Numerical Analysis 1 – Class 9

Thursday, March 18th, 2021

Subjects covered

- Linear interpolation in 1D.
- Polynomial interpolation in 1D – classical Lagrange and Barycentric.
- Chebyshev polynomals and their application to interpolation.
- Runge phenomenon.
- 1D Splines.
- Bilinear interpolation with application to images.
- Meshes and interpolation in triangles.

Reading

- "Barycentric Lagrange Interpolation", J-P. Berrut and L. N. Trefethen. (Linked on Canvas).
- "Chebyshev Expansions", chapter 3 from "*Numerical Methods for Special Functions"*, Amparo Gil, Javier Segura, and Nico M. Temme. (Linked on Canvas).
- Kutz, Chapter 3

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. Please use Canvas to upload your submissions under the "Assignments" link for this problem set.

Problem 1

A useful twist to normal linear interpolation involves using so-called linear B-splines (also called basis-splines). The idea is to express a function defined on the interpolation points $[x_k, y_k]$ via an expansion

$$y(x) = \sum_{k=1}^{N} y_k B_k(x) \tag{1}$$

where y_k is the function value at the interpolation knots and $B_k(x)$ is a set of basis functions for expansion. Using B-splines, the interpolation function is a polynomial of degree *M* defined piecewise on the domain *x*. The special case of linear B-splines take a particularly simple form – they are the triangle-shaped "hat functions" shown below and defined by

$$B_k(x)=\begin{cases} 0 & \text{if} \quad x\le x_{k-1} \\ \dfrac{x-x_{k-1}}{x_k-x_{k-1}} & \text{if} \quad x_{k-1}<x\le x_k \\ \dfrac{x_{k+1}-x}{x_{k+1}-x_k} & \text{if} \quad x_k<x\le x_{k+1} \\ 0 & \text{if} \quad x_{k+1}<x \end{cases}$$

where x_k are the nodes. Observe that at the kth node, $B_k(x)$ take the value $B_k(x_k)=1$ and is zero at all other nodes. Note that at the ends of the interval only half of the hat function is used. In the plot shown below, the knot points are at $x_k=-3,-2,\cdots 2,3$.

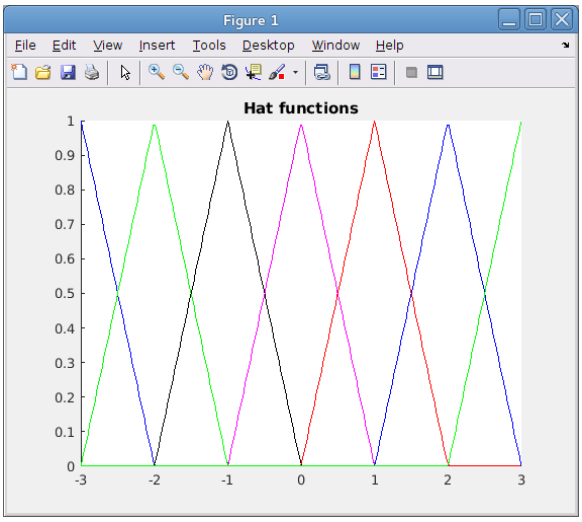
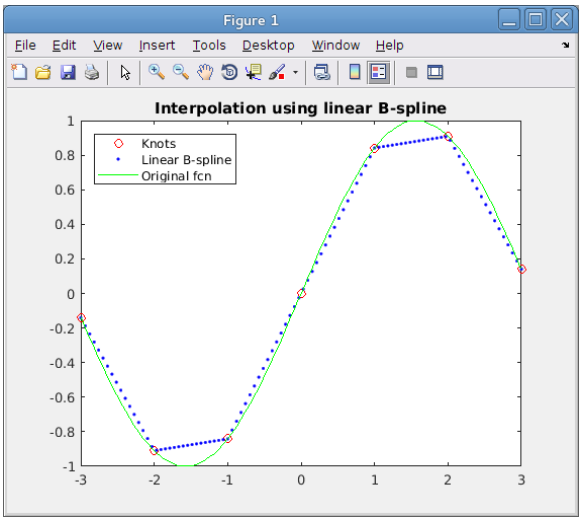


Figure 1: Each hat function carries a different color. Note that each hat function is one at its own node, and is zero at all other nodes.

An interesting feature of of this choice of basis function is that they are not mutually orthogonal, but since they are linearly independent, they form a valid basis for expansion. Moreover, since this basis is piecewise linear, the resulting expansion (1) results in a piecewise linear interpolation of the original function whose knot points are the pairs $[x_k, y_k]$. The goal of this problem is to explore interpolation using these functions. Please do the following:

- Please write a function *B*(*k*,*x*) which takes input an index *k* and an position *x* and returns the value of the *k*th hat function at that point. To test your function, I suggest you make a plot similar to the one above.
- Next write a function which, when given a position *x* and the knot pairs $[x_k, y_k]$ implements the sum (1).
- Test your function by interpolating a sine wave in the domain [-3, 3]. My result, using 7 knot points, is shown below.



Problem 2

In my class discussion of bilinear interpolation, I asserted that interpolation using the graphical "line method" gave equivalent results to the "matrix method". I then showed two Matlab plots of an interpolated function created using the two methods. The plots were identical.

Of course, "Proof by Matlab" is not a real proof. Using pencil and paper, start with the "line method" and show that the interpolation function obtained is of the form

$$z(x,y)=A+Bx+Cy+Dxy$$

and derive expressions for the coefficients A, B, C, and D. This is a pencil-and-paper exercise -- please hand in your derivation.

Problem 3

In class I advocated for using Chebyshev nodes as interpolation (sample) points if you wanted to interpolate your data with better fidelity. This problem aims to convince you of this.

Consider interpolating the function

$$f(x)=\sin(2x)e^{x} \qquad (2)$$

defined on the interval $x=[0,5]$. Please write programs which do the following:

1. In your first program, start by ploting the original function (2) on its domain of definition.
2. Next, sample the function at *N* Chebyshev nodes. Call your samples $[x_i, y_i], i\in 1\ldots N$.
3. Plot your samples on the same plot as in step 1.
4. Interpolate your sampled data using the Lagrange polynomial of order *N*. Interpolate it on a a fine grid of points and plot it on the same plot as in step 1.
5. In a second program, plot the original function (2) on its domain of definition, $x=[0,5]$.
6. Now sample the function at *N* equispaced points. Again call your samples $[x_i, y_i], i\in 1\ldots N$.

7. Plot your samples on the same plot as in step 5.
8. Interpolate your sampled data using the Lagrange polynomial of order *N*. Interpolate it on a a fine grid of points and plot it on the same plot as in step 5.

At this point you should have two plots showing the effectiveness of interpolating at Chebyshev points vs. at equispaced points. My results are shown below for $N = 19$. Note how the equidistant interpolation "goes crazy" at the left hand side of the plot.

Regarding testing, it's enough to get the plots, but I recommend you check your interpolations for varying *N* – you should see their accuracy improve for increasing *N*.