Spline interpolation – additional tips

Task 1) We have to calculate two spline polynomials in two intervals. In the data file we have three nodes (x) and three function values (f(x)) in these nodes, e.g. in the file SI-Task1-Data1.txt we have:

	x	f(x)	
x_0	2	1	y_0
x_1	4	4	y_1
x_2	6	2	y_2

As a result we have two intervals: first $[x_0, x_1]$ and second $[x_1, x_2]$.
A general formula for the spline polynomial is: $y = Ay_j + By_{j+1} + Cy_j'' + Dy_{j+1}''$ (eq. (1) in the file spline_ins.pdf). Based on it we can write two separate formulas for particular intervals:
first: j=0, j+1=1 => $y1 = Ay_0 + By_1 + Cy_0'' + Dy_1''$,
second: j=1, j+1=2 => $y2 = Ay_1 + By_2 + Cy_1'' + Dy_2''$.
The assumption is that $y_0'' = y_2'' = 0$. As a result we do not need C when calculating y1, also we do not need D when calculating y2.
Two calculate y_1'' we should use eq. (18) in the file spline_ins.pdf with the following notations:
$x_{j-1}=x_0$, $x_j=x_1$, $x_{j+1}=x_2$, and $y_{j-1}=y_0$, $y_j=y_1$, $y_{j+1}=y_2$. Please note that eq. (18) simplifies significantly because $y_0'' = y_2'' = 0$, i.e. two components with multipliers 1/6 “disappear”.
General formulas for A, B, C, D are in the file spline_ins.pdf: eq. (2) – eq. (5). Based on them you can write separate formulas for particular intervals: using j=0, j+1=1 for the first interval, and j=1, j+1=2 for the second interval.
Finally, we obtain two spline polynomials, e.g.
$y1= -5x^3 + 1.2x^2 + 10x -11.05$, $y2= 3.7x^3 -x^2 + 11.5x +5$, which should be write in the report.
Calculations can be done manually, but it is very convenient to use Matlab:

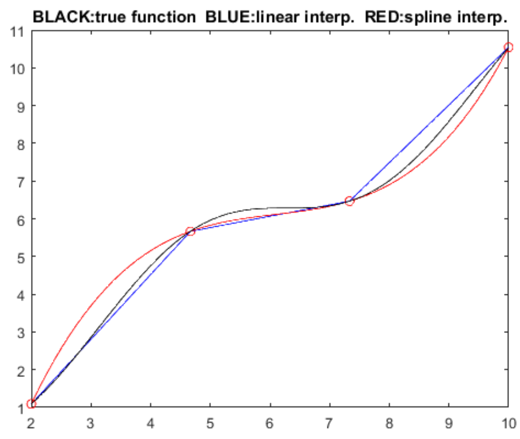
- Based on `solve` function you can solve the eq. (18) for y_1''.
  For example, the following code solves the equation: 1/3x-5 = –7x+10:
  ```
  syms x;
  my_formula=1/3*x-5==-7*x+10;
  x=solve(my_formula)
  ```
  Result: `x= 45/22`
- Based on `sym2poly` function you can find the coefficients at successive powers of x in the formulas of determined spline polynomials y1 and y2.
  For example, the following code finds the coefficients at successive powers of x, starting from x^3 and ending at free term, in the polynomial `7x-10x`^{2}`+(x-2)*(3-5x)-5x`^{3}`+1.2*x`:
  ```
  syms x;
  my_another_formula=7*x-10*x^2+(x-2)*(3-5*x)-5*x^3+1.2*x;
  sym2poly(my_another_formula)
  ```
  Result: `-5.0000   -15.0000   21.2000   -6.0000`

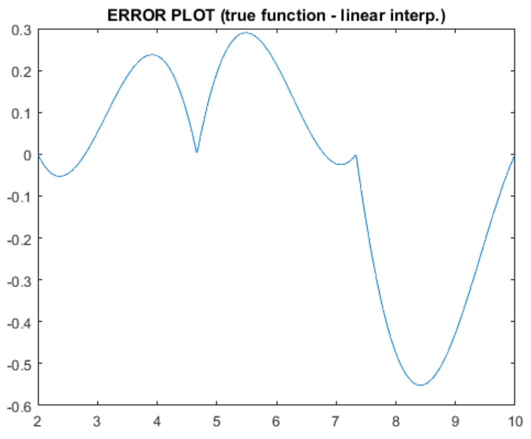
Note: the previous examples
y1= –5x^3 + 1.2x^2 + 10x –11.05, y2= 3.7x^3 –x^2 + 11.5x +5,
1/3x-5 = –7x+10,
7x-10x^2+(x-2)*(3-5x)-5x^3+1.2x
are not related to each other.

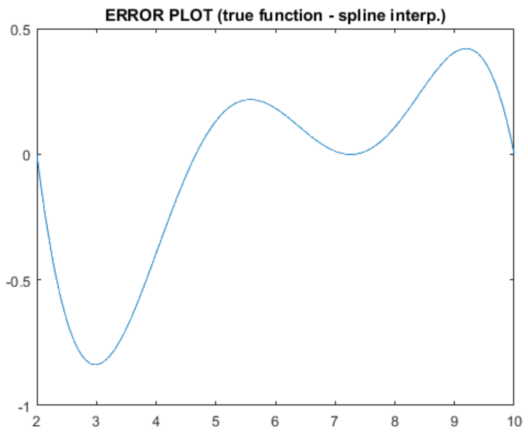
The results for the first dataset (SI-Task1-Data1.txt) are:
$y1 = -0.1563x^3 + 0.9375x^2 + 0.2500x - 2.0000$
$y2 = 0.1563x^3 - 2.8125x^2 + 15.2500x - 22.0000$

Task 2) We have to do linear and spline interpolation using Matlab function `interp1`. This time we do not have nodes and function values in these nodes directly – we have the function formula, a given range [a, b] and the number of interpolation nodes (i = 0, 1, 2, 3). Please note that we have **four nodes** numbered from 0 to 3. These nodes are equally distant in the range [a, b]. The Matlab function `interp1`:
`vq = interp1(x,v,xq,method);`
`vq –` new (interpolated) function values calculated by the `interp1`,
`x –` interpolation nodes, `v -` function values in the interpolation nodes,
`xq –` points where we want to have new (interpolated) function values,
`method –` interpolation method (we will use `'linear'` and `'spline'`, do not use `'cubic'`).
To obtain equally distant nodes (`x`) we can use Matlab function `linspace` (see in the Matlab help). Alter determining the interpolation nodes (`x`) we have to calculate true function values in these nodes (`v`), using the function formula. As `xq` please generate 500 equally distant points in the range [a, b] using `linspace` function.
After execution of the `interp1` function we will have new (interpolated) function values (`vq`).
Because we have to make also an error plots, we also must calculate (using the function formula) the true function values in `xq` points.

The results for the first dataset (SI-Task2-Data1.txt) are:



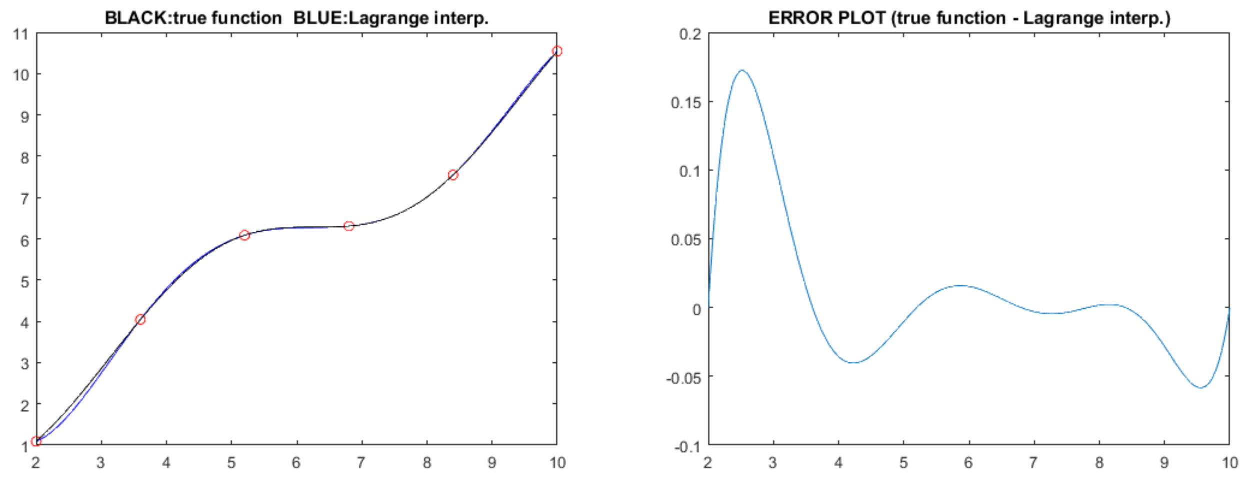




The maximum absolute error values are:
linear interpolation: 0.55176777, spline interpolation: 0.83760685.

Task 3) We have to do interpolation by means of the Lagrange polynomial, which formula is given on the first page in the file spline_lecture.pdf. Again, we have the function formula, a given range [a, b] and the number of interpolation nodes. Please pay attention to the correct number of nodes (e.g. 6). These nodes are equally distant in the range [a, b].
Lagrange polynomial formula (spline_lecture.pdf):
x – point where we want to have new (interpolated) function value,
x_i – interpolation node, $f(x_i)$ – function value in the interpolation node.
We are not interested in the formula of Lagrange polynomial – we have to calculate (and store in a vector) its value for each of the points where we want to have new (interpolated) function values. As these new points please generate 500 equally distant points in the range [a, b].
As in task 2, we have to plot original function and the interpolated function. We also have to make the error plot and calculate the maximum absolute error value.

The results for the first dataset (SI-Task3-Data1.txt) are:



The maximum absolute error value is: 0.17228556.

The report should include (in a single pdf file)
Number of the used dataset.
Task 1: formulas of the two spline polynomials in the form:
$y1 = -0.1563x^3 + 0.9375x^2 + 0.2500x - 2.0000$
$y2 = 0.1563x^3 - 2.8125x^2 + 15.2500x - 22.0000$
Task 2: plots of the original function and two interpolated functions, error plots (not absolute values), the values of maximum absolute error.
Task 3: plots of the original function and interpolated function, error plot (not absolute values), the value of maximum absolute error.