ESC 407H
Computational Methods in Engineering Science
Homework 2
Taylor Series, Computer Arithmetic, Errors

Due Date: Tuesday, September 22 at 11:59 p.m.

Problem 1

In numerical computation, the machine epsilon ε can be thought of as the smallest number that when added to one gives a number greater than one. Develop a MATLAB function called `macheps` that finds this number and compare it with the MATLAB command `eps`, which is is the positive distance from `abs(x)` to the next larger floating point number with the same precision as `x`. The function should have no inputs and the outputs should be the machine epsilon that your function finds and MATLAB's machine epsilon. *Hint:* Start with the number one and keep halving it until that number added to one is no longer different from one.

Problem 2

The ancient Babylonians (2000 B.C.) used an iterative method for computing the square root of a number a that worked as follows.

Start with $x = \sqrt{a}$ and square both sides

$$x^2 = a \quad \Rightarrow \quad 2x^2 = x^2 + a \quad \Rightarrow \quad 2x = x + \frac{a}{x} \quad \Rightarrow \quad x = \frac{x + a/x}{2}.$$

The idea is that you start with a guess for x, plug that guess into the right-hand side, and then this will generate a number on the left which will be your next guess for x, that is,

$$x_{\text{new}} = \frac{x_{\text{old}} + a/x_{\text{old}}}{2}$$

This is sometimes called the *divide and average* method for estimating a square root.

Write a MATLAB function called `approx_sqr` that approximates the square root of a given number a to within a user-specified relative error. The inputs to the function should be a, the relative error stopping criterion, and the maximum allowable number of iterations. The outputs should be the approximate value of $\sqrt{a}$, the relative error at the end of the computation, and the required number of iterations to reach that error. For example, it might look something like this. If `stop_err` is omitted, then function should assign the default value `stop_err = 0.0001`. If `max_it` is omitted, it should assign the default value `max_it = 50`. Note that the function could be called as follows

```
[sqa, rel_err, iter] = approx_sqr(23893243)
```

or as

```
[sqa, rel_err, iter] = approx_sqr(23893243, 0.0001)
```

or as

```
[sqa, rel_err, iter] = approx_sqr(23893243, [], 100])
```

or as

```
[sqa, rel_err, iter] = approx_sqr(23893243, 0.0001, 100)
```

so you will also have to check for empty values of those two input variables (these commands should help https://www.mathworks.com/help/matlab/ref/exist.html and https://www.mathworks.com/help/matlab/ref/isempty.html).

When computing the relative error in an iterative calculation like this, you determine it using

$$R_x = \left| \frac{\text{present approx} - \text{previous approx}}{\text{present approx}} \right|.$$

Problem 3

Write a MATLAB function called `maclaurin_sum` to compute the Maclaurin series expansion of a function to within a user-specified relative error and then evaluated at user-specified point x_0. The Maclaurin series must be representable in a closed form so that it can be passed as an anonymous function to your function. Examples of such functions include e^x, $\cos x$, and $\sin x$. The MATLAB function should take as inputs

- the anonymous function for the closed form version of the Maclaurin series for $f(x)$; for example, for e^x it would be `f = @(x,n) (x^n)/factorial(n);`
- the point x_0 at which you want the Maclaurin series to approximate $f(x_0)$,
- the desired approximate relative error of the approximation,
- the maximum number of terms allowed in the Maclaurin series.

Your MATLAB function should output

- the Maclaurin series approximation to $f(x_0)$,
- the approximate relative error of the approximation,
- the number of terms needed in the Maclaurin series to achieve the desired relative error.

For example, my function looks like:

```
function [mac_sum,err,n] = maclaurin_sum(f,x0,rel_err,max_it)
```

Now, for the following functions

(i) find the first four nonzero terms in the Maclaurin series,

(ii) find the general term and write the series in summation form,*

(iii) plot all four partial sums of the series as well as the original function on one plot,

*For example, $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

(iv) write a single script called `maclaurin_run` that uses your `maclaurin_sum` function to approximate each of the $f(x)$'s at the indicated x_0's and using the indicated relative error tolerances.

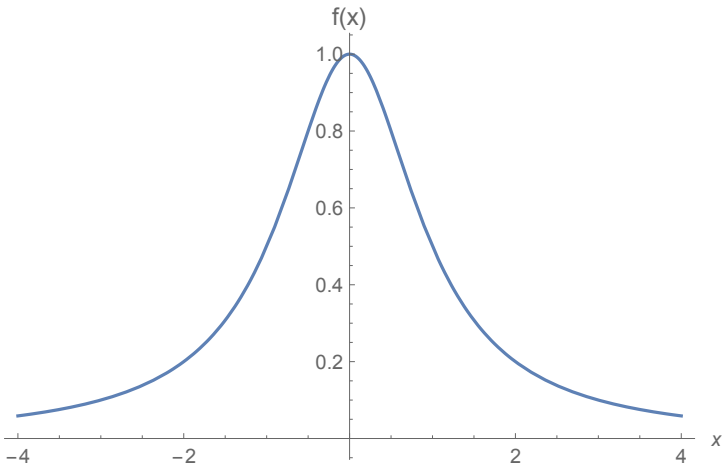
You should make an informed choice of the maximum number of terms allowed in the Maclaurin series to achieve the desired relative error.

(a) $f(x) = x^2 \sin x$ at $x_0 = 3$ and at $x_0 = 30$ with a relative error of 10^{-9},

(b) $f(x) = \cos(3x^2)$ at $x_0 = 2$ with a relative error of 10^{-10},

(c) $f(x) = \displaystyle\int_0^x \cos \xi^2 \, d\xi$ at $x_0 = 1$ with a relative error of 10^{-11},

(d) $f(x) = 1/\left(1 + x^2\right)$ at $x_0 = 0.5$ and at $x_0 = 2$ with a relative error of 10^{-9}. If you plot $f(x)$ (shown below), it looks perfectly well behaved everywhere. Why does the first x_0 converge and the second one does not? 🦄 This one is tricky and is worth bonus points if you get it.



In all cases, show the calculations you used to obtain the Maclaurin expansions.