Project: Cubic spline interpolation

Tomasz Chwiej

27th January 2025

1 Introduction

For a given set of nodes (x_i) and tabulated functions values (y_i)

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\} \tag{1}$$

we want to make the polynomial interpolation with cubic splines assuming vanishing second order derivative of unknown function y(x) at both ends of interpolation interval

$$\alpha_2 = \beta_2 = \left.\frac{d^2y(x)}{dx^2}\right|_{x=x_0,x_n} = 0 \tag{2}$$

From the lecture we know this problem can be defined as tridiagonal system of linear equation

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ \mu_1 & 2 & \lambda_1 & \cdots & \cdots & 0 \\ 0 & \mu_2 & 2 & \cdots & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & & \mu_{n-1} & 2 & \lambda_{n-1} \\ 0 & \cdots & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ d_1 \\ \vdots \\ \vdots \\ d_{n-1} \\ \beta_2 \end{bmatrix} \tag{3}$$

where M_i stand for the values of 2-nd order derivative at nodes and need to be found while the other terms are calculated as follows

$$h_i = x_i - x_{i-1} \tag{4}$$

$$h_{i+1} = x_{i+1} - x_i \tag{5}$$

$$\lambda_i = \frac{h_{i+1}}{h_{i+1} + h_i} \tag{6}$$

$$\mu_i = 1 - \lambda_i \tag{7}$$

$$d_i = \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right) \tag{8}$$

When above SLE will be solved, elements M_i shall be used to form the cubic spline in i-th interval $([x_i, x_{i+1}])$

$$s_i(x) = M_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + M_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + A_{i+1}(x - x_i) + B_{i+1}, \quad i = 0, 1, 2, \ldots, n-1 \tag{9}$$

$$A_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}(M_{i+1} - M_i) \tag{10}$$

$$B_{i+1} = y_i - M_i \frac{h_{i+1}^2}{6} \tag{11}$$

The aim of this project is to perform the numerical calculations for this kind of interpolation for the test function.

2 Practical part

1. Write the computer program which calculates the cubic spline interpolation for test function

$$y(x) = \frac{1}{1+x^2}, \qquad x \in [-5, 5] \tag{12}$$

according to procedure sketched in Sec.1. First, define set of positions of equidistant nodes and function values, in calculations use $n = 10$. Second, declare tridiagonal matrix as three arrays and fill their elements, then solve the system of linear equations with routine ***LAPACKE_dgtsv()***. Compare obtained numerically second derivative values at nodes with calculated separately exact values of $y(x)$. Both sets of data may differ partly for central nodes, the results for the outermost nodes shall agree quite well. Write both second derivatives to file and save it. Finally make a figure which show the test function $y(x)$ and the cubic spline within interplation interval. Repeat calculations for larger number of interpolation nodes $n = 5, 50$ and make additional figures with new data.

2. At home prepare report. Besides the figures with function values, for each n plot also an interpolation error defined as difference between exact value $y(x)$ and cubic spline $s(x)$

$$\Delta_x = y(x) - s(x) \tag{13}$$

Make a comment on quality of cubic spline interpolation refering to the number of used nodes. Does the Runge effect occur or not?

3 Computational hints

1. In this project we work with tridiagonal matrix so to slove SLE efficently we shall use appropriate routine, in Lapack package recommended is ***LAPACKE_dgtsv()***

```
lapack_int LAPACKE_dgtsv (int matrix_layout , lapack_int N, lapack_int nrhs,
double * dl , double * d , double * du , double * b , lapack_int ldb );
```

where: **matrix_layout** indicates on row-major or column-major storage of elements in matrix (here: array **b**), N is the number of equations in SLE, **nrhs** - is a number of right hand side vectors, **dl** and **du** - are the $N-1$ element arrays filled with subdiagonal and superdiagonal matrix elements, respectively, **d** is an N element array representing the diagonal of matrix, **b** is an array of size $ldb*nrhs$ for column-major storage or $ldb*n$ for row-major storage, and, **ldb** is the leading dimension which takes the value: $ldb \geq n$ for column-major or $ldb \geq nrhs$ for row-major. In project we operate on single SLE, so we may simply use: $N = n+1$, $nrhs = 1$, $LAPACK\_COL\_MAJOR$ as $matrix\_layout$, $ldb = N$. On output the routine **nrhs** returns the solution in array **b**.

2. When filling the **dl**, **d** and **du** in e.g. *for-loop* remember that:

   - diagonal elements **d**: are filled from 1-st to (n-1) with value 2 while the 0-th and n-th elements have value 1
   - superdiagonal elements **du**: are filled from 1-st to (n-1) element with λ_i and 0-th element has value 0

- (IMPORTANT) subdiagonal elements **du**: elements from 0-th to $(n-2)$ index are filled with values from μ_1 to μ_{n-1}, indices in matrix elements are shifted by 1 with respect to the indices of array **dl**, the last (n-1) element of array has value 0

3. In order to draw the figure of cubic spline we must move from the leftmost to the rightmost node with M small steps $\delta = \frac{x_{max}-x_{min}}{M}$ and at each position x identify the index p of a segment we are in and cubic spline $s_p(x)$. Both can be done with following code

```
define the number of steps: M=199
        δ = (x_max − x_min)/M

for m=1 to M-1 by 1 do

        x ← x_min + δ ∗ m
        p ← −1 !indicates if any segment was not found

        !find segment
        for i=0 to n-1 by 1 do
                if( x ≥ x_i and x ≤ x_{i+1}) then
                        p=i
                        break
                end if
        end do

        !calculate value of spline s_p(x)
        if( p > 0 ) then
                !calculate
                i ← p
                h_{i+1} = (Eq.5)
                A_{i+1} = (Eq.10)
                B_{i+1} = (Eq.11)
                s_i(x) = (Eq.9)
        end if

end do
```

4 Example results

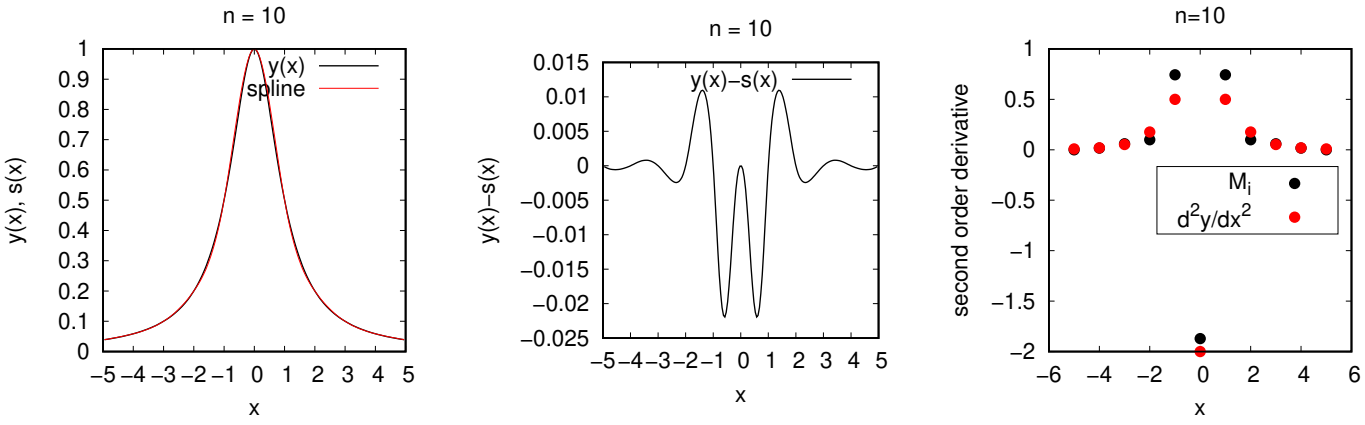

Figure 1: Cubic spline and test function (left), interpolation error (center) and second order derivative at nodes (right) calculated for $n = 10$.