MAT 627 Assignment 1

Brian Tekmen

January 2026

Source code available here: https://github.com/etekmen13/mat627

1 Cancellation Errors

We approximate e^{-x} two ways:

$$\text{Alternating:} \quad e^{-x} = 1 - x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \dots$$

$$\text{Reciprocal:} \quad e^{-x} = \frac{1}{1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots}$$

and compare the relative error using the exact answer `exp(-x)` (in Rust: `f64::exp(-x)`). To avoid factorial overflow, I use the recurrence relation

$$a_{k+1} = a_k \cdot \pm\frac{x}{k+1}.$$

Here are my observations:

1.1 Alternating approximation is stable for $-x$, not for positive x

For negative x, the alternating approximation's relative error steadily decreases with n. This is expected because when $x < 0$, the series for $e^{-x} = e^{|x|}$ has non-alternating positive terms, which means you don't get precision loss from cancellation.

For positive x, the series does alternate, resulting in a subtraction of almost-equal large numbers, which loses precision, and results in an accumulation of rounding errors, leading to catastrophic inaccuracies.

1.2 Reciprocal method is stable for positive x, not $-x$

For positive x, e^{-x} as $\frac{1}{e^x}$ is stable because the denominator e^x is large and the taylor series for e^x is all positive terms, so no alternating cancellation.

For negative x, the reciprocal method seems to be worse. e^x becomes e^{-x} and we have the same alternating subtraction problem as before, then taking the reciprocal makes it even worse.

1.3 Relative error breaks for large x

At $x = 1000$, I get `inf` relative errors. This is expected because the exact value is `f64::exp(-1000)` in Rust which underflows to 0 so

$$\frac{|A_n - A|}{|A|} = \frac{|A_n - 0|}{0} \to \infty$$

1.4 Error plateaus at machine precision

For several of the observed x values and after reaching a sufficient n, the error seems to stop improving and repeats around 10^{-16} or 0.0 due to rounding. This is expected because the theoretical truncation error becomes smaller than floating-point precision is able to represent.

The plot and the summary tables are provided in Appendix A (Figure 1, Tables 1–2).

2 Nonuniform Spacing of Floating-Point Numbers

The objective is to determine the smallest positive integer n such that n cannot be represented exactly in single precision (float32), and to provide evidence by inspecting nearby values.

I used a classic Binary Search over a Monotone Predicate to search for n in $O(\log n)$ time: Define $P(x)$: "x is unrepresentable as float32" (in Rust: `(x as f32) as u64 != x`).

For nonnegative integers, $P(x)$ is monotone (true for every integer), up until the threshold where it is no longer true for every integer. This allows me to use Binary Search to find:

$$n = 16,777,217$$

as the smallest positive integer that cannot be represented as an integer using single precision floating point. Upon printing the neighbors of this n:

$16,777,215 \quad 16,777,216 \quad 16,777,216 \quad 16,777,218 \quad 16,777,220 \quad 16,777,220$

We discover that $n = 16,777,219$ is also unable to be represented.

A Plots and Tables

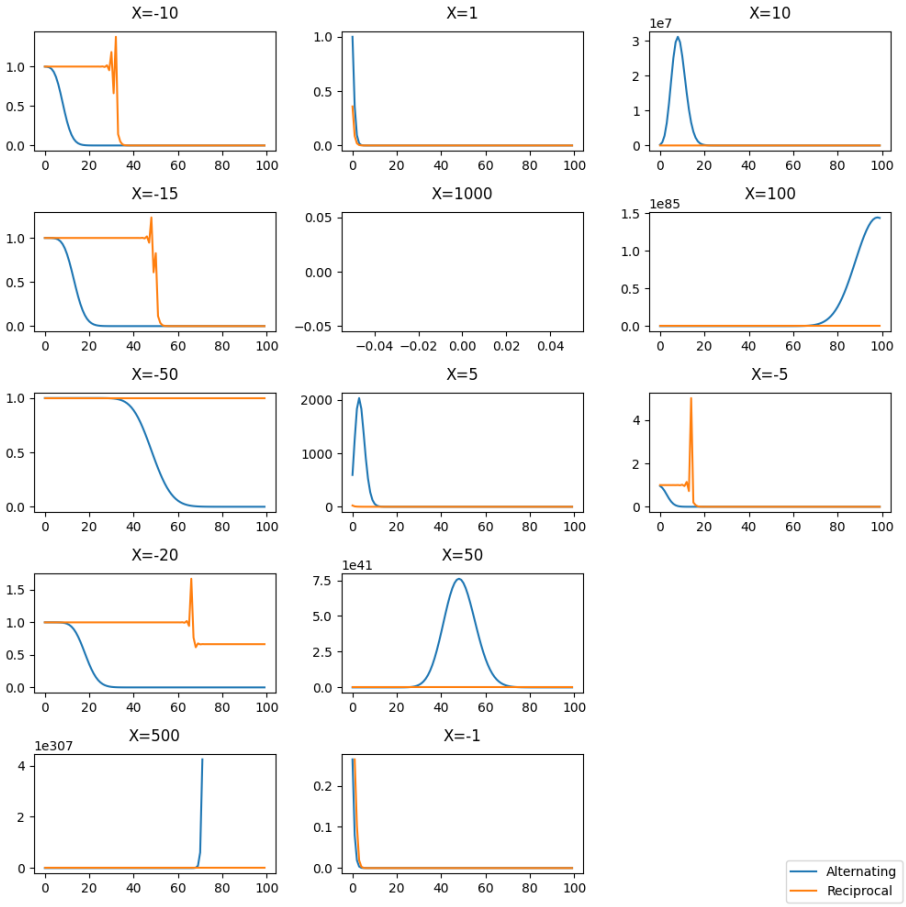

Figure 1: Relative Error for various x using Alternating and Reciprocal Approximations for e^{-x} for $n \in \{1..100\}$.

Table 1: Relative error for Alternating series approximation of e^{-x} at selected n.

x	$n = 1$	$n = 5$	$n = 10$	$n = 50$	$n = 100$
-50	1.00	1.00	1.00	4.62×10^{-1}	1.57×10^{-10}
-20	1.00	1.00	9.89×10^{-1}	4.83×10^{-9}	3.69×10^{-16}
-15	1.00	9.97×10^{-1}	8.82×10^{-1}	2.64×10^{-13}	0.00
-10	9.99×10^{-1}	9.33×10^{-1}	4.17×10^{-1}	1.65×10^{-16}	1.65×10^{-16}
-5	9.60×10^{-1}	3.84×10^{-1}	1.37×10^{-2}	0.00	0.00
-1	2.64×10^{-1}	5.94×10^{-4}	1.00×10^{-8}	1.63×10^{-16}	1.63×10^{-16}
1	1.00	3.30×10^{-3}	6.28×10^{-8}	3.02×10^{-16}	3.02×10^{-16}
5	5.95×10^{2}	1.83×10^{3}	1.27×10^{2}	1.23×10^{-13}	1.23×10^{-13}
10	1.98×10^{5}	1.19×10^{7}	2.96×10^{7}	2.02×10^{-9}	2.03×10^{-9}
50	2.54×10^{23}	1.23×10^{28}	1.16×10^{32}	7.53×10^{41}	1.45×10^{33}
100	2.66×10^{45}	2.13×10^{51}	6.73×10^{56}	5.88×10^{78}	1.44×10^{85}
500	7.00×10^{219}	3.62×10^{228}	3.70×10^{237}	3.73×10^{287}	∞
1000	∞	∞	∞	∞	∞

Table 2: Relative error for Reciprocal series approximation of e^{-x} at selected n.

x	$n = 1$	$n = 5$	$n = 10$	$n = 50$	$n = 100$
-50	1.00	1.00	1.00	1.00	1.00
-20	1.00	1.00	1.00	1.00	6.65×10^{-1}
-15	1.00	1.00	1.00	6.10×10^{-1}	2.51×10^{-5}
-10	1.00	1.00	1.00	2.02×10^{-9}	2.03×10^{-9}
-5	1.00	1.00	9.92×10^{-1}	1.23×10^{-13}	1.23×10^{-13}
-1	∞	3.31×10^{-3}	6.28×10^{-8}	3.27×10^{-16}	3.27×10^{-16}
1	3.59×10^{-1}	5.95×10^{-4}	1.00×10^{-8}	1.51×10^{-16}	1.51×10^{-16}
5	2.37×10^{1}	6.23×10^{-1}	1.39×10^{-2}	0.00	0.00
10	2.00×10^{3}	1.39×10^{1}	7.15×10^{-1}	0.00	0.00
50	1.02×10^{20}	1.80×10^{15}	1.55×10^{11}	8.60×10^{-1}	1.57×10^{-10}
100	2.66×10^{41}	3.07×10^{35}	8.79×10^{29}	4.16×10^{7}	8.99×10^{-1}
500	2.80×10^{214}	5.34×10^{205}	5.11×10^{196}	4.33×10^{146}	1.33×10^{105}
1000	∞	∞	∞	∞	∞