

**Homework 2**  
**Math 128A**  
**Due Monday, 10/27/08, 11:00 a.m.**

1. Given the points  $(x_i, y_i)$  for  $i = 0 \dots n$  the Newton form of the interpolating polynomial is

$$p(x) = a_0 + a_1(x - x_0) + \dots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}).$$

- (a) Write a routine that takes arrays of data  $x$  and  $y$  and returns an array,  $a$ , containing the coefficients of the Newton form of the interpolating polynomial.
  - (b) Write a routine to evaluate the interpolating polynomial at a point using Horner's method (nested multiplication). Your routine will take as input the arrays  $x$  and  $a$  and the scalar  $z$ , and it will return  $p(z)$ .
  - (c) Compute the coefficients of the Newton form of the interpolating polynomial for the fifth degree polynomial that interpolates  $f(x) = \cos(2\pi x)$  at the points  $x_j = j/5$  for  $j = 0, \dots, 5$ . Display the results in a table.
  - (d) Make a plot of  $f(x)$  and the interpolating polynomial,  $p(x)$ , from part (c) on the same axes for  $0 \leq x \leq 1$ . Plot the difference of  $f$  and  $p$  for  $0 \leq x \leq 1$ .
  - (e) Estimate the maximum of  $|f(x) - p(x)|$  on the interval  $[0, 1]$  by evaluating  $f$  and  $p$  for a large number of points between 0 and 1.
  - (f) Evaluate  $p(1.5)$ ,  $p(2.0)$ , and  $p(2.5)$  and report the results. Why are the values of  $p$  so different from the values of  $f$  at these points compared to points in  $[0, 1]$ ?
2. On the last page of this assignment, there is a MATLAB function that takes as input vectors  $x$  and  $y$ , and returns an array,  $P$ , which contains the coefficients of the natural cubic spline through the points  $(x_i, y_i)$ . The cubic spline is defined as  $S(x) = S_i(x)$  for  $x_i \leq x \leq x_{i+1}$ , and

$$S_i(x) = P_{i1} + P_{i2}(x - x_i) + P_{i3}(x - x_i)^2 + P_{i4}(x - x_i)^3.$$

- (a) Write a routine that takes as input the arrays  $x$  and  $P$  and the scalar  $z$  and returns  $S(z)$ .
- (b) In the absence of derivatives at the endpoints, one commonly uses the *not-a-knot* boundary condition rather than natural boundary conditions. If the points are indexed from  $j = 0 \dots n$ , then the not-a-knot condition requires that  $S'''(x)$  be continuous at  $x_1$  and  $x_{n-1}$ . For  $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ , our textbook derives the system of linear equations for the values of  $c_i$ :

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$$

for  $i = 1 \dots n - 1$ , where  $h_i = x_{i+1} - x_i$ . We need two more equations to be able to solve for the  $c$  values. Derive the two equations that come from the not-a-knot conditions involving only  $c$  values and  $h$  values.

- (c) Write a routine to compute the coefficients of a cubic spline with not-a-knot boundary conditions. This requires only minor changes from the natural spline code provided.

- (d) Plot the natural and not-a-knot splines that interpolate  $f(x) = \cos(2\pi x)$  at the points  $x_j = j/5$  for  $j = 0, \dots, 5$ . Which spline looks more like the cosine function?
- (e) For each spline, find the maximum error in using the spline to approximate  $f(x)$  between  $x = 0$  and  $x = 1$ .
3. (a) Use  $N = 11$  equally spaced points between  $x = -1$  and  $x = 1$ , and plot the  $N - 1$ -degree polynomial that interpolates

$$g(x) = \frac{1}{1 + 25x^2}$$

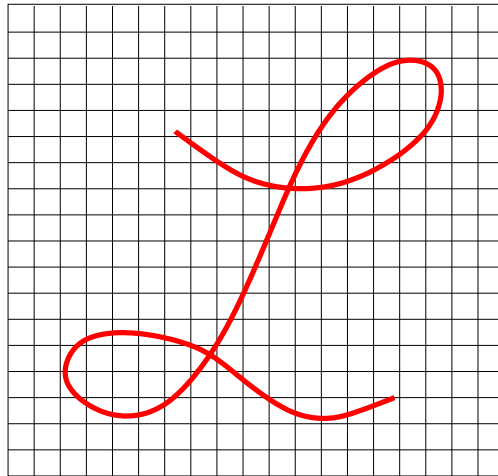
at these points for  $-1 \leq x \leq 1$ . Repeat for  $N = 21$ . What is the maximum error in using these interpolating polynomials to approximate  $g(x)$ .

- (b) Repeat part (a), but use a not-a-knot spline to interpolate the data.
- (c) Repeat part (a) using the points

$$x_j = \cos\left(\frac{2j+1}{2N+1}\pi\right), \quad j = 0 \dots N,$$

instead of equally spaced points.

- (d) Comment on your results.
4. Use cubic splines to represent a curve  $(x(s), y(s))$  that is in the shape of the script letter L below. List the points you used to make the splines, and make a plot of your curve. You may use the provided grid to estimate the nodes by hand, or use software to help identify points on the curve. For example, in MATLAB the commands `imread`, `image`, and `ginput` could be used to aid in selecting your points.



```

% Natural spline coefficients
%
% Input:  x - n by 1 vector of the nodes
%         y - n by 1 vector with the corresponding function values
% Output: P - n-1 by 4 matrix
%         the elements of the ith row of P give the coefficients
%         of the cubic between [x(i),x(i+1)] as
%         Si(x) = P(i,1) + P(i,2)*(x-x(i))
%                + P(i,3)*(x-x(i))^2 + P(i,4)*(x-x(i))^3
%
function P=naturalspline(x,y);

% initialize P
%
n = length(x);
P = zeros(n-1,4);

% compute the distances between the points
%
h = x(2:n) - x(1:n-1);

% set up the tridiagonal linear system to solve
%
d0 = 2*(h(1:n-2)+h(2:n-1));    % diagonal
d1 = h(2:n-2);                 % super and sub diagonal

% form the matrix and rhs
%
A = diag(d0,0) + diag(d1,-1) + diag(d1,1);
b = 3*(y(3:n)-y(2:n-1))./h(2:n-1) - 3*(y(2:n-1)-y(1:n-2))./h(1:n-2);

% solve the linear system
%
z = A\b;

% append the natural boundary conditions
%
z = [0; z; 0];

% compute the coefficients
%
P(:,1) = y(1:n-1);
P(:,2) = (y(2:n) - y(1:n-1))./h - h.*(z(2:n)+2*z(1:n-1))/3;
P(:,3) = z(1:n-1);
P(:,4) = (z(2:n)-z(1:n-1))./(3*h);

```