Computer Science CSCC51 — Spring 1999
Scarborough Campus — University of Toronto

Homework Assignment #2
Due: Friday February 26 1999, in tutorial
Worth: 8%

1. (20 points) This question is to be done by hand calculations.

(a) (10 points) Write down all conditions on a_i, b_i and c_i, $i = 1, 2$, which guarantee that the following function $\phi(x)$ is a continuous quadratic piecewise polynomial in $\mathbb{R}$ with knots $0, 1, 2$. Describe what each condition means. (Do not solve the system.)

$$\phi(x) = \begin{cases} a_1x^2 + b_1x + c_1 & \text{for } 0 \le x \le 1 \\ a_2x^2 + b_2x + c_2 & \text{for } 1 \le x \le 2 \\ 0 & \text{elsewhere} \end{cases}$$

(b) (10 points) With the additional conditions $\phi(x) = 1$, $\phi'(0) = -1$, and $\phi'(2) = 1$ determine a_i, b_i and c_i, $i = 1, 2$. Show your calculations and the numerical values for a_i, b_i and c_i, $i = 1, 2$.

2. (30 points)

(a) MATLAB provides a function `yi = spline(x, y, xi)` that uses cubic spline interpolation with the not-a-knot boundary conditions to find approximate values `yi` of the underlying function on the points `xi`. `x` and `y` are the given data vectors. The spline knot sequence coincides with `x`. `x` and `y` are vectors of the same length. `xi` and `yi` are also vectors of the same length, not necessarily the same as `x` (or `y`).

We would like to write a similar function `yi = clampspline(x, y, dy, xi)` that uses cubic spline interpolation with the clamped (derivative) boundary conditions to find approximate values `yi` of the underlying function on the points `xi`. `x`, `y`, `xi` and `yi` have the same meaning as above, while `dy` is a vector of length 2, holding the given derivative values at the left and right endpoints.

Implement the above function in MATLAB. Your code should follow these steps.

- Calculate the step-sizes from `x`.
- Form the cubic spline interpolation matrix for clamped boundary conditions (see p. 149 of the textbook or Equation (6) of the notes) in **sparse** format.
- Form the right side vector of the cubic spline interpolation system for clamped boundary conditions (see p. 149 of the textbook or Equation (6) of the notes).
- Solve the linear system to obtain the coefficients c_j, $j = 0, \dots, n$.

- Recover the rest of the coefficients of the cubic spline (see p. 145 and Equations (3.18) and (3.17) on p. 146 of the textbook or Equations (3), (2) and (4) of the notes).
- Evaluate the cubic spline at `xi` to obtain `yi`.

(b) Consider three sets of data given by the following piece of MATLAB code:

```
x1  = [1     2   5   6   7   8  10  13  17];          % Curve 1
y1  = [3.0 3.7 3.9 4.2 5.7 6.6 7.1 6.7 4.5];
dy1 = [1.0  -0.67];
x2  = [17    20  23  24  25  27  27.7];               % Curve 2
y2  = [4.5 7.0 6.1 5.6 5.8 5.2 4.1];
dy2 = [2.0  -4.0];
x3  = [27.7 28  29  30];                              % Curve 3
y3  = [ 4.1  4.3 4.1 3.0];
dy3 = [ 0.33 -1.5];
```

Each set of data corresponds to a curve. For each curve the x and y values on several points are given, as well as the y' values on the two endpoints of the curve.

Consider also the set of equidistant evaluation points $\{(xi)_i,\ i = 1, \ldots, 290\}$ in the interval $[1, 30]$, with $(xi)_1 = 1$ and $(xi)_{290} = 30$. Write a MATLAB program that

- constructs the clamped cubic spline interpolant of each of the above sets of data, evaluates it on the evaluation points and plots the resulting (three) vectors versus the (appropriate) evaluation points in one graph (Figure 1);
- constructs the non-a-knot cubic spline interpolant of each of the above sets of data, evaluates it on the evaluation points and plots the resulting (three) vectors versus the (appropriate) evaluation points in one graph (Figure 2);
- constructs the linear spline interpolant of each of the above sets of data, evaluates it on the evaluation points and plots the resulting (three) vectors versus the (appropriate) evaluation points in one graph (Figure 3);
- constructs the (appropriate degree) polynomial interpolant of each of the above sets of data, evaluates it on the evaluation points and plots the resulting (three) vectors versus the (appropriate) evaluation points in one graph (Figure 4);

Comment on what you observe. Notes:

- This question is inspired by exercise 28 on p. 157 of the Burden and Faires textbook. (Note though that the data are slightly different.)
- To create a banded (e.g. tridiagonal) matrix in sparse format use `spdiags()` in MATLAB.
- You can find an electronic version of the above piece of MATLAB code in the webpage of the course. (Hence, no excuses for typographical errors...)

- Use your own `clampspline` function (developed in Part (a)) to get the values of the clamped cubic spline interpolant on the evaluation points.
- Use `yis = interp1(x, y, xi, 'spline');` to get the values of the non-a-knot cubic spline interpolant on the evaluation points.
- Use `yil = interp1(x, y, xi, 'linear');` to get the values of the linear spline interpolant on the evaluation points.
- Use the built-in MATLAB functions `polyfit` and `polyval` to get the values of the polynomial interpolant on the evaluation points. Note that, although the MATLAB built-in function `polyfit` is designed to construct least-squares polynomial approximations, it can be used to construct polynomial interpolants as well, if the number of data and the degree requested are set appropriately.
- Use `axis equal` to scale the axes of the above plots in order to view them appropriately.
- For each **data** point, write an `o` on the graph, in addition to the line plotted. Use `title('text')`, where `text` is an appropriate string, to show an appropriate title to each of your graphs.
- Use `print file.ps` in MATLAB to write your first graph (Figure 1) in a postscript file, then use `print -append file.ps` for each of the next 3 graphs. Then use `psnup -4 file.ps > file.ps4` on the Unix shell, to write your 4 graphs in 1 physical page. Then use `lpr file.ps4` to get a hard-copy of your graphs. (In this way, you save paper.)
- Hand in a hard-copy of your code and plots, as well as your comments.

3. (50 points)

(a) (10 points) You are to approximate the function $g(x) = \ln(x)$ by a cubic spline interpolant $c(x)$ with clamped end-conditions. What is the least number of equidistant data points in $I = \left[\frac{1}{16}, \frac{17}{16}\right]$, needed to guarantee (according to the theory) that the absolute value of the error $|g(x) - c(x)|$ will not exceed 10^{-4} on any point x in I? Justify your answer.

Note: This question is to be done by hand calculations (except possibly the last few calculations which can be done on a calculator).

(b) (40 points) Consider the function $g(x) = \ln(x)$ in the interval $I = \left[\frac{1}{16}, \frac{17}{16}\right]$. Write a MATLAB script that, for $n = 4, 8, 16, 32, 64$,

- constructs a set of equidistant points x_i, $i = 0, \ldots, n$, with $x_0 = \frac{1}{16}$ and $x_n = \frac{17}{16}$, the respective set of function values $y_i = g(x_i)$, $i = 0, \ldots, n$ and the 2 derivative values $y_0' = g'(x_0)$ and $y_n' = g'(x_n)$;

- constructs the clamped cubic spline interpolant of the data (x_i, y_i), $i = 0, \ldots, n$, y_0' and y_n', evaluates it on a set of 100 equidistant points in I (including the endpoints) and computes the maximum in absolute value error of the interpolant on the set of 100 points;
- constructs the non-a-knot cubic spline interpolant of the data (x_i, y_i), $i = 0, \ldots, n$, evaluates it on a set of 100 equidistant points in I (including the endpoints) and computes the maximum in absolute value error of the interpolant on the set of 100 points;
- constructs the linear spline interpolant of the data (x_i, y_i), $i = 0, \ldots, n$, evaluates it on a set of 100 equidistant points in I (including the endpoints) and computes the maximum in absolute value error of the interpolant on the set of 100 points;
- constructs the polynomial interpolant of the data (x_i, y_i), $i = 0, \ldots, n$, evaluates it on a set of 100 equidistant points in I (including the endpoints) and computes the maximum in absolute value error of the interpolant on the set of 100 points;
- saves the (maximum in absolute value) errors computed above in 4 vectors, say `errcs, errs, errl, errp`, each of length 5;

after completing the computations for each value of n, **outputs** the 4 vectors so that the errors appear in table format such as the following:

n	4	8	16	32	64
errcs					
errs					
errl					
errp					

plots the 4 maximum in absolute value errors versus n in **one** graph, in **log-log** scale.

Comment on the results obtained and on any warning possibly given by MATLAB. Does the experimentally computed errors of each interpolant behave as expected from theory? What are the theoretically expected approximate slopes of the error versus n plots corresponding to the clamped cubic, non-a-knot cubic and linear spline interpolants? Notes:

- Use `format short e` or equivalent (i.e. exponential with 4 decimal digits after the decimal dot) and `format compact` in MATLAB for outputting the errors.
- To get the values of the 4 interpolants on the set of 100 equidistant points in I follow the notes of question 2.
- To get a log-log scale plot use `loglog()` in MATLAB. To distinguish between the 4 plots in the graph, use solid line for the clamped cubic spline, dashed line for the non-a-knot cubic spline, dotted line for the linear spline and dashed-dotted line for the polynomial error versus n plot.
- Do **not** output more results than requested. Indicate which output corresponds to which interpolant. Hand in a hard-copy of your code and output, as well as your comments.