The goal of this project is to develop a MATLAB function capable of computing the function

\[ f(x) = \log(x) \]

for all floating point numbers \( x \in [1, 2] \) and compare your function to the built-in function \( \log \) in MATLAB.

Below is a list of questions which will guide you towards this goal. You need to hand in the following material

1. Your answers to all these questions.
2. Your implementation of all MATLAB functions.
3. Documentation describing the purpose, call sequence and underlying algorithm of every MATLAB function which you have implemented.
4. A thorough comparison of your own function and MATLAB’s built-in function \( \log \).

You are free to use any piece of code which I have made available on the class website. You are not allowed to use built-in MATLAB functions in your own functions. You can only use the four basic arithmetic operations.

You should hand in your solution by noon on Monday, December 3rd, 2012.

You are allowed to work together in groups of 2 or 3 people. One report per group will suffice, but remember to state the name of all group members.

1. Let \( x \) be a positive double precision floating point number

\[ x = (1.f_1 f_2 f_3 \ldots f_{53})_2 \cdot 2^m. \]

Show that

\[ \log(x) = \log[(1.f_1 f_2 f_3 \ldots f_{53})_2] + m \cdot \log(2) \]

and explain why it is straightforward to compute \( \log(x) \) for every positive floating point number once we can handle \( x \in [1, 2] \).
2. Let $\alpha > 0$. Show that $\log(\alpha)$ is the unique solution of the equation

$$f(x) = \alpha, \quad \text{where} \quad f(x) = \exp(x). \quad (1)$$

**Caution:** You must not only show that $\log(\alpha)$ is the solution, but you must also explain why there are no other solutions.

3. It is clear that we will need a reliable implementation of $f(x) = e^x$ in order to solve equation (1). Let $p_n(x)$ denote the $n$th order Taylor polynomial of $f$ at the point $x_0 = 0$, i.e.

$$p_n(x) = \sum_{j=0}^{n} \frac{x^j}{j!}.$$  

Show that

$$\forall x \in \mathbb{R}: \left| p_n(x) - f(x) \right| \leq \frac{e|x|}{e^x} \frac{|x|^{n+1}}{(n+1)!}.$$  

What is the smallest value of $n$ for which you can guarantee that

$$\forall x \in [0, 1]: \left| p_n(x) - f(x) \right| \leq 2^{-53}.$$  

4. Now, consider the following MATLAB function `exp1`.

```matlab
function z=exp1(x,k)
    n=2^k; y=zeros(n,1); term=1;
    for j=1:n
        y(j)=term; term=term*(x/j);
    end
    z=tree_sum(y,k);
end
```

where the function `tree_sum` is given by

```matlab
function s=tree_sum(a,q)

% TREESUM
% Sums the first 2^q terms of the vector a using binary tree addition.
% Call sequence:
% s=tree_sum(a,q)
% Originally written in SUN Pascal by Ole Oesterby, DAIMI.
% MATLAB translation by Carl Christian Kjelgaard Mikkelsen
```
In particular, you must identify and describe the underlying idea of `exp1` and demonstrate that the function is completely unreliable for negative values of $x$. Write a new subroutine `exp2` which uses `exp1` as much as possible, but which does not have this problem. Compare `exp2` to the built-in function `exp` for a suitable set of values of $x$.

Remark: You are not required to explain the inner workings of the function `tree_sum`.

5. Now use Newton’s algorithm to compute an accurate value for $\log(2)$ and $\log(\log(2))$. Compare your values to MATLAB’s values obtained with the built-in function `log`.

6. In order to use Newton’s algorithm we need a good initial guess for

$$ y = \log(x). $$

One such initial guess is

$$ y = ax + b, $$

where

$$ a = \log(2), \quad b = \frac{1 + a + \log(a)}{2}. $$

Show that

$$ \phi(x) = \log(x) - (ax + b) $$

satisfies

$$ \forall x \in [1,2] : -(a + b) \leq \phi(x) \leq a + b. $$

Write a MATLAB function `init_log` which given $x \in [1,2]$ returns

$$ y = ax + b. $$
7. Write a MATLAB function `my_log1` which uses Newton’s method and `init_log` to compute \( \log(x) \) for any floating point number \( x \) in \([1, 2]\).

8. Compare `my_log1` with the built-in function `log` on the interval \([1, 2]\). Determine experimentally where the relative error is large.

9. It is necessary to adopt a different strategy at the left end of the interval. Show that the \( n \)th order Taylor polynomial for

\[
h(z) = \log(1 + z), \quad |z| < 1
\]

is given by

\[
q_n(x) = \sum_{j=1}^{n} \frac{(-1)^{j-1}}{j} x^j
\]

and write a MATLAB routine `log_series` which given an \( x \) and an integer \( k \) returns

\[
q_n(x), \quad \text{where} \quad n = 2^k.
\]

**Caution:** Remember to sum the terms using a good algorithm.

10. Finally, write a MATLAB function `my_log2` which can compute \( \log(x) \) using \( q_n(x - 1) \) for \( x \in [1, 3/2] \) and Newton’s method for \( x \in [3/2, 2] \). Compare your function to MATLAB’s built-in function on a suitable range of points.