MATLAB

Assignment 3

1. Calculate absolute and relative errors in the following approximations for x by x_n:

   (a) $x = \pi, \quad x_n = 22/7$

   (b) $x = e, \quad x_n = 2.718$

   (c) $x = e/100, \quad x_n = 0.02718$

   (d) $x = 10^\pi, \quad x_n = 1400$

2. Consider that a real number is represented with only 4 digits after the decimal point (*a.k.a. 4 digits precision*), then

$$1 = \frac{1}{3} \cdot 3 \approx 0.3333 \cdot 3 = 0.9999 \neq 1$$

   (a) Suggest a solution to the problem described above.

   (b) Using Matlab, find the *smallest* n such that

$$\underbrace{\frac{1}{3} \times \frac{1}{3} \times ... \times \frac{1}{3}}_{n} \times \underbrace{3 \times 3 \times ... \times 3}_{n} \neq 1 \ .$$

3. One approximates $y = \sin x$ using Taylor expansion around 0. The well known Taylor formula gives

$$y = \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!} + \underbrace{O\left(\frac{|x|^{2(n+1)-1}}{(2(n+1)-1)!}\right)}_{=R_n},$$

   where $= R_n$ is a remainder i.e.the error is bounded by $= R_n$. In this question we want $|R_n| \leq 10^{-3}$.

   (a) Write a matlab program that approximates $y = sin(x)$ using the algorithm above.

   (b) Find the minimal n required for any $|x| < \frac{\pi}{2}$. To do so one do either (i) solve the inequality $\frac{x^{2n-1}}{(2n-1)!} < 10^{-3}$ or (ii) use trial and error approach (may not be the best idea) with the program from (a).

   (c) Find $x > \frac{\pi}{2}$ such that your program return result with error greater then 10^{-3} Is this algorithm stable for large x's? Explain!

   (d) Write a new matlab program for a more stable algorithm that uses the following identity

$$\sin x = \sin(m\pi + z) = (-1)^m \sin z,$$

   where $x = m\pi + z$, m is a integer and $|z| < \frac{\pi}{2}$. Do you need n larger than one you found in (b)?

4. A floating point number can be represented by $Fl(x) = \pm(1.d_1 d_2 ... d_t)_2 2^e$, where the sequence $(.d_1 d_2 ... d_t)_2$ is called *mantissa*, t is the number of digits in the mantissa, 2 is the *base* (or *radix*) and e is the *exponent*.

   (a) For a DEC-VAX using **single precision** we have $t = 24$ and $-128 \leq e \leq 127$. What are the smallest and largest numbers, m and M, that can be stored in this computer?

   (b) Repeat (a) for the case of **double precision**, where $t = 52$ and $-1024 \leq e \leq 1023$.

   (c) What is the smallest value of x in (a) and (b) for which x^{100} will overflow (i.e., $x^{100} > M$)?

(d) The following calculations are done using single precision:

$$\text{(i) } 10^2 - \sqrt{10^4 - 1}, \qquad \text{(ii) } 10^4 - \sqrt{10^8 - 1}, \qquad \text{(iii) } 10^8 - \sqrt{10^{16} - 1}.$$

Determine whether each of the results of (i)–(iii) will be zero, nonzero or overflow.

5. Let $S_x(N) = \sum_{n=0}^{N} \frac{x^n}{n!}$. Then $\lim_{N\to\infty} S_x(N) = e^x$ for any $x \in \mathbb{R}$.

   (a) Using Matlab, calculate $S_{x=10}(N)$ for N=[10:10:100]. You obtain a vector of numbers, $\vec{S}$. The elements $S(i)$ of $\vec{S}$ are partial sums that approximate the function e^x with different accuracies.

   (b) Using Matlab's built-in function **exp(x)** you can calculate e^x "exactly" (i.e., with double precision accuracy). Using **exp(x)** plot the graph of relative errors

$$R_x(N) = \left| \frac{e^x - S_x(N)}{e^x} \right|$$

   for $x = 10$ as a function of N.

   (c) Repeat (a) and (b) with $x = -10$.

   (d) Do the graphs show convergence for both values of x?

   (e) Do the elements of the $\vec{S}$ converge to the correct value in both cases?

   (f) Explain your answers to (d) and (e).

Hand in: answers to questions, a program used to answer the question, i.e. all m-files, and all the output it creates including plots (whenever relevant).

THE MATLAB DIGEST

Here are some additional functions in MATLAB and some useful tips:

- **if ... elseif ... else ... end**. These are the MATLAB commands that perform conditional statements. Note that **elseif** is used as an **else + (new) if**, unlike C.

- ```
  while  expression
         operations
  end
  ```
  This is how MATLAB performs a loop with a conditional breaking point, i.e., the loop ends when the condition in **expression** is false, e.g., if **expression** is $\mathbf{1 < 0}$.

- To perform finite number of steps loop use

  ```
  for index = values
       statements
  end
  ```

  the **values** above can be a range, e.g. **values=a:b**, but can also be any vector, e.g. **values = [2.5,5,17]**

- To avoid unnecessary loops which are often slow - 'vectorize' your funciton, so it can be called with a vector and return a value per each element in the input vector like for instant matlab's **sin(0:0.1:2*pi)** . To do so all relevant operations should become element-wise operators, e.g. use **.*** instead of *****.
  ```