Math 151A (Sample) Project 2 Report

Jeffrey Wong

3/11/14

1 User Guide

- `divdif.m`: This implements the divided difference algorithm (3.2 in the textbook[1]) for finding the Lagrange interpolant through given points (x_i, y_i). The calling syntax is

  ```
  [P Q] = divdif(x,y)
  ```

  Inputs: The points $x_0, \cdots x_n$ and function values $y_0, \cdots y_n$.

  Outputs: The coefficients stored in P and divided difference table stored in Q, which is a lower triangular matrix. If $P = [c_0, \cdots, c_n]$,

  $$P_n(x) = c_0 + \sum_{k=1}^{n} c_k (x - x_0) \cdots (x - x_{k-1}).$$

  Note that if Q is not included as an output, it will not be computed.

- `evalP.m`: Evaluates the Lagrange interpolant P at a set of points. The calling syntax is

  ```
  pt = evalP(P,t_int,t_test)
  ```

  Inputs: The coefficients P (from `divdif.m`), interpolating points $t_0, \cdots t_n$ and a vector of test points in `t_test`.

  Outputs: The polynomial evaluated at `t_test` in the vector `pt`.

[1]Burden and Faires, *Numerical Analysis*, ninth edition.

- `spline3.m`: Generates a spline interpolating a set of points, either natural or clamped, with

  $$S_j(x) = f(x_j) + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \qquad j = 0, \cdots n-1.$$

  Implements Algorithm 3.4 and 3.5 from the textbook. An alternate form, `spline3_spdiags` is included that uses `spdiags` and MATLAB's backslash operator to solve the tridiagonal system. The calling syntax is:

  ```
  S = spline3(t,ft,varargin)
  ```

  Inputs: The interpolating points $t_0, \cdots, t_n$ and function values $y_0, \cdots y_n$ in `ft`. The third argument is a string, either `'natural'` or `'clamped'`. If the latter is chosen, then the derivatives at the left and right endpoints can be supplied:

  ```
  S = spline3(t,ft,'clamped',fp0,fpn).
  ```

  Outputs: The spline coefficients in the $n \times 4$ matrix `S`, in the form

  $$S = \begin{pmatrix} y_0 & b_0 & c_0 & d_0 \\ \vdots & \vdots & \vdots & \vdots \\ y_{n-1} & b_{n-1} & c_{n-1} & d_{n-1}. \end{pmatrix}$$

- `evalS.m`: Evaluates the spline S at a set of points. Calling syntax:

  ```
  st = evalS(S,t_int,t_test)
  ```

  Inputs: The coefficients S (from `spline3.m`), interpolating points $t_0, \cdots t_n$ and a vector of test points in `t_test`.

  Outputs: The spline evaluated at the test points, in the vector `st`.

- `findpath.m`: Computes an interpolant for a path $(x(t), y(t))$ in $\mathbb{R}^2$ defined by a given set of data, and evaluates it at a set of points. Calling syntax:

  ```
  [path, coeff] = findpath(ip,tp,method)
  ```

  Inputs: The data in the matrix `ip` where the i-th row of `ip` contains (t_i, x_i, y_i). The vector `tp` contains the times at which the interpolants are to be evaluated. The method is a string, either `'lagrange'` (uses `divdif` and `evalP`) or `'spline'` (uses `spline3` and `evalS`).

  Outputs: The path evaluated at the points `tp` (in the same format as `ip`) and the relevant coefficient matrix.

2 Implementation

2.1 Derivation of the spline algorithm

The derivation presented here is nearly identical to that described in the textbook. The important difference here is that each row is normalized by dividing by $h_i + h_{i+1}$, so that the diagonal entries are each 2. If the points are nearly equally spaced, then each entry in A is of order unity (rather than h), which reduces error when dividing by those values in the algorithm. Hence this version is more stable than the textbook variant.

Derivation: Let $t_0, \cdots t_n$ be the interpolating points with function values $y_0, \cdots y_n$. Let the splines on each sub-interval have the form:

$$S_i(x) = y_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3, \qquad i = 0, \cdots n - 1.$$

Set $h_i = t_{i+1} - t_i$ and $m_i = 2c_i$, and (for later use)

$$\mu_i = \frac{h_i}{h_{i+1} + h_i}, \qquad \tau_{i-1} = 1 - \mu_i, \text{ for } i = 1, \cdots n - 1$$

The goal is to obtain an $(n + 1) \times (n + 1)$ matrix A and vector r such that

$$Am = r. \tag{1}$$

Continuity of S'' at interior nodes gives

$$m_i + 6d_i h_i = m_{i+1}, \text{ for } i = 1, \cdots n - 1. \tag{2}$$

Continuity of S' gives

$$b_i + m_i h_i + 3d_i h_i^2 = b_{i+1}, \text{ for } i = 1, \cdots n - 1. \tag{3}$$

From continuity of S,

$$y_i + b_i h_i + \frac{1}{2} m_i h_i^2 + d_i h_i^3 = y_{i+1}, \text{ for } i = 1, \cdots n - 1. \tag{4}$$

Now use (2) to eliminate d_i from (3) and (4) and shift the index of (3) down by one to obtain the system

$$b_i - b_{i-1} = \frac{1}{2} h_{i-1}(m_i + m_{i-1}), \tag{5}$$

$$y_{i+1} - y_i = b_i h_i + \frac{h_i^2}{6}(m_{i+1} + 2m_i). \tag{6}$$

Finally, use (6) to eliminate b_i and b_{i-1} from (5), which gives

$$h_{i-1}m_{i-1} + 2(h_{i-1} + h_i)m_i + h_i m_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}). \tag{7}$$

Dividing by $h_{i-1} + h_i$ yields the equations for the interior values of m_i, and also provides an expression for the (interior) values of r in (1):

$$r_i = \frac{1}{h_i + h_{i-1}} \left(\frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1}) \right), \quad \text{for } \ i = 1, \cdots n-1.$$

For natural boundary conditions, $m_0 = m_n = 0$, so the full system (1) is, for $i = 1, \cdots n-1$,

$$\begin{cases} 2m_0 + \mu_0 m_1 = r_0 \\ \mu_{i-1}m_{i-1} + 2m_i + \tau_{i-1}m_{i+1} = r_i \\ \lambda_{n-1}m_{n-1} + 2m_n = r_n \end{cases} \tag{8}$$

if we take $\mu_0 = 0$, $\lambda_{n-1} = 0$ and $r_0 = r_n = 0$.

Clamped boundary conditions are equivalent to requiring $y'_0 = b_0$ and $y'_n = b_n$. Applying this and (6) at the left boundary:

$$\frac{y_1 - y_0}{h_0} - y'_0 = \frac{1}{6}h_0^2(m_1 + 2m_0). \tag{9}$$

And considering (5) with $i = n$ (the right boundary condition) along with (6) with $i = n-1$:

$$6b_n - \frac{6}{h_{n-1}}(y_n - y_{n-1}) = h_{n-1}m_{n-1} + 2h_{n-1}m_n. \tag{10}$$

Writing these conditions in terms of μ, τ:

$$2m_0 + m_1 = \frac{6}{h_0^3}(y_1 - y_0) - \frac{6y'_0}{h_0^2}, \tag{11}$$

$$\mu_{n-1}m_{n-1} + 2m_n = \frac{6y'_n}{h_{n-1}} - \frac{6}{h_{n-1}^2}(y_n - y_{n-1}). \tag{12}$$

This determines $\mu_0 = 0$, $\lambda_{n-1} = 1$ and the values of r_0 and r_n in (8).

2.2 Runge Phenomenon

Let P_n be the Lagrange interpolant using n equally spaced nodes in $[-1, 1]$ for the function

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

Let C_n be the Lagrange interpolant using the Chebyshev nodes $x_j = \cos\frac{(2j-1)\pi}{n}$. Though it is not necessary, also let S_n be the clamped spline through the equally spaced nodes. We observe Runge's phenomenon,

$$||P_n - f||_\infty \to \infty \text{ as } n \to \infty,$$

which is clear from Figure 2 (also Figure 1, which shows smaller N) and Figure 3, which shows the growth of the error. The Lagrange error is

$$E_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{k=0}^{n}(x - x_i).$$

The trouble is that the product term is not small enough as $n \to \infty$ to counteract the growth in the derivative term[2] If the derivative On the other hand, the Chebyshev nodes minimize the max norm of this product term (with the upper bound being 2^{-n}), which leads to the error bound[3]

$$||P_n - f||_\infty \le \frac{1}{2^n(n+1)!}||f^{(n+1)}||_\infty.$$

In particular, the error tends to zero (usually linearly) as $n \to \infty$. For the spline, the error is $O(h^4) = O(n^{-4})$, so the spline error also converges to zero but not as quickly (asymptotically). The apparent linear convergence is illustrated in Figure 3; the actual approximations are shown in Figure 1 for small N.

[2]This relationship can be made a bit more precise; one can show that if the derivatives do not increase too quickly, then even equally spaced points will be adequate to have $||P_n - f||_\infty \to 0$. A nice reference: Epperson, James F. "On the Runge example." Amer. Math. Monthly 94.4 (1987): 329-341.

[3]See Section 8.3 and Theorem 8.10 of the textbook for the details. Chebyshev polynomials are an interesting subject with important applications in approximation theory, where they are used to build good approximations in the L^∞ norm.

2.3 Path problem

The error bound for the clamped spline[4] is (Theorem 3.13 from the book)

$$||S(t) - x(t)||_\infty \leq \frac{5M_x}{384}h^4,$$

where h is the (equal) spacing between points and $M_x = \max_t |x^{(4)}(t)|$ is unknown. An analogous bound holds for $y(t)$. The Lagrange error is

$$||P_n(t) - x(t)||_\infty \leq \frac{M_{x,n+1}}{(n+1)!}\left|\prod_{k=0}^{n}(t-t_k)\right|$$

where $M_{x,n+1} = \max_t x^{(n+1)}(t)$ is unknown.

The interpolation code was first tested on the example problems given in the book. For the Runge problem, the error using Chebyshev nodes is increasing past $N \approx 40$;, this could be due to an error in the code (not yet fixed). Using equidistant points in $[0, 3.1]$ with $h = 0.01$ yields good results for both methods (see Figure 5), and also for the randomly distributed numbers given in `data.mat` (Figure 4).

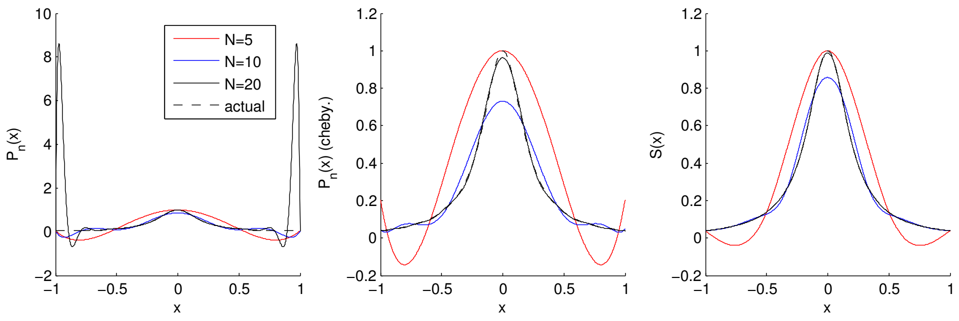

Figure 1: Plot of Lagrange interpolant P_n with $5, 10$ and 20 nodes, equally spaced (left), and Chebyshev nodes (middle). The right plot shows the equally spaced spline interpolant. The actual function overlaps the black curve in the middle and right plots.

[4]For a natural spline, the error is $O(h^2)$ near the boundary (the same order of error as linear interpolation). However, the error in the interior is still $O(h^4)$; more precisely, for each $[a', b']$ with $a < a' < b' < b$ the error is $O(h^4)$ in $[a', b']$.

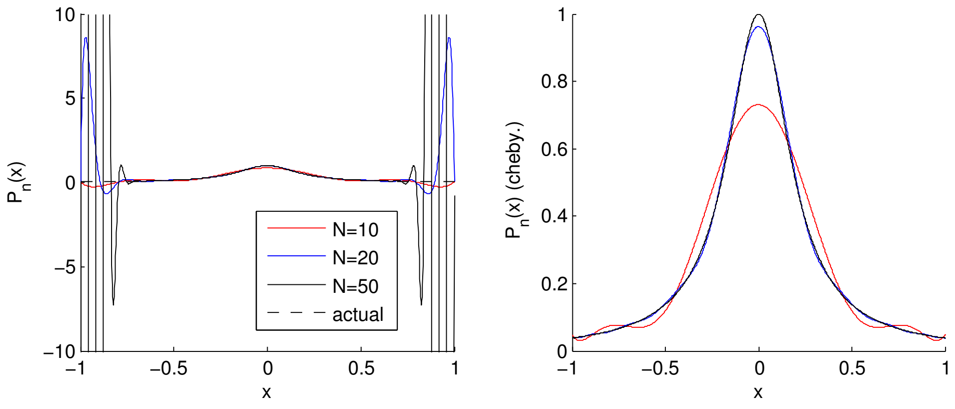

Figure 2: Plot of Lagrange interpolant P_n with $10, 20$ and 50 nodes, equally spaced (left), and Chebyshev nodes (middle). The equally spaced interpolant oscillates near the endpoints of the interval, up to about 10^6.

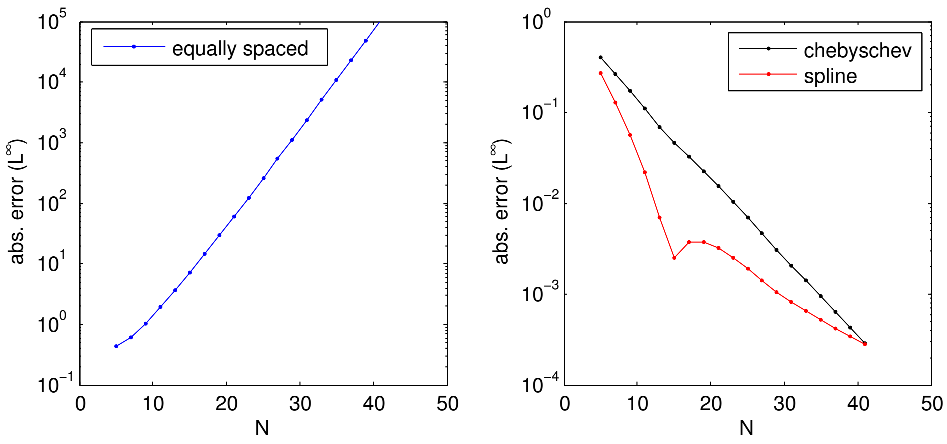

Figure 3: Plot of the error in the interpolant $||P_n - f||_\infty$ for the equally spaced nodes (left), the Chebyshev nodes and the spline (right). The number of points used are odd numbers from 7 to 41. Note that the Chebyshev error is decreasing linearly, while the spline error is not

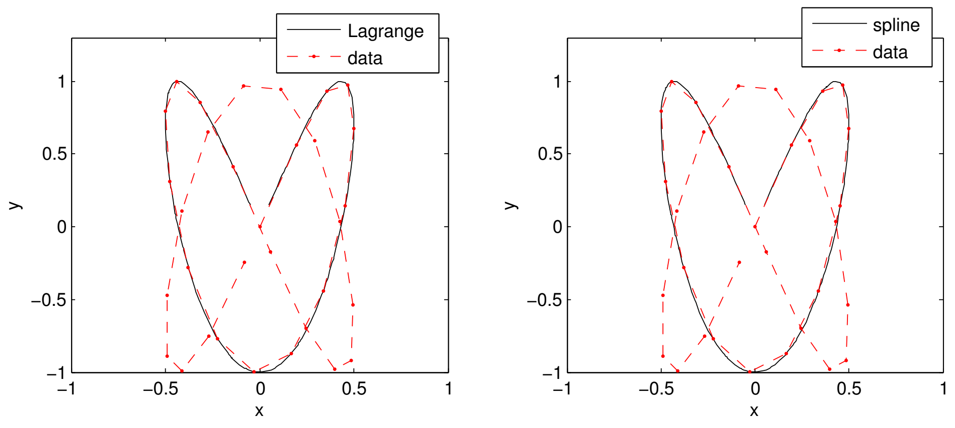

Figure 4: Plot of the Lagrange and clamped spline interpolants against the given data, showing good agreement in both cases. The curves are samples at a set of uniformly distributed random points in $[0, 3.1]$.

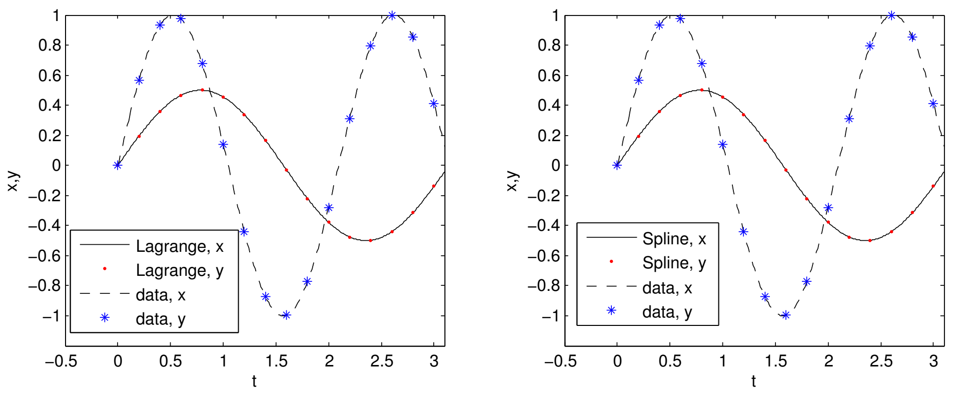

Figure 5: Plots of the approximated $x(t)$ and $y(t)$ (lines) against the data (dots) for Lagrange interpolation (left) and cubic spline (right). The sample points are in $[0, 3.1]$ with a uniform spacing of 0.01.