Home Assignment 1
The Log Function
Due 15.4.18

Intro. In this assignment you will implement the function $\log_a(x)$. For each question, please adhere to the specific signatures and file names you are required to, as well as the format of submission. Out of format submissions will not be checked or graded. Furthermore, we recommend you to document your code. This way you could get meaningful revisions from us, improve in life, prosper and achieve longevity.

---

1. **Newton-Raphson with Bisection.** Implement the (in)famous root-finding algorithm in a function with the following signature:

   [zero, Niters]=NewtonRaphson(func, functag, tol, a, b)

   where

   - **func** is a function handle for the function whose zero we are looking for. A function handle is a variable that represents functions and evaluate them only at run time. For example, if we want to store the function $f(x) = \sin(\pi x) + 1$, we write

     f = @(x) sin(pi*x) +1

     and we call it by **f(1)** or, if needed, **f([0.5,1])**.
   - **functag** a function handle for the derivative of the function.
   - **tol** is the allowed absolute error at the root.
   - **a,b** indicate the interval (a, b) in which we look for the zero.

   Recall that, as we learned in class, if at some iterations x_{n+1} is outside (a, b), we *implement bisection instead* to find x_{n+1}. The outputs are

- **zero** the root of the function.
- **Niters** the number of iterations it took the algorithm to converge to the required error-tolerance.

For each of the following functions compute the (single) root with error-tolerance of $10^{-2}, 10^{-3}, \ldots 10^{-9}$ error, and plot **Niters** as a function of the tolerance. Submit the plots in a file called **NewtRaphConv.jpg**:

(a) $f(x) = x^3$, in $(-100, 10)$

(b) $f(x) = x^{10}$, in $(-10, 10)$

(c) $f(x) = \tanh(9x) + \frac{x}{2} + 0.3$, in $(-1, 1)$.

Advisory: Use MATLAB 'subplot' to fit all three plots into a single figure, or even better, plot them on the same plot to compare between the cases.

---

2. **Log with exponent.** Implement a $\log_a(x)$ function with the following signature

$$\textbf{[y] = myLog(a,x)} \ ,$$

so that $a^y = x$. It is *obligatory* to use MATLAB's function 'exp' in your implementation, and it is *prohibitive* to use MATLAB's function 'log'. **The error should be smaller then** 10^{-8}.

In a text file called **myLogRes.txt** store the outputs of your function, and its error from the MATLAB 'log' function, for the following calls:

(a) myLog(2,16)

(b) myLog(2,2^{20})

(c) myLog(0.1, 2^{20})

(d) myLog(12345, 0.12456)

The error should be represented on a log scale, i.e., if $x \neq 1$, and you computed **y** and the *real* logarithm is **z**, write to the text file both **y** and **err = log10(abs(y-z)/z))** .

3. **Log without exponent.** Repeat item 2, when now it is *not allowed* to use the 'exp' function. You may use only the basic arithmetic operations $+, -, \times, \div$, as well as a maximum of 20 pre-computed calls to MATLAB 'log'. This means that you need to decide for what numbers $x_1, \ldots, x_{20}$ you want to compute $\log(x_1), \ldots, \log(x_{20})$ *before* you get the input. The files should be called **myLog2** and **myLogRes2.txt**.

Advisory: The Taylor series of a smooth function around x_0 is

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \ .$$

Cautionary advice: The Taylor series converges at some radius, but at what rate?

Cautionary advice 2: You do not *have* to use the pre-computed $\log(x_j)$ if you do not need it. There are at least two different ways to do it.