

ACM 106a, Problem Set 2

(Polynomial and Spline Interpolation)

Due: Thursday, November 1, 2007

Theory

These problems are taken from chapter 5 of the book *Bézier and B-Spline Techniques*. Scans of the relevant sections are available for download from the class web site.

1. Given a spline $s(u) = \sum_{i=0}^m c_i N_i^n(u)$ with the knots a_0, \dots, a_{m+n+1} , show that

$$\int_{a_0}^{a_{m+n+1}} s(u) du = \sum_{i=0}^m \frac{a_{i+n+1} - a_i}{n+1} c_i.$$

2. Use the derivative formula of B-splines to derive the recursion formula of de Boor, Mansfield, and Cox by induction.

MATLAB

1. Consider the function $f(x) = \frac{1}{1+x^2}$ over the interval $[-5, 5]$. This function has no singularities on the real line; however, it does blow up in the complex plane at $x = \pm i$. Therefore, if we approximate this “nearly singular” function with a polynomial, we might want to be careful about the method we use to evaluate this polynomial at interpolating points. To study this, do the following:
 - (a) Evaluate $f(x)$ at the 21 equally spaced points $x = -5, -4.5, \dots, 4.5, 5$. Then construct the degree-20 polynomial interpolating these points, using the Newton divided-difference form described in Stoer & Bulirsch section 2.1.3.
 - (b) Next, you will compare two methods for evaluating this polynomial on a finer grid: the 201 equally spaced points $x = -5, -4.95, \dots, 4.95, 5$. First, evaluate the polynomial at these points by computing along the “top descending diagonal of the divided-difference scheme” (p. 46). After this, evaluate the same polynomial along the more numerically trustworthy “zigzag” paths described on pp. 46b–47. The zigzag path uses a different Newton representation to evaluate each interpolation point ξ , so as to minimize the rounding error.
 - (c) Plot the graph of the polynomial at these 201 points, computed using each of the two methods, and also plot the difference between the two values at each point. (Three plots total.)

2. Repeat the first problem, with one modification: in part (a), instead of interpolating $f(x)$ at the 21 equally spaced points on $[-5, 5]$, use the *Chebyshev nodes*. The roots of the $(n + 1)$ st Chebyshev polynomial $T_{n+1}(x)$ on $[-1, 1]$ are given by the formula

$$x_k = \cos\left(\frac{2n + 1 - 2k}{2n + 2}\pi\right) \quad \text{for } k = 0, 1, \dots, n.$$

Taking $n = 20$, and multiplying by 5 to transform $[-1, 1]$ to $[-5, 5]$, gives the Chebyshev nodes

-4.9860	-4.8746	-4.6544	-4.3301	-3.9092	-3.4009	-2.8166
-2.1694	-1.4738	-0.7452	0.0000	0.7452	1.4738	2.1694
2.8166	3.4009	3.9092	4.3301	4.6544	4.8746	4.9860

How does using these nodes affect the difference between the two methods of polynomial evaluation?

3. Repeat the first problem, again interpolating at the original 21 equally spaced points. This time, however, interpolate using a *cubic spline with natural boundary conditions*, as described in section 2.4.2. To set up the tridiagonal sparse linear system, you will want to use the MATLAB function `spdiags`. (For more information on this function, type `help spdiags` at the MATLAB prompt, and/or check the online documentation.)
4. For the interpolation problem you computed above, one expects to see a much closer approximation of the original function f when using either the non-uniform points (problem 2) or a lower-degree piecewise curve (problem 3). In a paragraph, describe the trade-off between cost and accuracy for these two methods. For example, evaluating a degree 20 polynomial takes many more operations than a piecewise cubic—what accuracies does one get in return, and how does this relate to the cost?