CS4210 Assignment 4 Due: 10/17/14 (Fri) at 6pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the solutions you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Points may be deducted for poor style and reckless inefficiency.

Topics: Fast Trigonometric interpolation,numerical integration

1 Fast Trigonometric interpolation

In this problem you will use the FFT to compute vectors $a \in \mathbb{R}^{m+1}$ and $b \in \mathbb{R}^{m-1}$ so that if $y \in \mathbb{R}^{2m}$ is given, then

$$y_k \;=\; \frac{a_0}{2} + \sum_{j=1}^{m-1}\left(a_j\cos\left(\frac{kj\pi}{m}\right) + b_j\sin\left(\frac{kj\pi}{m}\right)\right) + \frac{(-1)^k}{2}a_m \qquad k = 0{:}2m-1 \tag{1}$$

The lecture codes for September 25 present a "slow" way for doing this. (The notation is slightly different and a_0 and a_m are different by a factor of two.)

Consideration of the case $m = 3$ shows the way. The starting point is to look at the 8x8 DFT matrix F_8. Recall that

$$[F_n]_{kj} \;=\; \omega_n^{kj} \qquad \omega_n = \cos\left(\frac{2\pi}{n}\right) - i\cos\left(\frac{2\pi}{n}\right).$$

Letting $\omega = \omega_8$ and using facts like $\omega_8^{15} = \omega_8^7 = \bar{\omega}_8 = \bar{\omega}^9$ we see that

$$F_8 \;=\; \left[\begin{array}{c|ccc|c|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & -1 & \bar{\omega}^3 & \bar{\omega}^2 & \bar{\omega} \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \bar{\omega}^6 & \bar{\omega}^4 & \bar{\omega}^2 \\ 1 & \omega^3 & \omega^6 & \omega^9 & -1 & \bar{\omega}^9 & \bar{\omega}^6 & \bar{\omega}^3 \\ 1 & \omega^4 & \omega^8 & \omega^{12} & 1 & \bar{\omega}^{12} & \bar{\omega}^8 & \bar{\omega}^4 \\ 1 & \omega^5 & \omega^{10} & \omega^{15} & -1 & \bar{\omega}^{15} & \bar{\omega}^{10} & \bar{\omega}^5 \\ 1 & \omega^6 & \omega^{12} & \omega^{18} & 1 & \bar{\omega}^{18} & \bar{\omega}^{12} & \bar{\omega}^6 \\ 1 & \omega^7 & \omega^{14} & \omega^{21} & -1 & \bar{\omega}^{21} & \bar{\omega}^{14} & \bar{\omega}^7 \end{array}\right]$$

Note that columns 7, 6, and 5 are the conjugate of columns 1, 2, and 3 respectively. (Proof: $\omega^{k(n-j)} = \omega^{nk}\omega^{-kj} = \bar{\omega}^{kj}$.) Using facts like $(\alpha + i\beta)\omega + (\alpha - i\beta)\bar{\omega} = 2\alpha\cdot\mathrm{Re}(\omega) + 2\beta\cdot\mathrm{Im}(\omega)$ it follows that

$$F_8\begin{bmatrix} a_0 \\ a_1+ib_1 \\ a_2+ib_2 \\ a_3+ib_3 \\ a_4 \\ a_3-ib_3 \\ a_2-ib_2 \\ a_1-ib_1 \end{bmatrix} \;=\; a_0\begin{bmatrix} 1\\1\\1\\1\\1\\1\\1\\1 \end{bmatrix} \;+\; 2\cdot\mathrm{Re}\left(F_8(:,1{:}3)\right)\begin{bmatrix} a_1\\a_2\\a_3 \end{bmatrix} \;+\; a_4\begin{bmatrix} 1\\-1\\1\\-1\\1\\-1\\1\\-1 \end{bmatrix} \;+\; 2\cdot\mathrm{Im}\left(F_8(:,1{:}3)\right)\begin{bmatrix} b_1\\b_2\\b_3 \end{bmatrix}. \tag{2}$$

The real and imaginary parts of $F_8(:,1{:}3))$ are relevant to the computation (1). Indeed, if $n = 2m$, then

$$\begin{aligned} [F_n]_{kj} &= \omega_n^{kj} \;=\; \left(\cos\left(\frac{2\pi}{n}\right) - i\sin\left(\frac{2\pi}{n}\right)\right)^{kj} \\ &= \cos\left(\frac{2kj\pi}{2m}\right) - i\sin\left(\frac{2kj\pi}{2n}\right) \;=\; \cos\left(\frac{kj\pi}{m}\right) - i\sin\left(\frac{kj\pi}{m}\right). \end{aligned}$$

It follows from (2) that to solve the $n = 8$ version of (1) we must compute $a \in \mathbb{R}^5$ and $b \in \mathbb{R}^3$ so that

$$y \;=\; \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} \;=\; \frac{1}{2}F_8 \begin{bmatrix} a_0 \\ a_1 + ib_1 \\ a_2 + ib_2 \\ a_3 + ib_3 \\ a_4 \\ a_3 - ib_3 \\ a_2 - ib_2 \\ a_1 - ib_1 \end{bmatrix}.$$

Thus, if

$$z = 2F_8^{-1}y$$

then $a = \mathrm{Re}(z(0{:}4))$ and $b = \mathrm{Im}(z(1{:}3))$. To carry out this linear system solve we use a crucial (but easily verified) fact about the inverse of the DFT matrix:

$$F_n^{-1} \;=\; \frac{1}{n}\bar{F}_n.$$

Use this fact to show how the vectors a and b can be extracted from the DFT of y. Extrapolating from the $n = 8$ case, implement the following function:

```
  function [a,b] = TrigInterp(y)
% y is a column n vector and n is a power of 2.
% a is a column m+1 vector and b is a column m-1 vector where m = n/2 so that if
%
%    f(t) = a(1)cos(0*t) + a(2)cos(t)  + a(3)cos(2t) + ... + a(m+1)cos(mt) +...
%           b(1)sin(t)   + b(2)sin(2t) + b(3)sin(3t) + ... + b(m-1)sin((m-1)t)
% then
%                        f(2*pi*k/n) = y(k+1), k=0:n-1
%
```

Make effective use of the MATLAB `fft` function. (To force issues, you cannot use `ifft`.) Submit your implementation of `TrigInterp` to CMS.

2 Periodic Functions and the Composite Simpson Rule

The composite Simpson rule defined by

$$S(a,b,n) \;=\; \sum_{k=1}^{n} \frac{h}{6}\left(f(z_k) + 4f\left(\frac{z_k + z_{k+1}}{2}\right) + f(z_{k+1})\right)$$

where

$$h \;=\; \frac{b-a}{n} \qquad \text{and} \qquad z = \texttt{linspace}(a,b,n+1).$$

It satisfies

$$\int_a^b f(x)dx \;=\; S(a,b,n) \;-\; \frac{f^{(4)}(\eta)}{2880}h^4(b-a)$$

where $a \le \eta \le b$. Complete the following function so that it performs as specified:

```
  function Q = PeriodicSimpson(f,a,b,P,M4,tol)
% f is a handle to a function f that is defined on [a,b] and whose
% fourth derivative is bounded by M4. f has period P, i.e. f(x+P) = f(x) for all x.
% Q is an estimate of the integral of f from a to b with the property that
% |Exact Integral - Q| <= tol where tol>0.
```

Your implementation must approximate all integrals using the composite Simpson rule. Strive to minimize the number of f-evaluations. You may assume that `f` is vectorized. A call of the form `f(v)` where `v` is an m-vector counts as m function evaluations. You will want to exploint periodicity. For example, if $b = a + 3P$, then

$$I \;=\; \int_a^b f(x)dx \;=\; 3\int_a^{a+P} f(x)dx$$

In this case you would approximate I with $3S(a, a{+}P, n)$ with the smallest possible n value that guarantees that the overall error is no larger than `tol`. It gets more complicated if b is arbitrary!. Submit `PeriodicSimpson` to CMS.

3 A Tough Integral

It turns out that

$$\lim_{\epsilon\to 0} \int_\epsilon^1 \frac{1}{x}\cdot\cos\left(\frac{\ln(x)}{x}\right) dx \;=\; .3233674316...$$

Write a script `ToughProb` that confirms this result. Make effective use of MATLAB's quadrature software. Call your script `ToughProb` and submit it to CMS. Be sure to explain the logic of your solution approach in the comments.

4 Position via Acceleration Snapshots

Let $a(t)$ denote the acceleration of an object at time t. If v_0 is the object's velocity at $t = 0$, then the velocity at time t is prescribed by

$$v(t) = v_0 + \int_0^t a(\tau)d\tau.$$

Likewise, if x_0 is the position at $t = 0$, then the position at time t is given by

$$x(t) = x_0 + \int_0^t v(\tau)d\tau.$$

Now suppose that we have snapshots $a(t_i)$ of the acceleration at times t_i, $i = 1{:}m$, $t_1 = 0$. Assume that we know the initial position x_0 and velocity v_0. Our goal is to estimate position from this data. Spline quadrature will be used to approximate the preceding integrals. Let $S_a(t)$ be the not-a-knot spline interpolant of the acceleration data $(t_i, a(t_i))$, $i = 1{:}m$, and define

$$\tilde{v}(t) = v_0 + \int_0^t S_a(\tau)d\tau.$$

Let $S_v(t)$ be the not-a-knot spline interpolant of the data $(t_i, \tilde{v}(t_i))$, $i = 1{:}m$, and define

$$\tilde{x}(t) = x_0 + \int_0^t S_v(\tau)d\tau.$$

The spline interpolant $S_x(t)$ of the data $(t_i, \tilde{x}(t_i))$ is then an approximation of the true position. Write a function

```
  function Sx = Position(a,t,x0,v0)}
% a and t are column m-vectors with 0 = t(1) < t(2) <...< t(m).
% a(i) is the acceleration of an object at time t(i)
% x0 and v0 are the  position and velocity of the object time 0.
% Sx the pp-representation of a spline that approximates position.
```

Submit `Position`to CMS. It must make effective use of the MATLAB `spline` function. A test script will be provided.