Homework 6: Interpolation: Spline Interpolation (due on March 7)

---

1. In class, we interpolated the function $f(x) = \frac{1}{x}$ at the points $x = 2,4,5$ with the cubic spline that satisfied the *natural* boundary conditions

$$S''(a) = 0\,; \qquad (1)$$
$$S''(b) = 0 \qquad (2)$$

for $a = 2$ and $b = 5$.

(a) Change conditions (1-2) to the *clamped* boundary conditions

$$S'(a) = f'(a)\,; \qquad (3)$$
$$S'(b) = f'(b)\,, \qquad (4)$$

find the corresponding cubic spline and evaluate it at $x = 3$. Is the result more accurate than the one of the natural cubic spline interpolation?

Note: No programming is necessary, but a calculator might help.

(b) Prove that if $S(x)$ is a cubic spline that interpolates a function $f(x) \in C^2[a,b]$ at the knots $a = x_1 < x_2 < \cdots < x_n = b$ and satisfies the clamped boundary conditions (3-4), then

$$\int\limits_a^b \left[S''(x)\right]^2 dx \le \int\limits_a^b \left[f''(x)\right]^2 dx\,. \qquad (5)$$

Hint: Divide the interval $[a,b]$ into subintervals and use integration by parts in each subinterval.

2. The natural boundary conditions for a cubic spline lead to a system of linear equations with the tridiagonal matrix

$$\left[\begin{array}{ccccc} 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ h_2 & 2(h_2+h_3) & h_3 & \ddots & \vdots \\ 0 & h_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & h_{n-2} \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{array}\right], \qquad (6)$$

where $h_k = x_{k+1} - x_k$. The textbook shows that the clamped boundary conditions lead to the matrix

$$\begin{bmatrix} 2h_1 & h_1 & 0 & \cdots & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & & \vdots \\ 0 & h_2 & 2(h_2+h_3) & & & \vdots \\ \vdots & 0 & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & h_{n-2} & 0 \\ \vdots & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} . \tag{7}$$

Find the form of matrices that correspond to two other popular types of boundary conditions:

(a) "not a knot" conditions:

$$S_1'''(x_2) = S_2'''(x_2) ; \tag{8}$$
$$S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1}) . \tag{9}$$

(b) periodic conditions:

$$S_1'(x_1) = S_{n-1}'(x_n) ; \tag{10}$$
$$S_1''(x_1) = S_{n-1}''(x_n) . \tag{11}$$

Here $S_k(x)$ represent the spline function on the interval from x_k to x_{k+1}, $k = 1, 2, \ldots, n-1$. The periodic conditions are applied when $S(x_1) = S(x_n)$.

3. The algorithm for solving tridiagonal symmetric systems, presented in class, decomposes a symmetric tridiagonal matrix into a product of lower and upper bidiagonal matrices, as follows:

$$\begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ b_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & b_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & 0 & \cdots & \cdots & 0 \\ b_1 & \alpha_2 & \ddots & & \vdots \\ 0 & b_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & b_{n-1} & \alpha_n \end{bmatrix} \begin{bmatrix} 1 & \beta_1 & 0 & \cdots & 0 \\ 0 & 1 & \beta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \beta_{n-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

The algorithm for solving the linear system

$$\begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ b_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & b_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ \vdots \\ g_n \end{bmatrix}$$

is summarized below.

```
TRIDIAGONAL(a_1, a_2, ..., a_n, b_1, b_2, ..., b_{n-1}, g_1, g_2, ..., g_n)
α_1 ← a_1
for k ← 1, 2, ..., n − 1
do
    β_k ← b_k / α_k
    α_{k+1} ← a_{k+1} − b_k β_k
c_1 ← g_1
for k ← 2, 3, ..., n
do
    c_k ← g_k − β_{k−1} c_{k−1}
c_n ← c_n / α_n
for k ← n − 1, n − 2, ..., 1
do
    c_k ← c_k / α_k − β_k c_{k+1}
return c_1, c_2, ..., c_n
```

(a) The algorithm will fail (with division by zero) if any α_k is zero. Prove that, in the case of cubic spline interpolation with the natural boundary conditions,

$$\alpha_k > b_k > 0\,, \quad k = 1, 2, \ldots, n\,.$$

Hint: Start with $k = 1$ and use the method of mathematical induction.

(b) Design an alternative algorithm, where the tridiagonal matrix is factored into the product of upper and lower bidiagonal matrices, as follows:

$$\begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ b_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & b_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_1 & b_1 & 0 & \cdots & 0 \\ 0 & \hat{\alpha}_2 & b_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & b_{n-1} \\ 0 & \cdots & \cdots & 0 & \hat{\alpha}_n \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \hat{\beta}_1 & 1 & \ddots & & \vdots \\ 0 & \hat{\beta}_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \hat{\beta}_{n-1} & 1 \end{bmatrix}$$

4. (Programming) In this assignment, you can use your own implementation of the natural cubic spline algorithm or a library function. For your convenience, here is the algorithm summary:

```
NATURAL SPLINE COEFFICIENTS(x_1, x_2, ..., x_n, f_1, f_2, ..., f_n)
for k ← 1,2,...,n−1
do
    h_k ← x_{k+1} − x_k
    b_k ← (f_{k+1} − f_k) / h_k
for k ← 2,3,...,n−1
do
    a_k ← 2 (h_k + h_{k−1})
    g_k ← b_k − b_{k−1}
c_1 ← 0
c_n ← 0
c_2, c_3, ..., c_{n−1} ← TRIDIAGONAL(a_2, a_3, ..., a_{n−1}, h_2, h_3, ..., h_{n−2}, g_2, g_3, ..., g_{n−1})
for k ← 1,2,...,n−1
do
    d_k ← (c_{k+1} − c_k) / h_k
    b_k ← b_k − (2 c_k + c_{k+1}) h_k
    c_k ← 3 c_k
return b_1, b_2, ..., b_{n−1}, c_1, c_2, ..., c_{n−1}, d_1, d_2, ..., d_{n−1}
```

```
SPLINE EVALUATION(x, x_1, x_2, ..., x_n, f_1, f_2, ..., f_n, b_1, b_2, ..., b_{n−1}, c_1, c_2, ..., c_{n−1}, d_1, d_2, ..., d_{n−1})
for k ← n−1, n−2, ..., 1
do
    h ← x − x_k
    if h > 0
       then exit loop
S ← f_k + h (b_k + h (c_k + h d_k))
return S
```

Using your program, interpolate Runge's function $f(x) = \frac{1}{1+25x^2}$ on a set of n regularly spaced spline knots

$$x_k = -1 + \frac{2(k-1)}{n-1}, \quad k = 1, 2, \ldots, n .$$

Take $n = 5, 11, 21$ and compute the interpolation spline $S(x)$ and the error $f(x) - S(x)$ at 41 regularly spaced points. You can either plot the error or output it in a table. Does the interpolation accuracy increase with the number of knots?

5. (Programming)

The values in the table specify $\{x, y\}$ points on a curve $\{x(t), y(t)\}$.

x	2.5	1.3	-0.25	0.	0.25	-1.3	-2.5	-1.3	0.25	0.	-0.25	1.3	2.5
y	0.	-0.25	1.3	2.5	1.3	-0.25	0.	0.25	-1.3	-2.5	-1.3	0.25	0.

In this assignment, you will reconstruct the curve using cubic splines and interpolating independently $x(t)$ and $y(t)$. We don't know the values of t at the spline knots but can approximate them. For example, we can take t to represent the length along the curve and approximate it by the length of the linear segments:

$$t_1 = 0\,;$$
$$t_k = t_{k-1} + \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}\,, \quad k = 2, 3, \ldots, n\,.$$

Calculate spline coefficients for the natural cubic splines interpolating $x(t)$ and $y(t)$, then evaluate the splines at 100 regularly spaced points in the interval between t_1 and t_n and plot the curve.

What other boundary conditions would be appropriate in this example?