# MATH 353: Engineering Mathematics III – Section 012

Spring 2013 (F.–J. Sayas) Lab # 5 March 8

Open Matlab and *move to the Desktop or to a folder where you can find your work* at the end of the session. Type these lines

```
>> diary myworkMarch8
>> format long
>> format compact
```

Download the functions `divideddiff.m` and `nested.m` from my website.

1. What do the following lines do?

```
>> linspace(1,3,7)
>> linspace(0,4,10);
```

2. Type up the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & -4 & 0 \\ 2 & 3 & 1 \\ 2 & 1 & 1 \end{bmatrix}.$$

(Remember: we write by rows, with blanks between elements and using semicolons to change rows.) How would you select the complete 3rd row? And the complete 3rd column?

3. If we define the following functions

```
>> f = @(x) x.^2-2*x;
>> g = @(x) f(x.^2);
```

What function is $g$ exactly? (Write down the mathematical expression.)

## Interpolation using Newton's formula

4. Consider the points

$$(0,1), \quad (1,3), \quad (2,4), \quad (3,6), \quad (4,-2), \quad (5,0).$$

Write a script that does all of the following steps:

- Compute the divided differences with `divideddiff`.
- Create a large list of points `xx` in the interval $[0,5]$ and evaluate the interpolation polynomial at `xx` using `nested`.
- Plot the points where we interpolate with circular markers. Plot the interpolation polynomial on top of the points.

Read this! Ask if you don't understand.

We are given a collection of points:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_{n+1}, y_{n+1})$$

with the $x_i$ points given in increasing order (this is a requirement here!)

$$x_1 < x_2 < x_3 < \ldots < x_n < x_{n+1}.$$

A **cubic spline** function with **knots or breakpoints** at the points $x_1, \ldots, x_{n+1}$ is a function $S : [x_1, x_{n+1}] \to \mathbb{R}$ such that

(a) It is a cubic polynomial on each interval $(x_i, x_{i+1})$

(b) It is continuous, with continuous derivative and continuous second derivative.

In principle, with the first rule, a cubic spline is determined by 4 parameters (the coefficients of the polynomial) on each subinterval. There are $n$ subintervals and therefore there are $4\,n$ parameters to determine. If we now look at the rule (b), we have to impose the conditions

$$S(x_i^-) = S(x_i^+), \qquad S'(x_i^-) = S'(x_i^+), \qquad S''(x_i^-) = S''(x_i^+) \qquad i = 2, \ldots, n.$$

(Why?) This means that we have $4n - 3(n-1) = n+3$ parameters to determine. We want this spline to interpolate our points, that is, we want

$$S(x_i) = y_i \qquad i = 1, \ldots, n+1,$$

which are $n+1$ additional conditions on the coefficients. This leaves us with

$$4n - 3(n-1) - (n+1) = 2 \qquad \text{free parameters.}$$

The world of splines is actually quite interesting and there are many options to fix these two missing parameters. One of them is called the **not-a-knot** option, which imposes two more conditions

$$S'''(x_2^-) = S'''(x_2^+), \qquad S'''(x_n^-) = S'''(x_n^+).$$

Now here's a tricky observation. These last conditions (added to the previous continuity conditions in $x_2$ and in $x_n$) imply that in $(x_1, x_2)$ and $(x_2, x_3)$, $S$ is the same polynomial (that is, its coefficients are the same). Likewise, in $(x_{n-1}, x_n)$ and $(x_n, x_{n+1})$, $S$ is also the same polynomial. Can you figure out why? In any case, this implies that the points $x_2$ and $x_n$ are not really knots of the not-a-knot spline, given the fact that $S$ does not break across these points.

Interpolating with cubic splines requires writing down a system of linear equations to compute the coefficients of the polynomials on each of the subintervals. Once we have the coefficients, to evaluate the spline, we have to determine in what subinterval we are and then we evaluate the corresponding polynomial. This time we are going to let Matlab do the hard work.

5. Run the following lines of code:

```
>> x=[0 1 2 3 4 5];
>> f = @(x) x.^4-cos(pi/3*x);
>> S=spline(x,f(x))     % S is a data structure with many things in it
S =
      form: 'pp'
    breaks: [0 1 2 3 4 5]
     coefs: [5x4 double]
    pieces: 5
     order: 4
       dim: 1
>> format short
>> A = S.coefs           % S.coefs are the coefficients of the spline
A =
    5.0222   -7.8167    4.2944   -1.0000
    5.0222    7.2500    3.7278    0.5000
    9.8889   22.3167   33.2944   16.5000
   14.9222   51.9833  107.5944   82.0000
   14.9222   96.7500  256.3278  256.5000
```

The way Matlab gives the coefficients is the following. To represent the cubic polynomial in the interval $(x_i, x_{i+1})$ the coefficients are given in reverse order in the $i$-th row of the matrix `A` and thinking of the representation:

$$S(x) = a_{i,1}(x - x_i)^3 + a_{i,2}(x - x_i)^2 + a_{i,3}(x - x_i) + a_{i,4} \qquad \text{in } (x_i, x_{i+1})$$

With this format, it is not obvious why the first and second rows are coefficients corresponding to the same polynomial. Also the last and last but one rows correspond to exactly the same polynomial that is written in a different way.

6. Instead of using the coefficients to evaluate the spline, we can do everything at the same time. If `x, y` are the interpolation points and `xx` is a large list of numbers between the first and last of the interpolation points, then `spline(x,y,xx)` evalutes the spline at all the points. Write a short script doing the following:

   - Define the same `x` and `f` as in Exercise 5.
   - Plot the interpolation points `x, f(x)` with circular markers.
   - On top of those points, plot the interpolation spline.