Assignment I: Numerical Computing
January 10, 2023
Due: January 23rd, 2023

The maximum number of points is 44 points. Please submit all `MATLAB` code attached at the back of your assignment write-up.

---

1. **[4 points] Plotting in Matlab**

$$\begin{aligned} x(t) &= \sin(2t), \\ y(t) &= \cos(t), \\ z(t) &= t \end{aligned}$$

for domain Ω with $t \in [0, 3\pi]$. (i) Plot the 3D curve in `MATLAB` using the command `plot3` (up to you to make it as beautiful and legible as you would like: label axes, darker line, color line based with height, etc.). (ii) Compute in `MATLAB` the arc length of the curve based on the formula: .

$$\Gamma = \int_\Omega \left(x'(t)^2 + y'(t)^2 + z'(t)^2 \right) \, dt. \tag{1}$$

Include the code you used and report the result.

2. **[10 points] Floating-Point Exceptions**
Using `MATLAB`, do some simple calculations that lead to exceptions. Write the code you used and report what you get [2pt per item]:

- Generate an overflow other than division by zero, in both single and double precision.
- Divide a normal number by infinity and zero and see if the answer makes sense to you. How about dividing infinity by -infinity? How about dividing 0 by 0?
- Perform a comparison (e.g., $x < y$, $x > y$, $x == y$) between infinities, NaNs, and normal numbers to determine if and how these are ordered.
- Generate a NaN and then perform some arithmetic operations between a NaN and a normal number. What do you get?
- Produce an IEEE floating-point signed positive and negative zero and take their square roots. Can you detect any difference between $+0$ and -0 in `MATLAB`?

3. **[10 points] Algorithm to Compute π**
π is an irrational number and therefore cannot be expressed as a common fraction. There are many methods to compute many digits of π, and lots of them suffer from

numerical accuracy problems. Here is one of them due to Archimede's: start with $t_1 = 1/\sqrt{3}$ and then iterate

$$t_{i+1} = \frac{\sqrt{1+t_i^2}-1}{t_i} \tag{2}$$

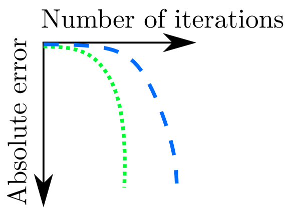
and for large i you can get a good approximation $\pi \approx 6 \cdot 2^i \cdot t_i$.

3.1 [7 points] Sources of Error

[4 pts] Using `MATLAB`, do this calculation up to $i = 30$ with both single and double precision, and report how many digits of accuracy you get and after how many iterations (Note:
$\pi = 3.141592653589793238462643383...$ or `MATLAB` has a built-in constant `pi`), accompanied with plots of the error as a function of number of iterations. Your choice if you use absolute or relative error (with or without the absolute value.)

<u>Hint</u>: How to set calculation in single or double precision?
Option 1: `format long` for double precision and `format short` for single precision;
Option 2: `a = zeros(n,1)` makes a vector of zeros in double precision of dimension $n \times 1$ but `a = zeros(n,1,'single')` makes a similar vector but in single precision.

Iteration	Single precision	Double precision	Single precision error	Double precision error
1	...	...	...	...
2	...	...	...	...
...	...	...	...	...



[3 pts] Estimate the relative error due to round-off and explain when it is large and why. You'll find it helpful to inspect the filled out table above and/or plots. What kind of error dominates the numerical estimate of π for small i and what kind of error dominates for large i?

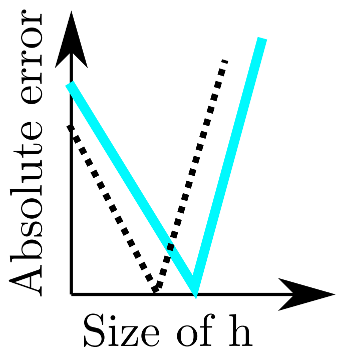
3.2 [3 points] The Fix

Find a way to rewrite the iteration (2) so that you avoid cancellation errors in the numerator and get much better accuracy and repeat the calculation. How many digits of accuracy can you get with the improved formula? Please reproduce a similar table and plot as you did for part 3.1 and refer to these results in your answers.

4. **[10 points] Round Off vs Truncation Error**
In the exercises below you will repeat the calculation presented in class for the approximation to the first derivative for the centered approximation to the second-order derivative:
$$f''(x = x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}.$$
This is called the **centered difference approximation** of the second derivative of a function $f(x)$ at $x = x_0$.

Using a simple function like $f(x) = \sin(x)$ and $x_0 = \pi/4$, calculate the second derivative at x_0 for the above approximation with several h on a logarithmic scale (say $h = 2^{-m}$ for $m = 1, 2, ..., 10$). Compare to the known derivative, using both single and double precision. For what h can you get the most accurate answer?

h	Single precision	Double precision	Single precision absolute error	Double precision absolute error
2^{-1}	...	...	...	...
2^{-2}	...	...	...	...
...	...	...	...	...
2^{-10}	...	...	...	...



5. **[10 points] Well- or Ill-conditioned?**
Show that the problem of computing $\sin(x)$ is well-conditioned for x near 0 but poorly conditioned for x near 2π. This is a paper & pencil problem (no need for `MATLAB`). *<u>Hint</u>: For inspiration, I suggest you look at the Numerical Computing Notes (day 1), second example of Conditioning. Slides are on Canvas.*