CSC336 Tutorial 8 – Splines

QUESTION 1 *Determine (if possible) constants $a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2$ so that the function*

$$\mathbf{Q}(x)=\begin{cases}\mathbf{Q_0}(x)=a_0+a_1x+a_2x^2 & \textit{if } 0\le x<1\\ \mathbf{Q_1}(x)=b_0+b_1(x-1)+b_2(x-1)^2 & \textit{if } 1\le x<2\\ \mathbf{Q_2}(x)=c_0+c_1(3-x)+c_2(3-x)^2 & \textit{if } 2\le x<3\\ \mathbf{Q_3}(x)=0 & \textit{elsewhere}\end{cases}$$

is a quadratic spline in $\mathcal{R}$.

SOLUTION:

Since $\mathbf{Q}(x)$ is defined everywhere, all knots $0, 1, 2, 3$ are interior. A quadratic spline must be continuous and have continuous 1^{st} derivative. The continuity conditions are

- on $x=0$ $\quad \mathbf{Q_0}(0)=\mathbf{Q_3}(0)\Rightarrow a_0=0$
  $\mathbf{Q_0}'(0)=\mathbf{Q_3}'(0)\Rightarrow a_1=0$
- on $x=1$ $\quad \mathbf{Q_0}(1)=\mathbf{Q_1}(1)\Rightarrow a_2=b_0$
  $\mathbf{Q_0}'(1)=\mathbf{Q_1}'(1)\Rightarrow 2a_2=b_1$
- on $x=2$ $\quad \mathbf{Q_1}(2)=\mathbf{Q_2}(2)\Rightarrow b_0+b_1+b_2=c_0+c_1+c_2$
  $\mathbf{Q_1}'(2)=\mathbf{Q_2}'(2)\Rightarrow b_1+2b_2=-c_1-2c_2$

- on $x=3$ $\quad \mathbf{Q_2}(3)=\mathbf{Q_3}(3)\Rightarrow c_0=0$
  $\mathbf{Q_2}'(3)=\mathbf{Q_3}'(3)\Rightarrow c_1=0$

Thus $a_0=0, a_1=0, c_0=0, c_1=0$. Let $a_2=\alpha$ (some parameter). It follows that $b_0=\alpha, b_1=2\alpha$ and from

$$\begin{cases} b_0+b_1+b_2=c_0+c_1+c_2\\ b_1+2b_2=-c_1-2c_2\end{cases}$$

we have that

$$\begin{cases}\alpha+2\alpha+b_2=c_2\\ 2\alpha+2b_2=-2c_2\end{cases}\Rightarrow\begin{cases}6\alpha+2b_2=2c_2\\ 2\alpha+2b_2=-2c_2\end{cases}\Rightarrow\begin{cases}b_2=-2\alpha\\ c_2=\alpha\end{cases}$$

We need an additional condition to uniquely determine the constants. Following a convention, we set $\mathbf{Q}(0)+\mathbf{Q}(1)+\mathbf{Q}(2)+\mathbf{Q}(3)=1$. Since $\mathbf{Q}(0)=\mathbf{Q}(3)=0$, it follows that $\mathbf{Q}(1)+\mathbf{Q}(2)=1\Rightarrow b_0+b_0+b_1+b_2=1\Rightarrow \alpha+\alpha+2\alpha-2\alpha=1\Rightarrow \alpha=a_2=\frac{1}{2}\Rightarrow b_0=\frac{1}{2}, b_1=1, b_2=-1, c_2=\frac{1}{2}$. Thus

$$\mathbf{Q}(x)=\begin{cases}\mathbf{Q_0}(x)=\frac{1}{2}x^2 & \text{if } 0\le x\le 1\\ \mathbf{Q_1}(x)=\frac{1}{2}+(x-1)-(x-1)^2 & \text{if } 1\le x\le 2\\ \mathbf{Q_2}(x)=\frac{1}{2}(3-x)^2 & \text{if } 2\le x\le 3\\ \mathbf{Q_3}(x)=0 & \text{elsewhere}\end{cases}$$

is a quadratic spline in $\mathcal{R}$.

QUESTION 2 *Determine a, b, c and d so that the piecewise cubic polynomial*

$$\mathbf{S}(x)=\begin{cases}\mathbf{S_0}(x)=1+2x-x^3 & \textit{if } 0\le x<1\\ \mathbf{S_1}(x)=a+b(x-1)+c(x-1)^2+d(x-1)^3 & \textit{if } 1\le x\le 2\end{cases}$$

is a natural (or free) cubic spline in $[0,2]$.

SOLUTION: Since $\mathbf{S}(x)$ is defined in $[0,2]$, the interior knot is $x=1$.

Conditions to make $\mathbf{S}(x)$ a cubic spline

A cubic spline must be continous and have continuous first and second derivatives:

- continuity on $x=1$: $\mathbf{S_0}(1)=\mathbf{S_1}(1)\Rightarrow 2=a$
- continuity of the 1st derivative on $x=1$: $\mathbf{S}_0'(1)=\mathbf{S}_1'(1)\Rightarrow -1=b$
- continuity of the 2nd derivative on $x=1$: $\mathbf{S}_0''(1)=\mathbf{S}_1''(1)\Rightarrow -6=2c\Rightarrow c=-3$

Conditions to make $\mathbf{S}(x)$ natural cubic spline

A natural (or free) cubic spline must have the 2^{nd} derivative at the two end-points equal to 0:

- on $x=0$: $\mathbf{S}_0''(x)=-6x\Rightarrow \mathbf{S}_0''(0)=0$.
- on $x=2$: $\mathbf{S}_1''(2)=0\Rightarrow 6d(2-1)+2c=0\Rightarrow 3d=-c\Rightarrow d=1$

Thus $\mathbf{S}(x)=$

$$\begin{cases}\mathbf{S_0}(x)=1+2x-x^3 & \text{if } 0\le x\le 1\\ \mathbf{S_1}(x)=2-(x-1)-3(x-1)^2+(x-1)^3 & \text{if } 1\le x\le 2\end{cases}$$

is a natural (or free) cubic spline.

QUESTION 3 *Consider the piecewise cubic polynomial*

$$\mathbf{S}(x) = \begin{cases} \mathbf{S_0}(x) \text{ if } 0 \le x < 1 \\ \mathbf{S_1}(x) \text{ if } 1 \le x \le 2 \end{cases}$$

where

$$\begin{aligned} \mathbf{S_0}(x) &= a_0 + b_0 x + c_0 x^2 + d_0 x^3 \\ \mathbf{S_1}(x) &= a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3. \end{aligned}$$

(a) *Write all conditions a_i, b_i, c_i and d_i, $i = 0, 1$, must satisfy, so that $\mathbf{S}(x)$ is a cubic spline in $[0, 2]$.*
(b) *Determine a_i, b_i, c_i and d_i, $i = 0, 1$, so that $\mathbf{S}(x)$ is the clamped cubic spline interpolant of $f(x)$ in $[0, 2]$.*

SOLUTION:

(a) Since $\mathbf{S}(x)$ is defined in $[0, 2]$, the interior knot is $x = 1$.
A cubic spline must be continous and have continuous first and second derivatives:

Conditions to make $S(x)$ a cubic spline

- continuity on $x = 1$: $\mathbf{S_0}(1) = \mathbf{S_1}(1) \Rightarrow a_0 + b_0 + c_0 + d_0 = a_1$
- continuity of the 1^{st} derivative on $x = 1$: $\mathbf{S_0'}(1) = \mathbf{S_1'}(1) \Rightarrow b_0 + 2c_0 + 3d_0 = b_1$
- continuity of the 2^{nd} derivative on $x = 1$: $\mathbf{S_0''}(1) = \mathbf{S_1''}(1) \Rightarrow 2c_0 + 6d_0 = 2c_1$

(b) A clamped cubic spline interpolant of a function $f(x)$ must satisfy all conditions that a cubic spline satisfies (see (a)), and, in addition, it must interpolate the function at the knots, and the derivative of the function at the two end-points:

Interpolating conditions

- on $x = 0$: $\mathbf{S_0}(0) = f(0) \Rightarrow a_0 = f(0)$
- on $x = 1$: $\mathbf{S_1}(1) = f(1) \Rightarrow a_1 = f(1)$
- on $x = 2$: $\mathbf{S_1}(2) = f(2) \Rightarrow a_1 + b_1 + c_1 + d_1 = f(2)$

Clamped end-conditions

- on $x = 0$: $\mathbf{S_0'}(0) = f'(0) \Rightarrow b_0 = f'(0)$
- on $x = 2$: $\mathbf{S_1'}(2) = f'(2) \Rightarrow b_1 + 2c_1 + 3d_1 = f'(2)$

By solving the 8 equations arising from the conditions (3 continuity, 3 interpolating and 2 end-conditions) assuming $f(0), f(1), f(2), f'(0), f'(2)$ are given, we determine the 8 unknowns a_i, b_i, c_i and d_i, $i = 0, 1$.

Notes:

Interpolating condition $\mathbf{S_1}(1) = f(1)$ is equivalent to $\mathbf{S_0}(1) = f(1)$, since we have already enforced continuity of $\mathbf{S}$ at $x = 1$.

In this example, there are $n = 2$ subintervals and $n+1$ knots and data points, resulting in $4n = 8$ unknowns and equations, of which three are resolved by continuity conditions, another three by interpolating conditions, and the last two by the (clamped) end-conditions.

We have a adopted a simple technique for setting-up the equations to determine a_i, b_i, c_i and d_i, $i = 0, 1$.
This is, however, inefficient when the number of knots (and data) is large.
In such cases, instead of setting-up $4n$ equations, computations are done by considering that $\mathbf{S}$ is written in terms of cubic spline basis functions, that satisfy the continuity conditions by construction, then setting up the remaining $n + 3$ conditions (interpolating and end-conditions) in the form of a linear system, and computing the $n + 3$ coefficients (degrees of freedom) by solving the linear system. As mentioned in class, the resulting linear system (possibly after elimination of two unknowns through the two end-conditions) is tridiagonal, and, therefore, the cost of solving it is $O(n)$.

QUESTION 4 *How to experimentally determine the order of convergence of an interpolation method.*

SOLUTION:

We assume we have some software implementing the method on functions of our choice.
We pick a “test” function $f(x)$ that is infinitely differentiable in the domain of interpolation, e.g. $\exp(x), \sin(x)$, etc.
We pick a set of evaluation points v_i, $i = 0, \ldots, M$, in the domain of interpolation, for some M large.

For $n = n_0, 2n_0, \ldots, n_{\max}$
  sample $f(x)$ at points x_i, $i = 0, \ldots, n$,
  apply the method (run the software) to the set of data $\{(x_i, f(x_i))\}_{i=0}^{n}$
  this gives an interpolant $r_n(x)$ of $f(x)$
  compute $e_n = \max_{i=0}^{M} |f(v_i) - r_n(v_i)|$
endfor

A method with order ρ satisfies

$$e_n \approx \kappa n^{-\rho}, \tag{1}$$

for some κ independent of n, therefore

$$\rho = \log(e_n/e_{2n})/\log(2) \tag{2}$$

or, more generally, for $n \neq m$,

$$\rho = \log(e_n/e_m)/\log(m/n). \tag{3}$$

Thus, for each pair $(n, 2n)$ and the corresponding computed errors (e_n, e_{2n}), we can calculate ρ from (2). The ρ's we get for each pair, are usually slightly different to each other, since (1) is only an approximation. However, in most cases, we are interested only in approximately estimating ρ, thus small variations are acceptable.

Note: Taking logs on both sides of (1), we have

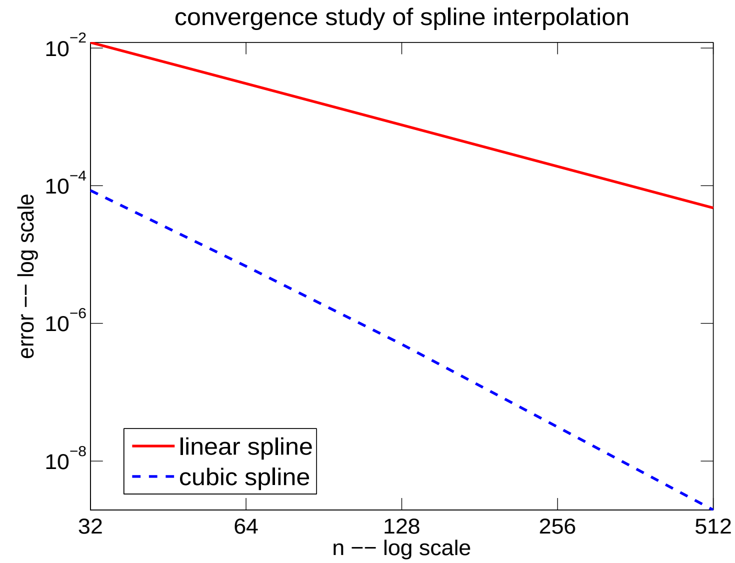
$$\log(e_n) = \log(\kappa) - \rho \log(n) \tag{4}$$

which indicates a linear relation between the logs of the errors and the logs of the grid sizes (n's) with slope $-\rho$.


```
a = 0; b = 10; M = 1000; v = linspace(a, b, M); u = sin(v);
fprintf('   n    linear     cubic\n');
for nn = 1:5
    n(nn) = 2^(nn+4);
    x = linspace(a, b, n(nn)+1); y = sin(x);
    e1(nn) = max(abs(interp1(x, y, v, 'linear') - u));
    e3(nn) = max(abs(spline(x, y, v) - u));
    fprintf('%4d %10.2e %10.2e\n', n(nn), e1(nn), e3(nn));
end
sz = 18;
hp = loglog(n, e1, 'r-', n, e3, 'b--');
set(hp, 'Markersize', sz-2, 'LineWidth', 2);
axis tight;
hax = gca;
set(hax, 'FontSize', sz-2, 'TickLength', [0.02 0.05])
set(hax, 'XTick', n); %, 'YTick', 10.^([-10:2:-1]))
hlx = xlabel('n -- log scale');
hly = ylabel('error -- log scale');
[hl, ho] = legend('linear spline', 'cubic spline', 0);
set(hl, 'FontSize', sz);
ht = title('convergence study of spline interpolation');
set(ht, 'FontSize', sz);
```



```
print -depsc tutpp.m.eps
```




```
  n    linear     cubic
 32   1.21e-02   8.52e-05
 64   3.05e-03   6.76e-06
128   7.62e-04   4.99e-07
256   1.91e-04   3.12e-08
512   4.77e-05   1.95e-09
```

Notes:

If we pick a test function which is not infinitely differentiable, depending on the differentiability of the function, we may (or may not) get a reduced order of convergence. We should get enough data (from enough n's), keeping in mind that data from very small n's may be unreliable, and at very large n's we may be reaching errors close to machine epsilon (therefore, contaminated with lots of round-off error which may dominate the interpolation error).

The number M should be fairly large, and not a small multiple of n.

The data points x_i do not necessarily have to be equidistant, as long as we always halve the stepsizes when doubling n. However, it is convenient if they are equidistant. Similarly, while v_i do not necessarily have to be strictly equidistant, it is preferable (and convenient) if they are. The v_i's should cover all the domain of interpolation.