Assignment #2

This assignment is due on April 16th one hour before class via email to **`christian.wallraven+EMS2019@gmail.com`**.

If you are done with the assignment, make one zip-file of the `assignment2` directory and call this zip-file `STUDENTID1_STUDENTID2_STUDENTID3_A2.zip` (e.g.: `2016010000_2017010001_A2.zip` for a team consisting of two students or `2016010000_2017010001_2017010002_A2.zip` for a three-student team). The order of the IDs does not matter, but the correctness of the IDs does! **Please double-check that the name of the file is correct!!**

Please make sure to comment the code, so that I can understand what it does. Uncommented code will reduce your points!

Also, please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!

Part1 Numerical Derivation (20 points):

Write a function that implements numerical derivation on a known function with the following definition:

```
function [derivative]=derive(function_handle,x_values,h)
```

Its input consists of a function handle and an array of x-values. The third argument h should be optional (check the value nargin!) and if not supplied should be set to h=1e-5.

The function should use the standard numerical approximation of the derivative:

$f'(x) = \frac{f(x+h)-f(x)}{h}$, with h very small.

Add a script testDerivative.m in which you plot the SSE for all points summed up for h=10^[-1:-1:-14] and x_values=[.001:.01:2] for the following functions and their HAND-DERIVED ☺ derivatives [use a semilog plot like in class – take inspiration from the class jupyter notebook as it has a lot of code that is similar to this assignment part]:

$f(x) = x^2$

$f(x) = \cos(x)$

$$f(x) = \frac{\sin(x)}{x}$$

Answer the following questions as comments in the code:

1) Why do the error curves first go down and then go up again?
2) Why do you think the error curves for the three functions are different?

Part2 Numerical Integration (20 points):

Write a function that implements numerical integration on a known function with the following definition:

```
function [integral]=integrate(function_handle,x_values,h,type)
```

Its input consists of a function handle and an array of x-values. The third argument h should be optional (check the value nargin!) and if not supplied should be set to h=1e-5. The fourth argument type is a string (or char array) with possible values 'trapezoid' and 'midpoint' – it should also be optional and if not supplied should be set to 'trapezoid'.

The function should evaluate the integral for each PAIR of SUCCESSIVE x-values and so will return `length(x_values)-1` values. In order to do so, you need to make sure that h is smaller than the difference between successive x_values. Insert error-handling to guarantee that $h < average_i(x_{i+1} - x_i)$ and throw an error if it doesn't!

If `type` is `'trapezoid'` the function needs to implement trapezoidal approximation of the integral values, if `type` is `'midpoint'`, the function needs to implement the midpoint method (look it up!).

Add a script `testIntegrate.m` in which you plot the SSE for all points summed up for `h=10^[-2:-1:-7]` and `x_values=[.001:.1:2]` for $f(x) = \mathrm{x}^2$ for both trapezoid and midpoint integration. Be VERY careful about the TRUE integral here and remember that we can only compare DEFINITE integrals in this way.

Answer the following questions as comments in the code:

1) What happens to the error curves as h becomes smaller and why?
2) Which integration method is better?

Part3 Numerical Integration of sinc (20 points):

Let's compare two ways to integrate the sinc-function $f(x) = \frac{\sin(x)}{x}$ numerically as a **definite** integral from 0.5 to 15.

The first way uses the integrate function from Part2.

The second way uses the fact that we can approximate the sin-function with a Taylor series:

$$\sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1}$$

So, then, we simply divide the series through by x and integrate the result term by term and we end up with the approximate Taylor series for the integral! Implement this in a function

```
function [integral]=SI(x,n)
```

where the first input should be clear. The second input is the degree of the Taylor approximation and should be optional with a default argument of `n=10`.

In a script `integrateSinc.m` implement calls that compare the values of both ways for

$\int_{0.5}^{15} \frac{\sin(x)}{x} dx$ for `h=10^[-2:-1:-7]` and `n=[10:2:20]` in a scatter plot.

Which method would you choose and why?