

Exercise L^AT_EX: A "quick and dirty" implementation of the exponential function

L.C. Ahler

1 Introduction to exp

The exponential function is usually defined as the solution to the differential equation:

$$\frac{dy(x)}{dx} = y(x) \quad (1)$$

This behavior is exhibited by the well known function e^x where e is Euler's number.

2 Numerical approximation of the exponential-function

As for many other functions the exponential function can be expanded as a Taylor series:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots \quad (2)$$

Numerically it is impossible to incorporate an infinite number of terms, and such this Taylor expansion is going to have an error. This error is proportionate to the number of terms in the Taylor expansion and the size of the argument x . Thus the best way to calculate the Taylor series of the exponential function is given in the appendix.

3 Explanation of the code

The code calculates the exponential function of a given argument via a Taylor expansion which includes 11 terms. This causes very large errors for big arguments. Therefore the function uses that fact that $e^x = (e^{x/2})^2$. It can be shown that for arguments $x < \frac{1}{8}$ the error of the Taylor expansion with 11 terms is of magnitude $\epsilon < 10^{-15}$. Thus by dividing the argument by 2 until it is less than $\frac{1}{8}$, calculating the exponential function via the Taylor expansion and then squaring the result accordingly the error can be kept relatively low.

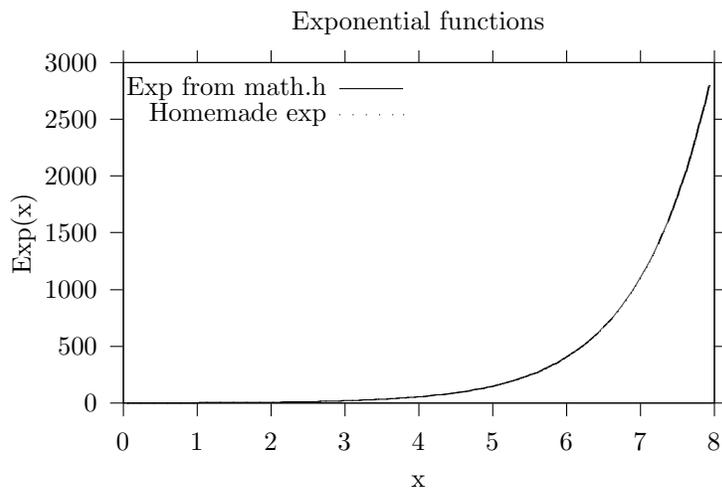
Another remark should be made about negative arguments. Such arguments would cause the Taylor expansion to have alternating positive and negative terms. This is ineffective and since $e^{-x} = \frac{1}{e^x}$ it is smarter to calculate the function of the corresponding positive argument and then dividing. Lastly it is noted that the code calculates the expansion as:

$$1 + x \cdot \left(1 + \frac{x}{2} \cdot \left(1 + \frac{x}{3} \cdot \left(1 + \frac{x}{4} \cdot \left(1 + \frac{x}{5} \cdot \left(1 + \frac{x}{6} \cdot \left(1 + \frac{x}{7} \cdot \left(1 + \frac{x}{8} \cdot \left(1 + \frac{x}{9} \cdot \left(1 + \frac{x}{10}\right)\right)\right)\right)\right)\right)\right)\right)\right)\right)$$

Instead of using the formal description shown in equation (2). This is simply because it reduces the number of operations required for calculation and as such increases efficiency.

4 figures

To illustrate how well the code approximates the exponential function here is a function in which it is plotted alongside the `exp(x)` function defined in `math.h`.



References

- [1] Federov, Dmitri (2021), "Practical programming and numerical methods"

9

Appendices

A Implementation code [1]

```
double ex(double x){
    if(x<0)return 1/ex(-x);
    if(x>1./8)return pow(ex(x/2),2);
    return 1+x*(1+x/2*(1+x/3*(1+x/4*(1+x/5*(1+x/6*
        (1+x/7*(1+x/8*(1+x/9*(1+x/10)))))))));
}
```