5DV005, Fall 2018, Lab session 3

Carl Christian Kjelgaard Mikkelsen

November 21, 2018

Contents

1 The time and the place

The lab session will take place on

Wednesday, November 21th, 2018, (kl. 13.00-16.00), Room MA416-426.

2 The problems

Problem 1 The function `BadExp` is a very, very bad implementation of the natural exponential function

$$f(x) = e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!} \tag{1}$$

which has been written by somebody who understands that

$$f(x) = \lim_{n \to \infty} \sum_{j=0}^{n} \frac{x^j}{j!} \tag{2}$$

but has no real appreciation of the difference between real arithmetic and floating point arithmetic!

1. Explain how the update of the variable `term` is related to the fact that

$$\frac{x^{j+1}}{(j+1)!} = \left(\frac{x^j}{j!}\right) \frac{x}{j+1}. \tag{3}$$

2. Explain the logic underlying the termination of the `while` loop. Why does it make sense?

3. Develop a minimal working example `work/BadExpMWE1.m` which uses `BadExp` to generate a plot similar to Figure 1. Use the points `x = linspace(-30,30,1025)` to sample `BadExp`.

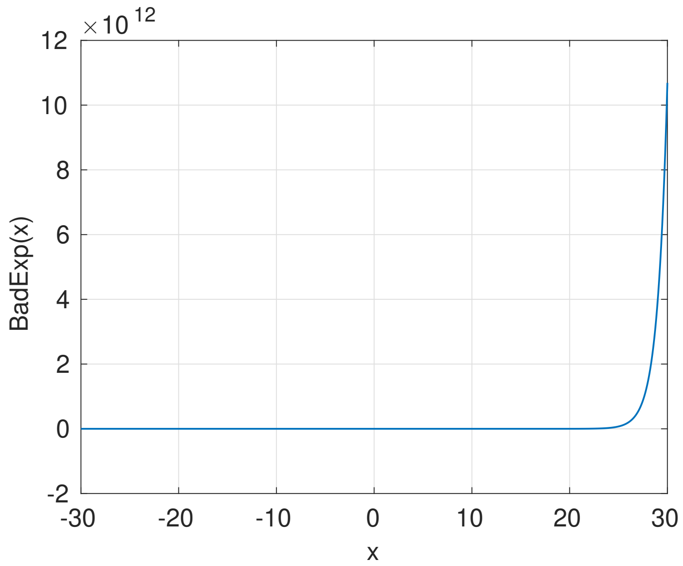

Figure 1: A deceiving plot of the graph of `BadExp`, the bad implementation of the natural exponential function

The graph appears acceptable, but appearances can be quite deceiving! An alarm bell should go off in your head when you notice the extreme range of function values which completely obfuscates any details towards the left end of the plot. Our target T is $T(x) = f(x)$. Our approximation A is $A(x) = \texttt{BadExp}(x)$.

4. Extend `BadExpMWE1` to generate a second figure which shows the relative error, i.e., the fraction
$$R(x) = \frac{T(x) - A(x)}{T(x)} \tag{4}$$
Aggressive measures are necessary to get a good impression of the relative error. Apply the `MATLAB` commands `abs` and `log10` as needed. Produce a figure similar to Figure 2.

5. Identify the points where the absolute value of the relative error is strictly less than unity.

6. What can be said about the sign of the computed value if the relative error is strictly less than unity?

7. The real function $x \to e^x$ is always strictly positive, but `BadExp` returns negative values, which is totally unacceptable! Use the `find` and `numel` commands to verify, that exactly 75 of the 1025 values returned by `BadExp` are negative! `idx = find(y<0)` and `x(idx)` are useful commands, right?

8. Determine the sign of the entries of `x` for which the corresponding entries of `y` are all negative. The `sign` command can be useful.

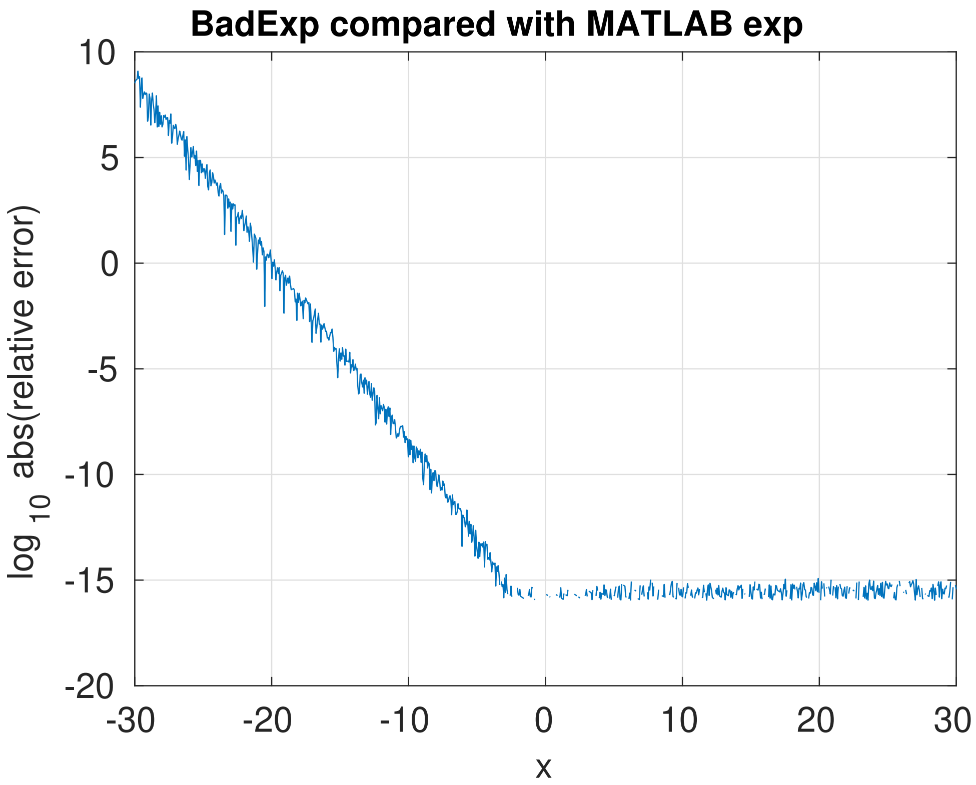

Figure 2: An illustration of the size of the absolute value of the relative error for `BadExp`. The functions `abs` and `log10` have been applied to allow us to see both very small and very large values of the relative error.

Problem 2 The problems in `BadExp` are limited to negative values of the input argument! There is nothing wrong with the sign of `BadExp` for positive values and the relative error is rather small here. There is a function called `MyExp` which calls `BadExp`, exploiting the trivial identity

$$e^x = \frac{1}{e^{-x}}, \tag{5}$$

in order to avoid working directly with negative values of the argument x.

1. Explain how `GoodExp` isolates the positive and negative input arguments.
2. Explain how `GoodExp` exploits equation (5).
3. Develop a minimal working example `GoodExpMWE1` which is similar to `BadExpMWE1` and generates a plot similar to Figure 3.
4. Verify that absolute value of the relative error is bounded by $12u$ where $u = 2^{-53}$ is the unit round off error in double precision.
5. At this point you should be wondering how accurately we can compute $x \to e^x$ when x is a real number and $x \in [-30, 30]$. What are the theoretical limitations? Compute the condition number and show that you can expect the relative error to be around $30u$, where u is the unit round off error.

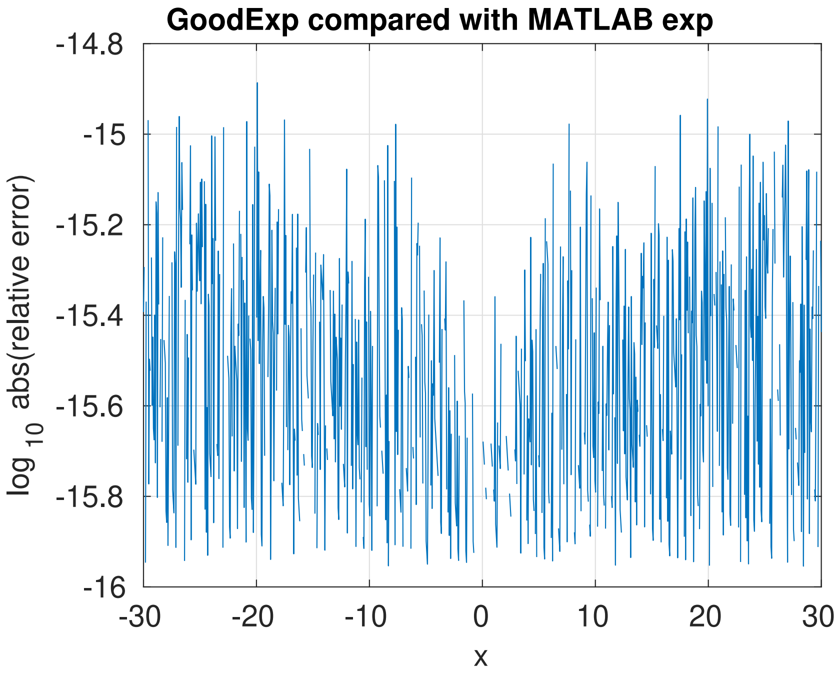

Figure 3: A plot of $\log_{10}$ of the absolute value of the relative error for `GoodExp` when compared with the built-in function `exp`.

Problem 3 In this problem you will develop a subroutine which can be used to compute $\sqrt{\alpha}$ for all $\alpha > 0$. Recall that any floating point number $\alpha > 0$ can be written as

$$\alpha = (1.f_1 f_2 f_3 \dots f_k)_2 \times 2^m, \tag{6}$$

where m is the exponent and $f_i \in \{0, 1\}$ are the individual bits of the mantissa. In IEEE single precision $k = 23$. In IEEE double precision $k = 52$.

1. Show that in order to compute $\sqrt{\alpha}$ is suffices to have the ability to compute $\sqrt{x}$ where $x \in [1, 4]$.

   Hint: In general $\sqrt{ab} = \sqrt{a}\sqrt{b}$ and it is very simple to compute the square root of 2^{2k}, is it not?

2. Copy `lab3/scripts/l3p3.m` into `/lab3/work/MySqrt.m` and complete the function according to the specification and the inline comments.

   Remark 1 If your function `MyRoot` (Assignment 1) is ready, then your are encouraged to use it here. Otherwise, you are free to use the more general code `bisection` in the class repository.

3. Develop a minimal working example `/lab3/work/MySqrt.m` which compares `MySqrt` to the built-in function `sqrt` and generates a figure similar to Figure 4.

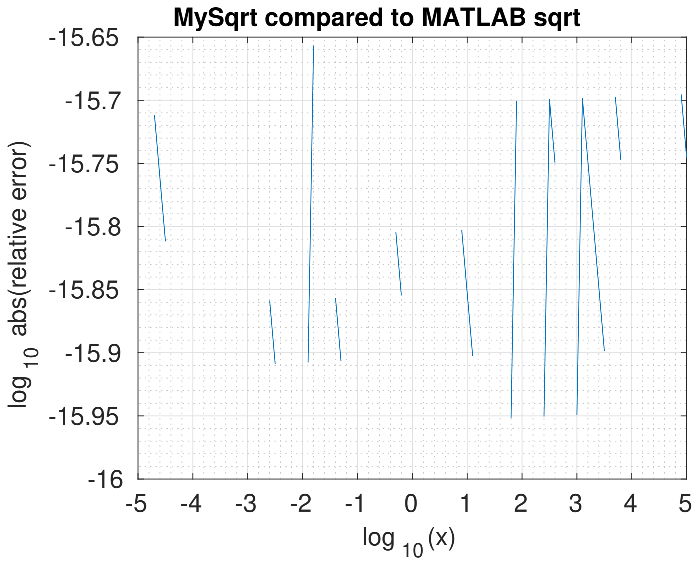

Figure 4: Comparison of `MySqrt` to MATLAB's built-in function `sqrt`.

Problem 4 This problem develops a function which can compute a standard ballistic table, i.e., the elevation as a function of the range to the target.

1. Copy the function `l3p4.m` into `/work/MyComputeRange` and complete the function according to the specification and the inline comments.

2. Copy the script `l3p4mwe.m` into `/work/MyComputeRangeMWE1.m` and execute it.

3. What evidence can you find online to suggest that the computed ballistic table is even remotely related to the real 152 mm D20 howitzer?

4. Can your program cope with the introduction of a head wind?