DERIVATIVES AND APPROXIMATIONS: THE CASE OF THE FUNCTION $f(x) = e^x$ AND THE AFFINE APPROXIMATION $\ell(x) = x + 1$.

Humberto José Bortolossi and Luciana Prado Mouta Pena

Department of Applied Mathematics, Universidade Federal Fluminense

Abstract

This text explores, from a numerical perspective, the use of the affine function $\ell(x) = x+1$ to approximate the function $f(x) = e^x$, including values of x not so close to $p = 0$. The exposition is visually and graphically implemented in GeoGebra, allowing for an interactive understanding of the approximation.

Keywords: calculus, affine approximation, exponential function, GeoGebra.

1 First-order Derivative & Affine Approximation in the Neighborhood of a Point

In Calculus I courses, we learn that derivatives allow for the construction of simpler functions from potentially more complex differentiable functions. Let's look at a classic example: consider the exponential function

$$f(x) = e^x,$$

with $e = 2.718281828459045\ldots$ the Euler constant (an irrational number, the base of natural logarithms)[1].

Evaluating $f(x) = e^x$ for different numerical values of x can be challenging. This involves issues of precision, speed, and the operations that a computer's *hardware* can perform (essentially strict versions of the four arithmetic operations). In Calculus, we show that the *linearization*

$$\ell(x) = f(p) + f'(p)(x - p)$$

is the *best* affine approximation of the function f near the point p. Certainly, the function ℓ is much easier to evaluate for different values of x. Here, *best* is in the sense of minimizing an expression for the approximation error, which can be established by Taylor's Theorem. Let's delve into the details: for a differentiable function f in an open interval I, with $p \in I$, by Taylor's Theorem, for any $x \in I$ we have

$$f(x) = \underbrace{f(p) + f'(p)(x - p)}_{\ell(x)} + R(x),$$

[1]The universality of the exponential function in both pure and applied mathematics led the renowned mathematician Walter Rudin to call it the "most important function in Mathematics."

where $f(p)$ and $f'(p)$ are the values of f and its derivative f' at p, and the remainder $R(x) = f(x) - \ell(x)$ measures the approximation error. Furthermore, there exists a function h such that

$$R(x) = f(x) - \ell(x) = h(x)(x-p), \text{ with } \lim_{x \to p} h(x) = 0.$$

Now, consider any other affine function $g(x) = m\,(x-p) + f(p)$ whose graph (a line!) passes through the point $(p, f(p))$ of the graph of f, here m is a real number that determines the slope of the line.

The approximation error of the function $f(x)$ by this other affine function g at a point x is given by:

$$E(x) = \Big| f(x) - \underbrace{[m(x-p) + f(p)]}_{g(x)} \Big|. \tag{1}$$

Using Taylor's Theorem, we can write:

$$f(x) = \underbrace{f(p) + f'(p)(x-p)}_{\ell(x)} + R(x)$$

Substituting in (1), we get:

$$E(x) = \Big| \underbrace{f(p) + f'(p)(x-p)}_{\ell(x)} + R(x) - \big[m(x-p) + f(p)\big] \Big|$$

simplifying the expression, we obtain:

$$E(x) = \Big| \,(f'(p) - m)(x-p) + R(x) \,\Big|.$$

We then see that the approximation error will be minimized when the slope of the line m equals the slope of the tangent, that is, when $m = f'(p)$. In this case, the approximation error is reduced to the remainder term $R(x)$, which tends to zero as x approaches p.

Conclusion: Taylor's Theorem demonstrates that the tangent line to the graph of the function f at the point $(p, f(p))$ is the best affine approximation of the function at that point, in the sense of minimizing the approximation error.

In our example, $f(x) = e^x$. Let's choose $p = 0$, a point where it is easy to calculate $f(x) = e^x$ and its derivative: $f'(x) = f(x) = e^x$. Thus, we have: $p = 0, f(p) = f'(p) = e^0 = 1$ and $\ell(x) = f(0) + f'(0)(x-0) = 1 + x$. Note, if x is close to p, then $\ell(x)$ is close to $f(x)$.

Let's look at a numerical example:

$$e^{0.01} = f(0.01) \;\approx \ell(0.01) \;=\; 1 \;+ 0.01 \;=\; 1.01.$$

A common scientific calculator gives: $e^{0.01} = 1.01005016\ldots$ The linear approximation yields three correct decimal places, with the added bonus that computing the function $y = \ell(x)$ requires only a single arithmetic operation.

2 Approximations for other values of x

When x is not near 0, for example, $x = 1\ldots$ In the specific case of the function $f(x) = e^x$, we can use the following smart trick:

$$e^1 = \left(e^{0.5}\right)^2 = \left(\left(e^{0.25}\right)^2\right)^2 = \left(\left(\left(e^{0.125}\right)^2\right)^2\right)^2 = \left(\left(\left(\left(e^{0.0625}\right)^2\right)^2\right)^2\right)^2 = \cdots .$$

Proceeding with this idea we arrive at, for example,

$$e^1 = \left(\left(\left(\left(\left(\left(\left(\left(\left(\left(e^{(0.00048828125)}\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2$$

Now 0.00048828125 is a number closer to 0 and we can use the affine approximation $\ell(x) = x+1$:

$$e^{0.00048828125} \approx 1 + 0.00048828125 = 1.00048828125$$

Substituting this approximation in the previous expression, we obtain that

$$e^1 \approx \left(\left(\left(\left(\left(\left(\left(\left(\left(\left((1.00048828125)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2\right)^2$$
$$= 1.00048828125^{2048},$$

which gives us

$$e^1 \approx 2.7176184823368797609$$

a scientific calculator provides

$$e^1 \approx 2.7176184823368796$$

Note that we are using a recursive procedure: $u(x) :=$ if ($x < 0.001$, return $1 + x$, otherwise return $(u(x/2))^2$).

3 TheGeoGebra Applet

The technique described in the previous section was implemented in the GeoGebra Applet available at `https://www.geogebra.org/m/twdraccq` (Figure 1). In this applet, the slider n determines the recursion depth of the function r. The slider δ determines the interval $(-\delta, \delta)$ in which the affine approximation $\ell(x) = 1 + x$ is used. Besides the Recursive Affine Approximation, the applet displays the graph of $f(x) = e^x$ as natively drawn by GeoGebra. The Taylor polynomial of order k is also displayed for comparison purposes. Finally, it is also possible to numerically compare the values of all these functions for different values of x. The value of x can be changed by moving the point named P.

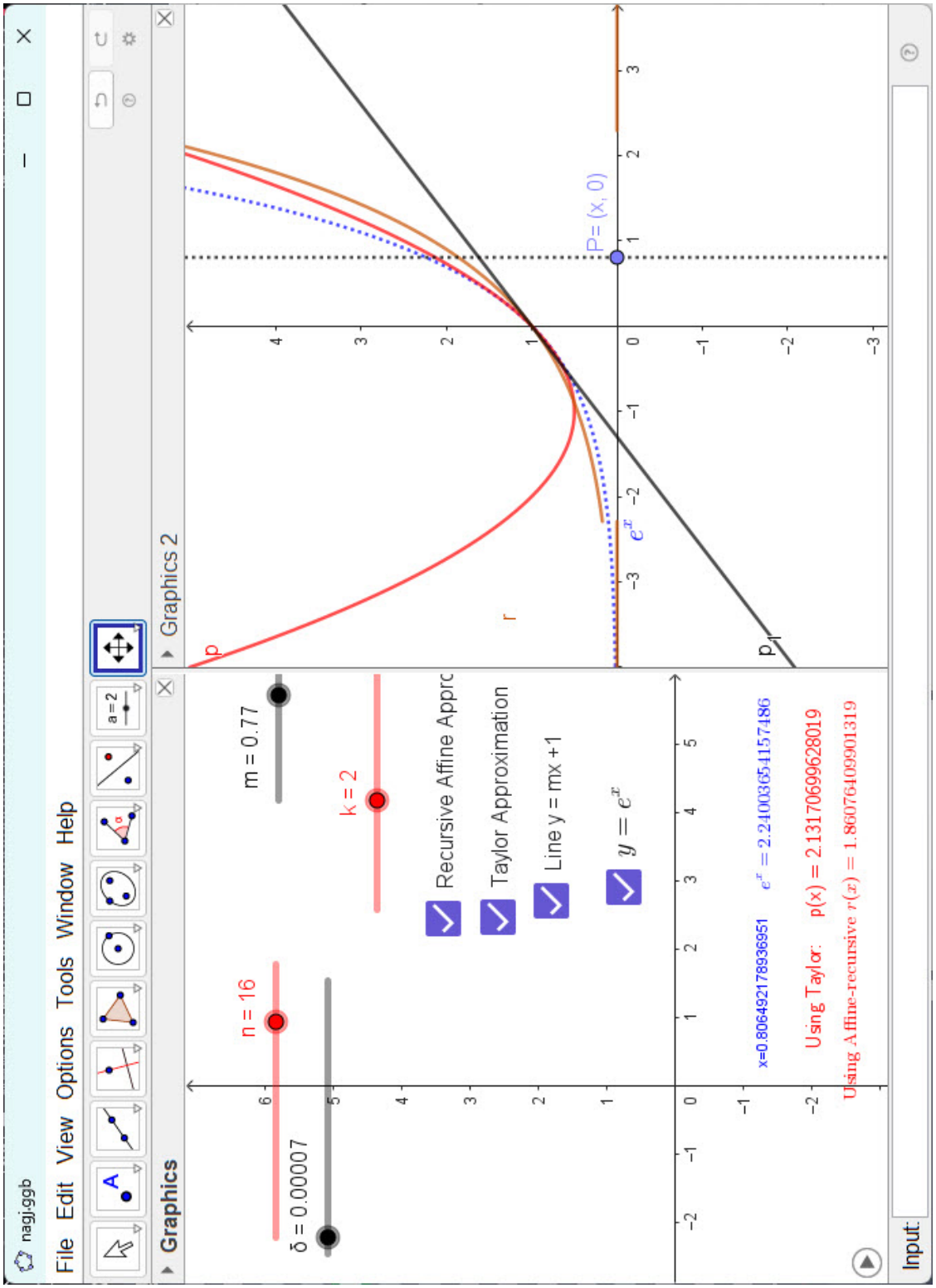

Figure 1. GeoGebra Applet with Recursive Affine Approximation of the Function $f(x) = e^x$.
Source: `https://www.geogebra.org/m/twdraccq`

4 Final Considerations

The IEEE (*Institute of Electrical and Electronics Engineers*) does not directly specify a single way to implement the calculation of elementary functions such as the exponential function $f(x) = e^x$ in *hardware*, leaving this task to each CPU manufacturer. Several algorithms have been proposed:

1. *Polynomial or rational approximations:* Many algorithms use polynomial or rational approximations of elementary functions. This is especially useful for functions that cannot be calculated directly in an exact manner, such as sine, cosine, and logarithms.

2. *Pre-calculated tables:* Some CPUs use tables of pre-calculated values combined with correction methods to refine the function value. This is done to speed up the calculation of complex elementary functions, reducing the number of necessary arithmetic operations.

3. *CORDIC algorithms* (*Coordinate Rotation Digital Computer*): Widely used for calculating trigonometric and hyperbolic functions efficiently, CORDIC is also employed for calculating exponents and logarithms. It relies on simple iterations of addition and shifting.

4. *Newton-Raphson iterations*: This iterative method is used for calculations such as division and square root, providing high precision in few iterations.

5. *Factorization and decomposition methods*: For functions like e^x, breaking down into smaller factors[2] makes it easier to approximate in larger values of x, which improves the accuracy of the calculation with fewer operations.

For a general overview of these techniques, we refer to the source Muller (2006). The monograph Detrey et al. (2007) notes that:

> The study of specific hardware circuits for evaluating elementary functions in floating-point was once an active research area, until it was realized that these functions were not frequent enough to justify dedicating silicon to them. Research then shifted towards functions implemented in software. However, this situation may be about to change again with the advent of reconfigurable coprocessors based on FPGAs (field-programmable gate arrays). These coprocessors now have capabilities that allow accommodating double-precision floating-point calculations. Hardware operators for elementary functions, targeted at these platforms, have the potential to significantly outperform software functions and will not permanently waste silicon resources.

Acknowledgments

The authors extend their profound gratitude to Professors Carlos Tomei and Marco Moriconi for their meticulous review, invaluable suggestions, and critical corrections to the manuscript. We further wish to express our sincere appreciation to the anonymous referees for their rigorous evaluation and constructive feedback, which substantially improved the quality of this work. Additionally, we acknowledge with gratitude the diligent oversight and guidance provided by the editorial board throughout the publication process.

[2]The technique we used in our example.

References

Detrey, J., Dinechin, F. d., and Pujol, X. (2007). Return of the hardware floating-point elementary function. In *18th Symposium on Computer Arithmetic*, pages 161–168, Montpellier, France.

Muller, J.-M. (2006). *Elementary Functions: Algorithms and Implementation*. Birkhäuser, New York, 2 edition.

Humberto José Bortolossi is an Associate Professor at Universidade Federal Fluminense and Chair of the international GeoGebra Institute at Rio de janeiro.

Luciana Prado Mouta Pena is an Associate Professor at Universidade Federal Fluminense.