Numerical Methods
Bachelor in Physics

# Practicum 3: Interpolation and splines

*Contact: jeroen.mulkers@uantwerpen.be U.305*

1. Consider the function
$$f(x) = \frac{1}{1+x^2}.$$
Use the command polyfit to determine the 10th order polynomial which interpolates the data points at $x = -5, -4, ..., 4, 5$. Plot the function, the data points and the interpolating polynomial. Next, use the command spline to determine the spline interpolation. Plot also the spline interpolation. Make sure to plot both interpolations on a finer grid than the given $x$-values.

2. Make a drawing of your hand. Start with
```
figure
[x,y]=ginput;
```
Place your hand on the screen of your computer and add points by clicking your mouse along the contours of your hand. Stop the input through ginput by pressing the `enter`-key. Consider `x` and `y` as data points from two independent functions. Interpolate both functions with a spline and a polynomial interpolation. Make a plot of your interpolated hand contour.

3. So far, we only have used the spline command with three arguments: the $x$ and $y$ values of the data points which need to be fitted, and the $x$ values at which we want to know the values of the spline interpolation. Have a look at the output if only the first two arguments are given:
```
≫ spdata = spline(x,y)
```
The matrix coefs contains the coefficients of the $n-1$ polynomials
$$S_j(x) = a_j(x-x_j)^3 + b_j(x-x_j)^2 + c_j(x-x_j) + d_j, \ \ x_j \leq x \leq x_{j+1}.$$
You can retrieve them by using following command:

```
≫ [breaks,spcoefs,pieces,order,dim] = unmkpp(spdata);
```

spcoefs is a matrix with the spline coefficients in following order: $a_j, b_j, c_j, d_j$. Print the coefficients on the screen. Why is $a_1 = a_2$ and $a_{n-2} = a_{n-1}$? Why is $d_j = y_j$? You can read about spline interpolations in the lecture notes.

It is possible to construct a piecewise polynomial with the commands mkpp and ppval. As an example, we will construct the following piecewise polynomial:

$$y = 2(x-1) - 4 \text{ for } 1 \leq x \leq 2;$$
$$y = 3(x-2) + 6 \text{ for } 2 \leq x \leq 6.$$

(Note the $x - x_{\text{begin interval}}$ in both equations.) We start by creating a vector with the intervals

```
≫ breaks = [1 2 6]
```

and a matrix with the coefficients

```
≫ coef = [ 2 -4 ; 3 6 ]
```

Construct and visualize the piecewise polynomial by executing following commands:

```
≫ lindata = mkpp(breaks,coef)
≫ xi = [1:0.1:6];
≫ yi = ppval(lindata, xi);
≫ plot(xi,yi)
```

Consider the following data set:

| $x$ | 1.0 | 2.3 | 3.1 | 4.0 | 5.2 | 5.9 | 6.7 |
|---|---|---|---|---|---|---|---|
| $y$ | 1.7 | 2.8 | 3.6 | 4.5 | 3.4 | 3.1 | 3.1 |

Use the above method to calculate (and plot) the derivative of the spline interpolation of this data set. Make use of the commands polyder and polyval.

4. **Trigonometric interpolation**

Rewriting a function as a linear combination of sine and cosine functions, decomposes the function in its frequency components (analogous to a prism which splits a light beam in its different colors). The coefficients of the trigonometric basis functions tell us 'how much' each frequency is present in the given function. Furthermore, it is much easier to work with functions in the frequency domain (instead of the real domain) in many applications, e.g. in signal processing or when solving differential equations. In this exercise, we will see that calculating a **trigonometric interpolation** of a sampled function is remarkably fast and efficient with the 'fast fourier transform' algorithm.

Assume that $f(t)$ is a periodic function with period $T$ and we want to interpolate $n$ data points $(t_0, f_0), \dots, (t_{n-1}, f_{n-1})$ with $f_k = f(t_k)$ and

$$t_k = k\frac{T}{n}, \text{ met } k = 0 : n-1.$$

Since the data is periodic, it makes more sense to interpolate the data points with periodic functions instead of polynomials. That is why we will use an interpolating function which is a linear combination of sine and cosine functions instead of $1, x, x^2, \dots$

For an integer $j$, the functions $\cos(2\pi jt/T)$ and $\sin(2\pi jt/T)$ have the same period $T$ as $f$, so let us write

$$F(t) = \frac{1}{n}\left(\sum_{j=0}^{m}\left(A_j\cos\left(\frac{2\pi j}{T}t\right) + B_j\sin\left(\frac{2\pi j}{T}t\right)\right)\right),$$

with $n = 2m$. Now we want to determine the coefficients $A_k$ and $B_k$ in such a way that $F(t_k) = f_k$ for $k = 0 : n-1$. This yields a system of $n$ equations to determine $2(m+1) = 2m+2$ unknowns. Note however, that $B_0$ and $B_m$ are irrelevant since $\sin(2\pi jt_k/T) = 0$ for $j = 0$ or $j = m$. So, we only have to determine the coefficients $A_0, \dots, A_m$ and $B_1, \dots, B_{m-1}$ in such a way that if

$$\begin{aligned} F(t) &= \frac{1}{n}\left(A_0 + \sum_{j=1}^{m-1}\left[A_j\cos\left(\frac{2\pi j}{T}t\right) + B_j\sin\left(\frac{2\pi j}{T}t\right)\right]\right. \\ &\qquad \left. + A_m\cos\left(\frac{2\pi m}{T}t\right)\right), \end{aligned} \tag{1}$$

then $F(t_k) = f_k$ for $k = 0 : n-1$.

With the substitutions

$$Y_0 = A_0, Y_m = A_m, Y_j = (A_j - iB_j)/2, Y_{n-j} = (A_j + iB_j)/2,$$

with $j = 1, 2, \ldots, n-1$, we can rewrite the interpolating series as

$$F(t) = \frac{1}{n} \sum_{j=0}^{n-1} Y_j e^{i2\pi jt/T}.$$

The coefficients $Y_k$ are complex. They can be determined by solving following system of equations:

$$F(t_k) = f_k = \frac{1}{n} \sum_{j=0}^{n-1} Y_j e^{i2\pi jk/n} = \frac{1}{n} \sum_{j=0}^{n-1} Y_j \omega_n^{-jk}, \text{ for } k = 0 : n-1$$

with $\omega_n = e^{-2\pi i/n}$. This becomes in matrix notation

$$\mathbf{f} = \frac{1}{n} \Omega_n^{-1} \mathbf{Y}, \tag{2}$$

with $[\Omega_n^{-1}]_{jk} = \omega_n^{-jk}$. E.g. for $n = 4$ the matrix looks like

$$\Omega_4^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}.$$

Note that $\Omega_n^{-1}$ is a Vandermonde matrix.

It is important to note that the inverse matrix is simple and can be written down explicitly:

$$\frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} =$$

$$\frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & -i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This gives $[\Omega_n^{-1}]^{-1} = \Omega_n/n$ with $[\Omega_n]_{jk} = \omega_n^{jk}$. It is now easy to calculate the coefficients $Y_k$:

$$Y_k = \sum_{j=0}^{n-1} f_j \omega_n^{jk}, \text{ for } k = 0 : n-1 \text{ of } \mathbf{Y} = \Omega_n \mathbf{f}. \tag{3}$$

This is called the disecrete Fourier transform (DFT). Instead of solving the system (2) ($O(n^3)$ operations), we calculate sum (3), which is nothing more than a matrix multiplication ($O(n^2)$ operations). The purpose of this section is to show that there exists an algorithm, the 'fast Fourier transform', which only needs $O(n \log_2 n)$ operations.

The DFT (3) decomposes a given input $\mathbf{f}$ in its frequencies components given by $\mathbf{Y}$. There are two components which are especially important; $Y_0$ corrsponding with the zero frequency (the DC component, a constant function) and $Y_{n/2}$ corresponding with the *Nyquist freqency*, the largest representable frequency for a given sampling rate. From (1), we can conlude that the Nyquist frequency is equal to $\nu_{max} = m/T = n/(2T)$. It is important to sample a signal with a high enough rate, such that there are at least two sample points per cycle of the highest frequency component. If this is not the case, e.g. the signal has a frequency component with $\nu_1 > \nu_{max}$, then this will yield an unwanted contribution to the DFT spectra at frequency $\nu_2$ for which it holds true that $\nu_1 + \nu_2 = 2\nu_{max}l$ or $\nu_1 - \nu_2 = 2\nu_{max}l$, with iteger $l$.

**The 'Fast Fourier Transform' algorithm**

We illustrate the 'Fast Fourier Transform' (FFT) algorithm for $n = 4$. The first Fourier matrices $\Omega_n$ are:

$$\Omega_1 = 1, \qquad \Omega_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \Omega_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

Define permutation matrix $P_4$ as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the diagonal matrix $D_2$ as

$$D_2 = \mathrm{diag}(1, \omega_4) = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

We can then write

$$\begin{aligned}
\mathbf{Y} &= \Omega_4 P_4 P_4^{-1}\mathbf{f} \\
&= \left[\begin{array}{rr|rr} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array}\right] \begin{bmatrix} f_0 \\ f_2 \\ f_1 \\ f_3 \end{bmatrix} \\
&= \begin{bmatrix} \Omega_2 & D_2\Omega_2 \\ \Omega_2 & -D_2\Omega_2 \end{bmatrix} \begin{bmatrix} \mathbf{f}(0:2:3) \\ \mathbf{f}(1:2:3) \end{bmatrix} \\
&= \begin{bmatrix} I & D_2 \\ I & -D_2 \end{bmatrix} \begin{bmatrix} \Omega_2\mathbf{f}(0:2:3) \\ \Omega_2\mathbf{f}(1:2:3) \end{bmatrix}.
\end{aligned}$$

Thus, we can split matrix $\Omega_4$ in 4 blocks, which are scaled versions of $\Omega_2$. In this way, we can easily derive the 4-point $\mathbf{Y} = \Omega_4\mathbf{f}$ from the two-point DFTs $Y_T = \Omega_2\mathbf{f}(0:2:3)$ and $Y_B = \Omega_2\mathbf{f}(1:2:3)$:

$$\begin{aligned}
\mathbf{Y}(0:1) &= Y_T + d_2. * Y_B \\
\mathbf{Y}(2:3) &= Y_T - d_2. * Y_B,
\end{aligned}$$

with weight factors $d_2 = [1 - i]^T$. To conclude, in order to derive the FFT for $n = 4$, we only need to calculate the FFT for $n = 2$ and multiply it with $d_2$. We can go on, and calculate the FFT for $n = 1$ (the single-point FFT of a number, is the number itself).

In general, if $n = 2m$, then $\mathbf{Y} = \Omega_n\mathbf{f}$ is given by

$$\begin{aligned}
\mathbf{Y}(0:m-1) &= \mathbf{Y}_T + d. * \mathbf{Y}_B \\
\mathbf{Y}(m:n-1) &= \mathbf{Y}_T - d. * \mathbf{Y}_B,
\end{aligned}$$

with $\mathbf{Y}_T = \Omega_m\mathbf{f}(0:2:n-1)$, $\mathbf{Y}_B = \Omega_m\mathbf{f}(1:2:n-1)$ and $d = [1\omega_n \cdots \omega_n^{m-1}]$. For $n = 2^t$, we can perform the decomposition until $n = 1$. For this reason, one usally uses $t^2$ sampled data points. This recursive method is called the FFT algorithm. Since there are $\log_2 n$ recursion levels, and each level takes $O(n)$ operations, the total number of operations is of the order $O(n\log_2 n)$. Of course, the FFT algorithm can also be used to calculate the inverse DFT (2) efficiently.

With the command fft you can calculate the FFT in MATLAB. The use of this command is demonstrated in following examples.

**Example 1** We want to calculate the DFT of the function $y = 0.5 + 2\sin(2\pi\nu_1 t) + \cos(2\pi\nu_2 t)$, with $\nu_1 = 3.125$ Hz and $\nu_2 = 6.25$ Hz. We will use 64 equidistant data points sampled over a period of 3.2 sec. The script DemoFFT1.m does exactly this. Study and execute the script. Make sure that you understand every line. Consult the documentation of the command bar. Give an answer to the following questions.

(a) What is shown in figure 1, 2 and 3?

(b) What is the meaning of real(Yss) and imag(Yss)?

(c) Explain the line Yss(1)=Yss(1)/2; Yss(nt/2+1)=Yss(nt/2+1)/2;.

(d) Why are there 64 $k$ values in figure 2, and only 33 frequency values in figure 3.

Let us play with some of the parameters (the frequencies, sample time and number of samples)

(a) Change $\nu_2$ in 7 Hz. What happens to the frequency spectrum? How can you explain this? How does the spectrum change when $\nu_2 = 7.5$ Hz.

(b) We will now vary the total number of sample points (choose $\nu_1 = 3.125$ Hz and $\nu_2 = 7.5$ Hz). The implementation of the FFT algorithm in MATLAB allows an arbitrary number of sample points. However, using a power of two will be the most efficient. What happens to the $\nu_2$ peak when you change the number of samples to 50,48,46,...?

(c) Use again 64 sample points, and choose $\nu_2 = 105$. Explain the large peak at $\nu = 5$ in the spectrum.

(d) In order to get a peak at $\nu = 105$, we could lower the sample time to $T = 0.2$ s. What is the disadventage of doing this? What can we do instead?

**Example 2** Now, we will calculate the spectrum of the function

$$y = 0.5 + 0.2\sin(2\pi\nu_1 t) + 0.35\cos(2\pi\nu_2 t) + \textbf{noise},$$

with $\nu_1 = 20$ Hz and $\nu_2 = 50$ Hz. We will use 512 sample points in a timeinterval of 2 s. The noise has a normal distribution with mean 0 and standard deviation 0.5. Script DemoFFT2 shows the spectrum without the noise. Add the noise, and study the change in the spectrum.

**Example 3** Calculate the DFT of $y = 32\sin^5(2\pi\nu t)$ with $\nu = 30$ Hz by modifying the script DemoFFT1. Use 512 data points sampled over 1 s. Using the DFT coefficients, estimate the values $a_0, a_1$ and $a_2$ in the following approximation

$$y = 32\sin^5(2\pi\nu t) \approx a_0\sin(2\pi\nu t) + a_1\sin(2\pi(3\nu)t) + a_2\sin(2\pi(5\nu)t).$$