Alessandro Sciarra: sciarra@itp.uni-frankfurt.de

Exercise sheet 6

To be corrected in tutorials in the week from 25.11 to 29.11.2019

Exercise 1 [*Numerical derivative*]

Given a function $f(x)$, calculating its derivative is usually a quite straightforward task and we rarely need to do it numerically. Nevertheless, if $f(x)$ can only be evaluated numerically, a non-analytical approach is required. The simplest way is using finite difference formulas, which exists in several variants[1]. Symmetric ones, also known as central, are the most common ones and the two simplest read

$$f'(x) \simeq \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon} \equiv g_{\mathrm{I}}(x) \tag{1}$$

$$f'(x) \simeq \frac{f(x-2\varepsilon) - 8f(x-\varepsilon) + 8f(x+\varepsilon) - f(x+2\varepsilon)}{12\varepsilon} \equiv g_{\mathrm{II}}(x)\ . \tag{2}$$

In this exercise we will use them in a particular case, trying to understand how they work and their limitations. You are highly encouraged to carefully read and think about the *whole* exercise before you start writing code. Since we want to run our code using both `float` and `double` variables, you are encouraged to use a pre-processor statement like `#define real double` at the beginning of your code and, then, to use `real` everywhere as type of your variables. To avoid to have troubles with the `printf` format specifier, use only `%e` which work both with `float` and with `double` variables.

(i) Consider the function $f(x) = \dfrac{x + \cos(x) - 1}{x}$ and calculate $f'(x)$ analytically.

(ii) Evaluate $\lim\limits_{x\to 0} f(x)$ and $\lim\limits_{x\to 0} f'(x)$.

(iii) Write a program which evaluates the function $f(x)$ in the interval [`xMin`, `xMax`], discretising it with a step $\varepsilon = 10^{-1}$, so that you evaluate the function in $N = (\texttt{xMax} - \texttt{xMin})/\varepsilon + 1$ points. Use the results to calculate the derivative $f'(x)$ inside the given interval. Fix `xMin=-10` and `xMax=10`. Your code should print enough information to the standard output, such that it is then possible to explore the properties Eqs. (1) and (2). A possible way to structure your code is to,

- (a) set up a pair of `C` functions which, given x, return the analytical values $f(x)$ and $f'(x)$;
- (b) add two more functions which implement Eqs. (1) and (2);
- (c) use a `printf` statement in order to print to the standard output the values of x, $f(x)$, $f'(x)$, $g_{\mathrm{I}}(x)$ and $g_{\mathrm{II}}(x)$.

(iv) Redirect the standard output of your code to a file and use `gnuplot` to plot $f(x)$, $f'(x)$ and $g_{\mathrm{I}}(x)$ on top of each other. Zoom around $x = 0$ and check if what you expect is indeed displayed. If not, go back to your code and try to fix it. Keep in mind what you learnt about decimal numbers comparison and decide whether it is correct to use the `==` operator. Does the same problem occur for $g_{\mathrm{II}}(x)$? If so, act accordingly.

(v) Plot now $|f'(x) - g_{\mathrm{I}}(x)|$ as well as $|f'(x) - g_{\mathrm{II}}(x)|$. Use a logarithmic scale for the y-axis setting its range properly. The command `set format y '10^{%T}'` may be useful. What do you learn from this plot about Eqs. (1) and (2)?

[1] If you are interested in reading more about it, `https://en.wikipedia.org/wiki/Numerical_differentiation` can be a good starting point. If you are interested in higher order derivatives or more accurate formulas also for the formulas for forward and backward derivatives, `https://en.wikipedia.org/wiki/Finite_difference_coefficient` should provide you with some answers. A general calculator of coefficients (for any order and type of derivative) is available on the web at `http://web.media.mit.edu/~crtaylor/calculator.html` where a great explanation about how to obtain the coefficients in general can be found.

(vi) Modify now slightly your code and move some code from your `main` to a

```
real CalculateAverageDerivativeDeviation(real xMin, real xMax,
                                         real epsilon){/*...*/}
```

function, which should now calculate

$$\delta = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\Big[f'(x_i) - g_{\mathrm{I}}(x_i)\Big]^2},$$

always discretising the interval [xMin, xMax] with a step $\varepsilon = x_{i+1} - x_i$.

(vii) Make your code calculate δ for several values of ε, starting with $\varepsilon = 10^{-1}$ and halving it, until it goes below the threshold $\varepsilon_{\min} = 10^{-6}$. Use a `printf` statement to print in an exponential form ε and δ to the standard output and redirect it to a datafile. Repeat this step both with `float` and `double` variables.

(viii) Use `gnuplot` to visualise your data. Below, you find a possible way to plot them.

```
set logscale xy
set format xy '10^{%T}'
set xrange[1:1.e-7] reverse
plot "data_float.dat"  u 1:2 w lp pt 6,\
     "data_double.dat" u 1:2 w lp pt 6
```

Can you explain the outcome? What do you learn from your plot?

(ix) Repeat task (vii) using g_{II} in the expression of δ and plot again your data. You can add the new points on the old plot produced in task (viii). Is the outcome as you expected?

(x) Is there a way to decide whether to use `float` or `double` at compile time without editing your code? Said differently, what does the option `-D` of `gcc` do and how should it be used? Be careful, though! If the user forgot to use it, your code should still compile! Can `#ifndef` and `#endif` help you?

Advanced: A `bool` parameter in your function(s) may be used to decide which finite difference formula should be used. On top, you could use a pre-processor macro calling your function(s) (maybe defined when compiling the code).