Assignments III

Exercise 1 *Write a* MATLAB*-function which fits a linear combination of exponential functions to data.*

The input consists of the data, an $(m \times 2)$ matrix, and a vector of initial guesses for the exponents. The data are fit with a function of the form

$$f(x) = C_1 e^{\lambda_1 x} + C_2 e^{\lambda_2 x} + \ldots + C_n e^{\lambda_n x}.$$

The number of exponents n follows from the length of vector of the initial guesses given as input to the function. The function looks for the exponents and constants such that f is the best fit of the data in the sense of least squares. The output are two vectors (one with the calculated optimal exponents λ_i and one with the calculated optimal constants C_i) as well as the residue. Moreover, the function produces a picture of the data and the fit f.
N.B.: you are *not* supposed to use the optimisation function `lsqnonlin` even if it is available on your computer. On the other hand, you can use `fminsearch`.

1. For fixed $\lambda = (\lambda_1, \ldots, \lambda_n)$ you should use the least square method to find the best coefficients $C_i = C_i(\lambda)$ and calculate the corresponding residue $R = R(\lambda)$.
2. Using `fminsearch` you can subsequently minimize $R(\lambda)$.
3. You can, among others, test you function on the data in the file `expo-examples.mat` on the website. After downloading, `load expo-examples` gives you three data sets `data1`, `data2` and `data3`.
4. Estimate the values of the exponents which are "hidden" in these signals. You also still have to determine *how many* exponents are present.

Exercise 2 *Write a* MATLAB*-function which determines the center of mass of a domain enclosed by an interpolated curve.*

This function has no input or output variables. It opens a figure with a big rectangle on the screen (use for example the property `'position'` of a figure). The function then asks the user to put his hand on the rectangle and outline the circumference of his hand with a few dozen mouse clicks. This information can be retrieved with the function `ginput`. Next, the function interpolates these points with splines to obtain a nice (hand) shape and calculates (approximately) the center of mass. Finally, the function produces a picture of the input points, the interpolated curve, and the area inside the curve, with a clear mark at the center of mass.

1. Give the user instructions on the screen.
2. The function should ignore subsequent clicks on the same point (i.e. a double click of the mouse).
3. Interpolation should be done using pseudo arc-length parametrization.
4. The center of mass of a set $\Omega \subset \mathbb{R}^2$ has coordinates $(\int_\Omega x)/(\int_\Omega 1)$ and $(\int_\Omega y)/(\int_\Omega 1)$.
5. A nice way to calculate, for example, the area of an enclosed domain is via Green's theorem: $\oint_{\partial\Omega}[f_1\, dx + f_2\, dy] = \int_\Omega [\frac{\partial f_2}{\partial x} - \frac{\partial f_1}{\partial y}]\, dxdy$.

Exercise 3 *Write a* MATLAB*-function which calculates both the discrete Fourier transform of a vector and its inverse Fourier transform.*

The input variables are a vector of length 2^n and a variable which indicates whether the normal or inverse transform needs to be calculated.

1. Clearly, you are *not* allowed to use the built-in functions like `fft` and `ifft`.
2. You may assume that the length of the input vector is a power of 2.
3. The speed is an important aspect of the Fourier transform and, consequently, of this exercise. Hence, test your program also for large n. The faster the algorithm, the better (it should obviously be correct as well).
4. Finally, after writing your program, define two (approximately) periodic functions f and g, and show how you can determine numerically
   - their convolution $f * g$, and
   - the first and second derivatives f' and f''

   using the discrete Fourier transforms.