Due: Tuesday, January 28th, before class.

Name: Solution

Document your work and note any collaborators. In your write-up, please neatly document any code you have written, alongside its output, and including any necessary commentary to answer the question. Please use an `.ipynb` notebook for any code, include commentary in markdown code blocks, and print the `.ipynb` notebook to PDF to attach to your submission.

The LaTeXsource for this file may be helpful: https://www.overleaf.com/read/hxdgczphzyhh#9cc09e.

Problems:

1. IEEE 754 *single-precision* specifies that 23 binary bits are used for the value f in the significand $1 + f$ where

$$f = \sum_{i=1}^{23} b_i 2^{-i}, \qquad b_i \in \{0, 1\}.$$

Because they need less storage space and can be operated on more quickly than double-precision values, single-precision values can be useful in low-precision applications (e.g., neural networks often use single-precision or even lower precision representations for weights). They are supported as type `Float32` in Julia.

   (a) In base-10 terms, what is the first single-precision number greater than 1 in this system?
   Machine epsilon is 2^{-23} since we have 23-bit mantissa. Exponent for 1 is $n = 0$, so the next single-precision number is

$$(1 + 2^{-23}) \cdot 2^0 = \boxed{1 + 2^{-23}}.$$

   (b) What is the smallest positive integer that is not a single-precision number? (Hint: For what value of the exponent does the spacing between floating-point numbers become larger than 1?)
   Starting from exponent $n = 24$, the spacings become greater than 1. This is because $2^{24}\epsilon = 2$. Thus the smallest positive integer that is not a single-precision number is $\boxed{2^{24} + 1}$.

2. Calculate the relative condition number of

$$f(x) = \frac{e^x - 1}{x}$$

and show that $|\kappa_f(x)| < 1$ for all $|x| < 1$ and $x \neq 0$.

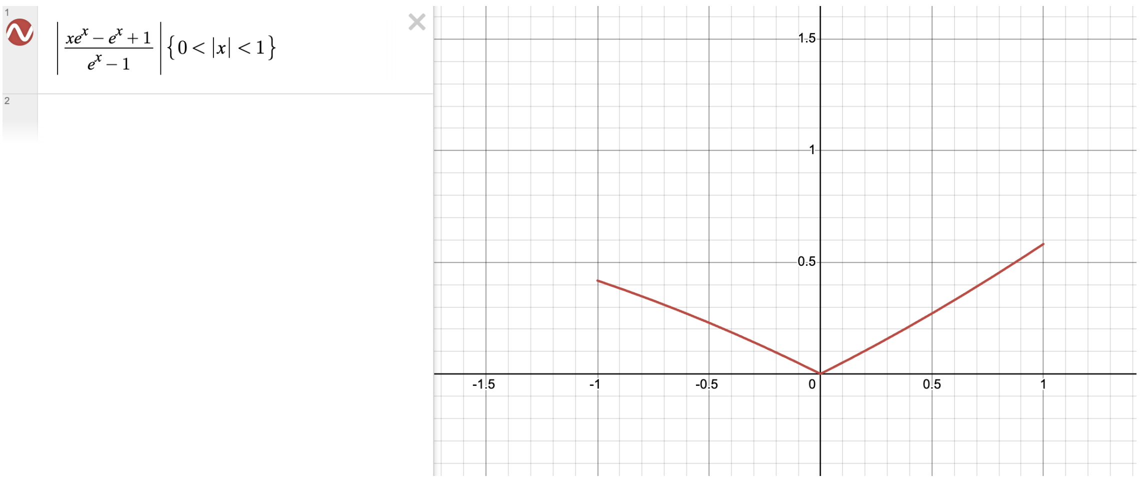
The derivative of f is

$$f'(x) = \frac{e^x}{x} - \frac{e^x - 1}{x^2}.$$

f' is continuous on the domain $(-1,1)/\{0\}$, thus we can use the formula $\kappa_f(x) = \left|\frac{xf'(x)}{f(x)}\right|$ to calculate the relative condition number. This yields

$$\kappa_f(x) = \left|\frac{xe^x - e^x + 1}{e^x - 1}\right|.$$

The simplest way to show that $|\kappa_f(x)| < 1$ is by graphing it:



An alternative way is to show that the expression inside the absolute value is monotonically increasing in the domain $(-1,1)/\{0\}$ and the boundary values are within $(-1,1)$.

Lab:
(Not entirely stable) Consider the function

$$f(x) = \frac{e^x - 1}{x}.$$

As we saw in the previous problem, $|\kappa_f(x)| < 1$ for all $|x| < 1$ and $x \neq 0$. In that sense, f is "easy" to compute accurately. In practice, though, it's not so simple.
An obvious sequence of steps to compute f is as follows:

$$y_1 = e^x, \qquad y_2 = y_1 - 1, \qquad y_3 = y_2/x. \tag{1}$$

The operations for y_1 and y_3 are well conditioned for $|x| < 1$, but the subtraction to get y_2 will suffer from cancellation error if $y_1 \approx 1$, or $x \approx 0$. That error makes this sequence of operations unstable for $x \approx 0$.
Now consider the Maclaurin series expansion of f,

$$f(x) = 1 + \frac{1}{2!}x + \frac{1}{3!}x^2 + \cdots. \tag{2}$$

For $x > 0$, every term in the series is positive, leaving no possibility of cancellation error. (The analysis for negative x is more subtle.) If $x \approx 0$, then we should be able to find n such that $x^n/(n+1)!$ is smaller than machine precision, so that adding it to the terms before it will not change the result numerically. Thus a truncated form of the series can serve as a stable method for $x \approx 0$.

Goals: You will experiment with the two methods of computing f and observe their relative accuracy.

Procedure: Perform the following steps.

(a) Julia has a stable way of computing y_2 in (1) without using subtraction. You will use it to get reference "exact" values of f. Define `x` as

```
x = exp10.(range(-16, stop=0, length=500));
```

which creates a vector of 500 logarithmically spaced points between decades 10^{-16} and 10^0. (Note that this means that the points x_j satisfy $\log_{10}(x_{j+1}) - \log_{10}(x_j) = 16/500$.)

Create a vector `y` the same size as `x` such that y_j is the result of `expm1(x[j])/x[j]`. (Note that you do not need to use a `for loop` to complete this task – you can *vectorize* this operation using the "dot" operation. To vectorize the `expm1()` function, you use `expm1.()`, and to vectorize the `/` operation, you use `./`.)

See `https://docs.julialang.org/en/v1/base/math/#Base.expm1` for more information about the `expm1()` function.

(b) Create a vector `z` such that z_j is computed using the three steps in (1) for x_j.

(c) Compute a vector of relative differences between z_j and y_j. Make a semi-log plot (x axis on logarithmic scale and y axis on linear scale) of the result as a function of x. You will see a loss of accuracy as $x \to 0$. To make this plot, you will need to use the `Plots` package, which you will need to install.

(d) By trial and error, find a value of n such that $0.01^n/(n+1)!$ is less than `eps()` (machine epsilon). This defines the last term to keep in truncating the series (2).

(e) Create a vector `w` such that w_j is computed using the truncated form of (2). Repeat step 3 for `w` in place of `z`. This time you should see accuracy maintained as $x \to 0$. (If more terms of the series were kept, the accuracy could be maintained as $x \to 1$ as well, but the direct method is more efficient there.)