CS 222 - Practice Problems
July 20, 2001

Here are 3 practice problems to get you thinking about the upcoming midterm. You are encouraged to work with others to do these problems.

1. **Faster Trigonometric Interpolation**

   The trigonometric problem is discussed in Section 2.4.4 and was on HW2. The central computation involves the solution of an $n \times n$ system of linear equations. In particular, if $f_0, \ldots, f_{n-1}$ is the given data and $n = 2m$, then we find real numbers $a_0, \ldots, a_m$ and $b_1, \ldots, b_{m-1}$ such that

$$f_k = a_0 + \sum_{j=1}^{m-1} (a_j \cos(kj\pi/m) + b_j \sin(kj\pi/m)) + a_m \cos(\pi k) \qquad k = 0 : n-1$$

   Here is the system for $n = 6$:

$$\begin{bmatrix} \cos(0^\circ) & \cos(0^\circ) & \cos(0^\circ) & \cos(0)^\circ) & \sin(0^\circ) & \sin(0^\circ) \\ \cos(0^\circ) & \cos(60^\circ) & \cos(120^\circ) & \cos(180)^\circ) & \sin(60^\circ) & \sin(120^\circ) \\ \cos(0^\circ) & \cos(120^\circ) & \cos(240^\circ) & \cos(360)^\circ) & \sin(120^\circ) & \sin(240^\circ) \\ \cos(0^\circ) & \cos(180^\circ) & \cos(360^\circ) & \cos(540)^\circ) & \sin(180^\circ) & \sin(360^\circ) \\ \cos(0^\circ) & \cos(240^\circ) & \cos(480^\circ) & \cos(720)^\circ) & \sin(240^\circ) & \sin(480^\circ) \\ \cos(0^\circ) & \cos(300^\circ) & \cos(600^\circ) & \cos(900)^\circ) & \sin(300^\circ) & \sin(600^\circ) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix}$$

   The chapter 2 function that sets this up and solves this system is:

```
function F = CSInterp(f)
n = length(f); m = n/2;
tau = (pi/m)*(0:n-1)';
P = [];
for j=0:m,   P = [P cos(j*tau)]; end
for j=1:m-1, P = [P sin(j*tau)]; end
y = P\f;
F = struct('a',y(1:m+1),'b',y(m+2:n));
```

   Note that subscripts start at 1, not 0. Therefore `f(k)` houses f_{k-1} for $k = 1 : n$. Likewise, the coefficient a_{k-1} is returned in `F.a(k)` for $k = 1 : m+1$.

   This is an $O(n^3)$ algorithm that involves $O(n^2)$ storage. This algorithm can be reduced to $O(n^2)$ flops with $O(n)$ storage using the fact that $P^T P = D$, where D is a diagonal matrix.

   For example, when $n = 6$,

$$D = \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

   In general, if `D=diag(d(1) d(2) ... d(k))` then

$$d_k = \begin{cases} n & \text{if } k = 1 \text{ or } k = m+1 \\ m & \text{otherwise} \end{cases}$$

Use this fact to adjust `CSInterp.m` to involve $O(n^2)$ flops and $O(n)$ storage. This means you should not explicity form P.

2. **Periodic Cubic Splines**

   We would like to use a cubic spline to interpolate a periodic function, f. That is, $f(x) = f(x+T)$ for some period $T > 0$.

   Given points (x_i, y_i) for $i = 1 : n$ where

$$0 = x_1 < x_2 < \ldots < x_{n-1} < x_n = T$$

   we need to find a piecewise cubic function $S(x)$ that

   - interpolates the data
   - has first derivative continuity
   - has second derivative continuity
   - is periodic

   We know each cubic has 4 unknowns. Since there are $n-1$ cubics, this gives $4(n-1)$ total unknowns.

   (a) Write down the $2(n-1)$ constraints that cause S to interpolate the data.

   (b) Write down the $n-2$ constraints that cause S to have continuous first derivatives at x_2 through x_{n-1}.

   (c) Write down the $n-2$ constraints that cause S to have continuous second derivatives at x_2 through x_{n-1}.

   (d) You should have written down $4(n-1)-2$ constraints so far. There are 2 remaining. Use these constraints to satisfy the periodicity.

3. **Fun with Splines**

   Write MATLAB code to compute the arc length of a cubic spline curve. Recall that the arc length of $S(x)$ from a to b is

$$\int_a^b \sqrt{1 + (S'(x))^2} dx$$

   Use the Vandermonde representation for the cubic to make calculations easier.

   Do not worry too much about MATLAB syntax when you write your code. You may simply write down the steps you would take in order to calculate the length. You have the MATLAB built-in function `quad` at your disposal.