# CS 222 - Practice Problems Solutions
July 20, 2001

1. **Faster Trigonometric Interpolation (but not the fastest!)**

   We need to solve the system $Py = f$ in $O(n^2)$ flops. Rewriting this system we have that $y = P^{-1}f$. Using the fact that $P^TP = D$ we have:

$$P^TP = D \quad \Rightarrow \quad D^{-1}P^TP = I \quad \Rightarrow \quad P^{-1} = D^{-1}P^T$$

   Substituting what we got for $P^{-1}$ above into $y = P^{-1}f$ we have

$$y = D^{-1}P^Tf$$

   Now, note that

$$\text{if} \quad D = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \quad \text{then} \quad D^{-1} = \begin{bmatrix} \frac{1}{d_1} & 0 & 0 & 0 \\ 0 & \frac{1}{d_2} & 0 & 0 \\ 0 & 0 & \frac{1}{d_3} & 0 \\ 0 & 0 & 0 & \frac{1}{d_4} \end{bmatrix}$$

   Note that the operation $P^Tf$ is $O(n^2)$ flops since we can think of this as $n$ inner products of a column of $P^T$ and $f$. We would only store the appropriate column of $P^T$ resulting in $O(n)$ storage. Taking into account $D^{-1}$, then we just need to divide the inner products by the appropriate $d_k$:

```
function F=myCSInterp(f)
n=length(f); m=n/2; y=zeros(n,1);
tau=(pi/m)*(0:n-1);

for j=0:m
  if (j==0 | j==m)
    y(j+1)=(cos(j*tau)'*f)/n;
  else
    y(j+1)=(cos(j*tau)'*f)/m;
    y(j+m+1)=(sin(j*tau)'*f)/m;
  end
end

F=struct('a',y(1:m+1),'b',y(m+2:n));
```

2. **Periodic Cubic Splines**

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

   (a) The constraints that cause $S$ to interpolate the data are ($q_i(x)$ is the $i$-th local cubic)':

$$\begin{aligned} q_i(x_i) &= y_i \quad \text{for } i = 1 : n-1 \\ q_i(x_{i+1}) &= y_{i+1} \quad \text{for } i = 1 : n-1 \end{aligned}$$

   (b) The constraints that cause $S$ to have continuous first derivatives at $x_2$ through $x_{n-1}$ are:

$$q_i'(x_{i+1}) = q_{i+1}'(x_{i+1}) \quad \text{for } i = 1 : n-2$$

(c) The constraints that cause $S$ to have continuous second derivatives at $x_2$ through $x_{n-1}$ are:

$$q_i''(x_{i+1}) = q_{i+1}''(x_{i+1}) \quad \text{for } i = 1 : n-2$$

(d) The periodicity constraints are that $S'(0) = S'(T)$ and $S''(0) = S''(T)$ (note that the constraint $S(0) = S(T)$ has already been accounted for in (a)). This translates to:

$$\begin{aligned} q_1'(x_1) &= q_{n-1}'(x_n) \\ q_1''(x_1) &= q_{n-1}''(x_n) \end{aligned}$$

3. **Fun with Splines**

Here are the steps to calculate the arc length:

(a) Find the spline coefficients $a_i, b_i, c_i, d_i$:

`V=MySpline(x,y);`

where `MySpline` is a function that computes the coefficients using the Vandermonde representation. This function returns a matrix $V$ that contains the coefficients.

(b) Create a function for $S'(x)$ that evaluates the derivative (using `Locate.m` and Horner's rule for evaluation)

```
function y=g(z,x,V);
% z is a vector of evaluation points
% x is a vector of interpolation points
% V is a matrix that contains the coefficients of the spline
n=length(z);
i=zeros(n,1);
for k=1:n
  i(k)=Locate(x,z);
end
sp=V(i,2) + z.* ( 2*V(i,3) + 3*V(i,4) .*z); %Horner's rule
y=sqrt(1+sp.*sp);
```

(c) Integrate using `quad`

```
n=length(x);
arclength=quad('g',x(1),x(n),1e-5,[],x,y);
```

For this problem, I'm more concerned with the steps used to find the arc length than the actual MATLAB code. Nevertheless, I have included the code to aid in studying.