
(c) Set up padded vector cp.
(d) Compute the interpolant F4 at the padded nodes xp using TRIGEVAL and cp.
(e) Compute and report the estimate of the error:

$$e_k^m = \left(\frac{1}{4N} \sum_{i=0}^{4N-1} |f_m(xp_i) - F4_i|^2 \right)^{1/2}.$$

(The above is a quadrature approximation to the error in the norm in $L^2(0, 2\pi)$).

For each function, generate two columns of output of the form

```
  error      Rate

e^m_{k=3}    *
e^m_{k=4} r^m_{k=4}
e^m_{k=5} r^m_{k=5}
```

$\vdots$	$\vdots$

Here r_k^m is a convergence rate estimator and is defined by

$$r_k^m = log(e_{k-1}^m / e_k^m)/log(2).$$

You need not compute a rate estimator for $k = 3$ hence the *. The rates you see should coincide with α in the regularity of f_m although there will be no obvious rate for the $m = 4$ function. Assemble all columns into one table and include the table with your submission.

Include the code that you developed for this assignment along with the table generated in the last problem.