15-11-2023

Numerical Analysis

Finding Roots

Haka Kevin

Introduction

One of the biggest challenges for scientists is finding the roots of a function that explains a natural phenomenon. Roots are very important because they help us better understand the behavior of functions and by extension, the interpretation of the phenomenon. For example, roots can tell us where the function changes sign or where the function turns direction (root of first derivation) and more.

Unfortunately, the roots' founding analytically is possible only for a very small percentage of function, for all the other functions we can only approximate the true value with numerical methods.

In this work, the equation $e^x + x - 2 = 0$ will be numerically solved using three popular methods: the bisection method, the regula falsi method, and the fixed-point iteration method. Before delving into the results of these methods, their capabilities and limitations will be briefly analyzed theoretically. The graphical method will be used to estimate the initial estimates of each method. Finally, the results will be compared and discussed.

Defining the tolerance error upfront is crucial as it is used in any approximation method. The tolerance error is necessary because all numerical algorithms approximate the true value as the number of iterations tends to infinity. However, this is not a realistic scenario, as even the faster computers are limited by memory and time, which tend to infinity for an infinite number of iterations. Therefore, **Scarborough** and his formula (1) came up to solve this problem. The **'n'** in the Scarborough formula assures us that if the criterion $\varepsilon_\alpha < \varepsilon_s$ is true, we have at least 'n' significant figures. The formula (2) shows us the relative change of the two last approximation values.

$$\varepsilon_s = \frac{1}{2} 10^{-n} \tag{1}$$

$$\varepsilon_{\alpha} = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \tag{2}$$

Theoretical solution

$$e^x + x - 2 = 0 \Leftrightarrow (e^x + x - 2) * e^{-x} = 0$$

$$1 + (x - 2) * e^{-x} = 0 \Leftrightarrow (2 - x) * e^{-x} = 1$$

$$e^2 * (2 - x) * e^{-x} = e^2 \Leftrightarrow (2 - x) * e^{2-x} = e^2$$

$$W\big((2 - x) * e^{2-x}\big) = W(e^2) \Leftrightarrow 2 - x = W(e^2)$$

$$x = 2 - W(e^2) \Leftrightarrow x \approx 0.442854401$$

The Lambert W function is multivalued function. However, when dealing with real numbers only, the two branches W_0 and W_{-1} suffice. For $W(x)$ if $x \in [-\frac{1}{e}, 0)$ it has two solution and if $x \geq 0$ it has only one, as in the above case.

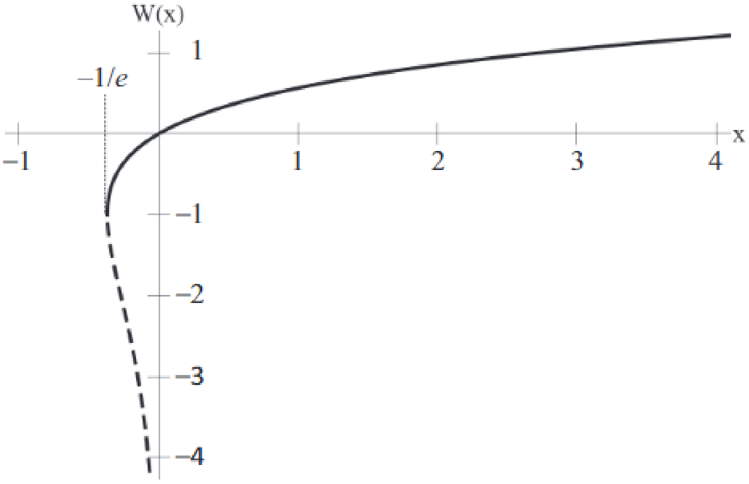


Figure 1. Real values of the Lambert W function. The solid curve is the principal branch (W_0 or W), and the dashed curve is the -1 branch (W_{-1}).

Methodology

With the graphical method, we can represent the function $f(x) = e^x + x - 2$ graphically, as shown below (Figure 2). The graph helps us to make a good choice for our initial guess, and to see the possible weird behavior of the function, in that way we will avoid unwanted results. As we can see, the root lies between 0 and 1 in our function. This interval will be used for the two bracket methods, bisection and the regula falsi method. For the fixed-point iteration method, which is an open method, we only need one initial guess, and we select 0.5 because it seems to be close to the root.

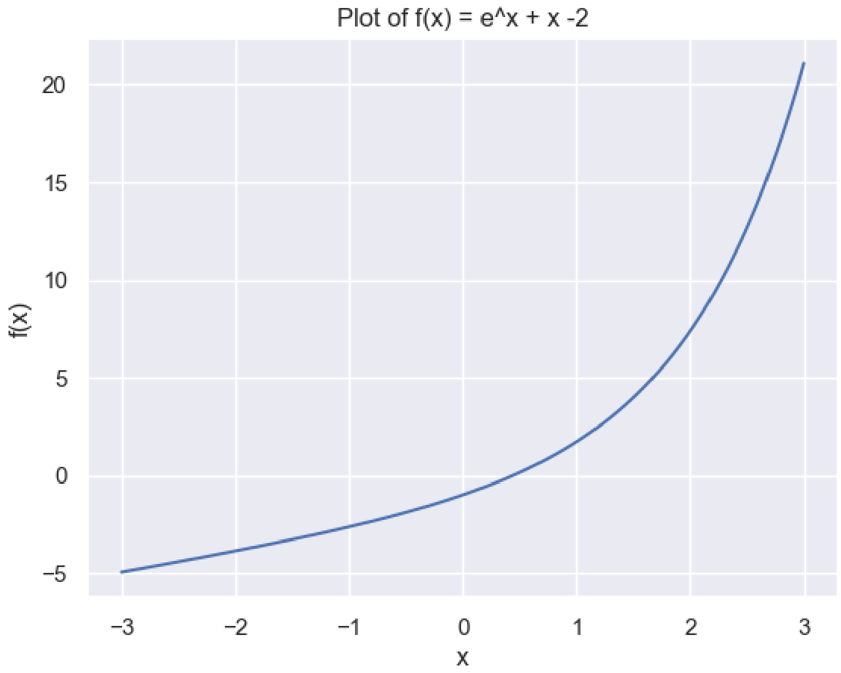


Figure 2. This plot shows the behavior of the function f(x) = e^x + x – 2, over the interval [-3, 3].

The first method we will discuss is the **bisection** method, also known as Bolzano's method, which is founded on the Bolzano theorem. The basic idea is very simple: If we have a continuous function f(x) defined over the closed interval [a, b] and f(a) and f(b) have opposite signs, then there exists at least one root of f(x) within the interval [a, b].

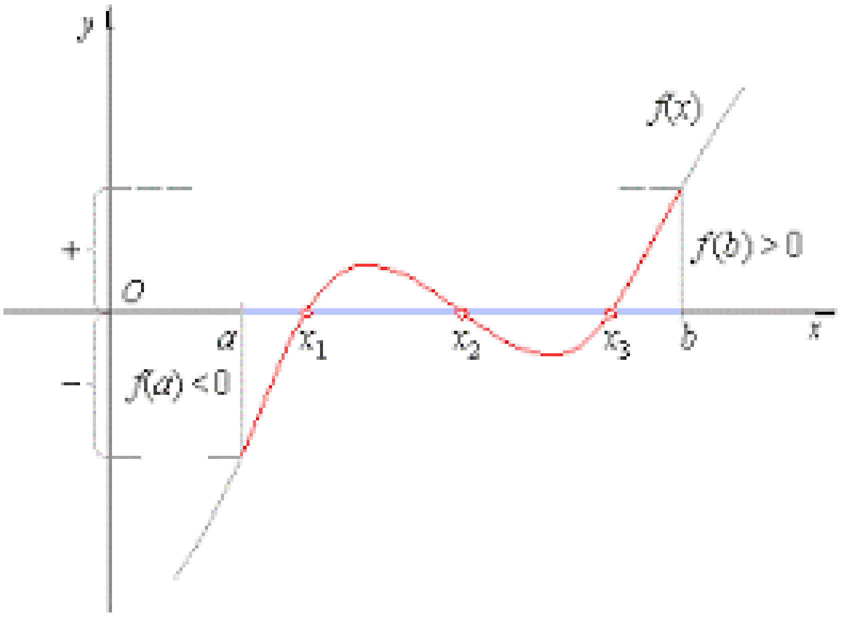


Figure 3. A graphical representation of the Bolzano theorem.

The steps of the algorithm are as follows:

1) We take the interval [a, b] and divide it into two halves, [a, x_r] and [x_r, b], by finding the value of x_r, which is the center of the interval.

$$x_r = \frac{(a\ +\ b)}{2} \tag{3}$$

2) After step 1, we perform some checks:
    a. If f(x_r) = 0 we have found the root and everything stops there.
    b. Else if f(a) * f(x_r) < 0 (Bolzano's theorem), the root is in the [a, x_r] interval. In this case, the new interval will be [a, b=x_r].
    c. Else, the root is in the remaining interval [x_r, b]. In this case, the new interval is [a=x_r, b].

3) Last step is to check if the relative error ε_α of our new root x_r in comparison with the previous root is smaller than a tolerance error ε_s.

    a. If $\varepsilon_\alpha < \varepsilon_s$, then stop and we have a good approximation of the root.

    b. Else, go to step 1.

Based on the property of interval subdivision in each iteration, it can be proven that the maximum number of iterations is given by the function (4), where a and b are the bounds of the initial interval [a, b]. The estimation below is not sufficient if the root is in the interval $(-1, 1)$ because the Scarborough criterion is more conservative.

$$n = log_2 \frac{b - a}{\varepsilon_s} \tag{4}$$

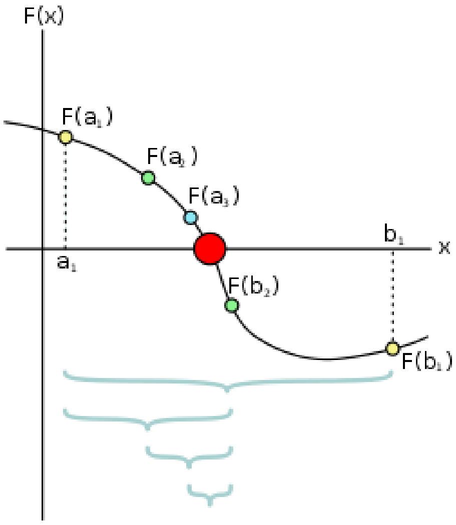


Figure 4. A few steps of the bisection method applied over the starting interval [a_1, b_1]. The bigger red dot is the root of the function.

Bisection is a very conservative and easy-to-understand method. If there is a root over the interval, it will find it. However, in comparison with the other methods that we will show next, it is the slowest. The limitation for this method is that the function must be continuous over the interval and have opposite signs at the bounds.

The second method that will discase is the regula falsi method also known as false position method. It is a bit more complex than bisection but it is generally more efficient. The limitations, conditions and generally the whole algorithm is the same as the bisection. The only thing that changes is the way that approximate and estimate the root in every iteration (Figure 5).

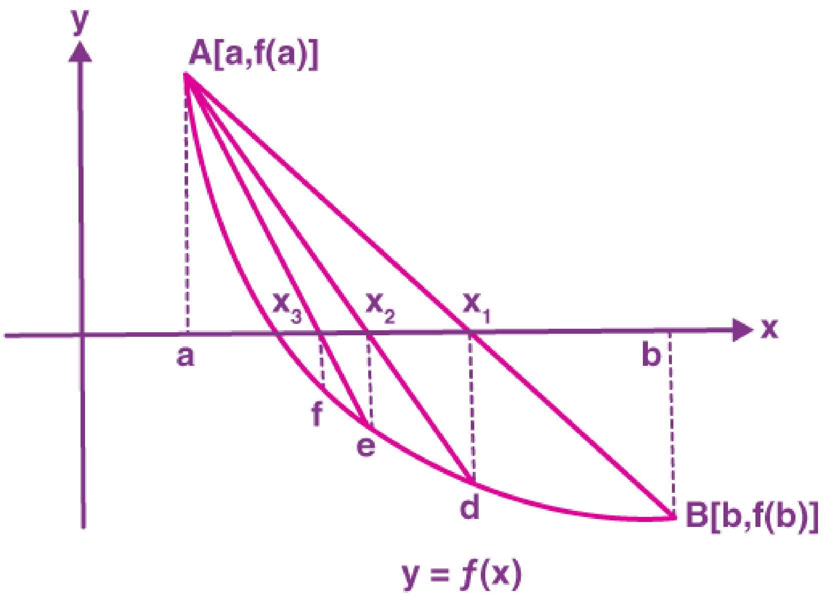


Figure 5. A few steps of the regula falsi method applied over the starting interval [a, b].

The underlying principle of the method involves drawing a line from f(a) to f(b). This line intersects the x-axis at a point, which we will be our new root estimate x_r. After determining x_r, we calculate f(x_r) and repeatedly replace one of the interval's endpoints with our new x_r, as specified in step 2 of the bisection algorithm. This process continues until x_r converges to the root, such as in step 3 of the bisection algorithm. There are two basic methods to determine x_r, which lead to two different forms of it. The first method involves using the line equation $y = m * x + c$, which gives us equation (5). The other method involves using similar triangles, which gives us equation (6). The two forms are equal, but we will use equation (6) due to its computational advantage for our algorithm. As a solution is approached, a and b will be very close together, and nearly always of the same sign. Such a subtraction can lose significant digits. Because f (b) and f (a) are always of opposite sign the “subtraction” in the numerator of the equation (6) is effectively an addition (as is the subtraction in the denominator too).

$$x_r = b - f(b)\frac{b-a}{f(b)-f(a)} \tag{5}$$

$$x_r = \frac{af(b)-bf(a)}{f(b)-f(a)} \tag{6}$$

In this work, we are not going to use the regula falsi method directly but a modified version of it known as the Illinois algorithm. This algorithm improves the regula falsi method when the absolute values of our function f at the bounds have a big difference $\left||f(a)|-|f(b)|\right| \gg 0$. This problem drives the classical method to very slow convergence because the algorithm replaces only one of the two endpoints. The Illinois algorithm solves this problem by downweighting the value of the stable endpoint. It achieves this by inserting counters at 2.b and 2.c of the bisection algorithm, which control how many iterations in a row an endpoint didn’t change. If the counter counts more than a predefined number, then at the new iteration, the stable endpoint will be downweighted to force it to change. The most common limit for the counters is 2, and for the downweight, it is 0.5. However, theoretically, the limit of the counters can be any natural number ($\mathbb{N}$) except zero, and the downweight takes values over an interval (0, 1], where 1 converges to the classical bisection method.

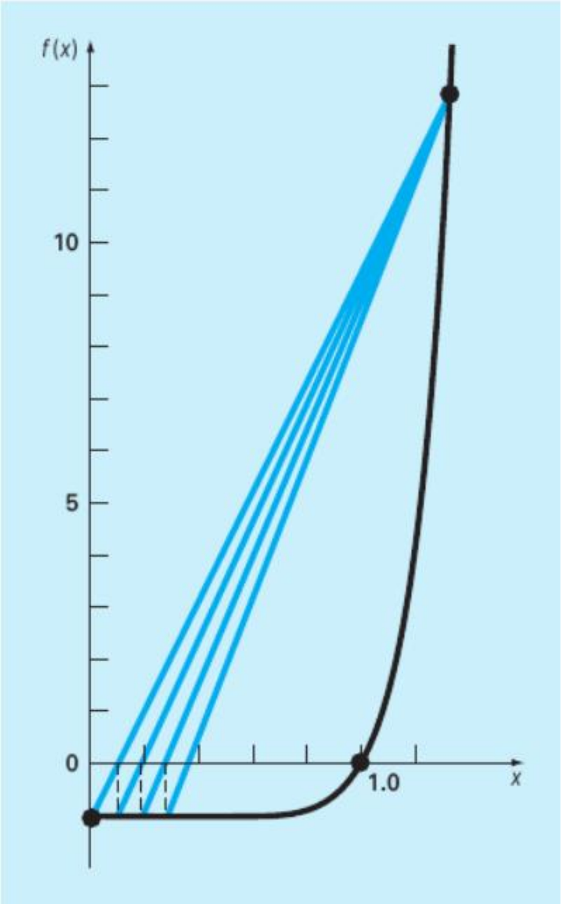


Figure 6. Slow convergence of the regula falsi method due to a stable point.

The third and last method that we will use in this work is an open method named fixed point iteration method. Open methods are generally based on formulas that require only a single starting value of x or two starting values that do not necessarily bracket the root, such as bracket methods do. In most cases, open methods are much faster than bracket methods, but the understanding and conditions required to work with an open method are more complex.

For the fixed-point method, the idea is that we can move the target from searching for the roots of a function f(x) to searching for the roots of the equation x=g(x) (Figure 7), which is just the rearrangement of f(x)=0 so that x is on the left-hand side of the equation x=g(x).

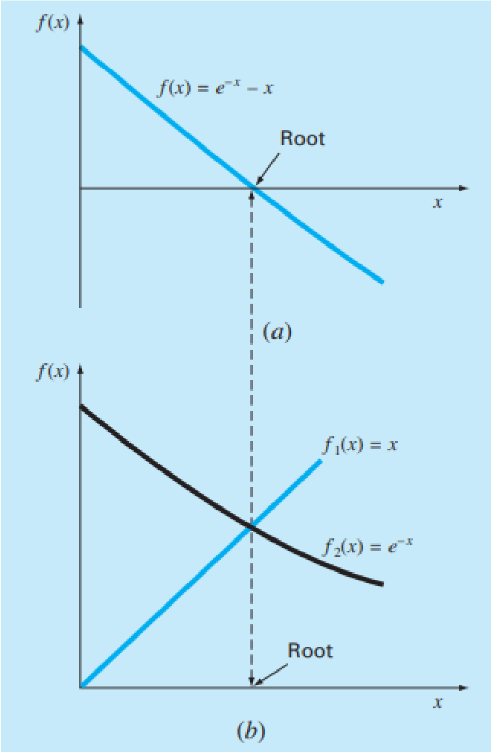


Figure 7. Two alternative graphical methods for determining the root of $f(x) = e^{-x} - x$.
(a) Root at the point where it crosses the x axis, (b) root at the intersection of the component functions.

The algorithm is very simple and straightforward. The only thing that we need to do is calculate the equation xi = g(xi) in each iteration, as we can see in Figure 8 (a) and (b), until the Scarborough criterion is true. However, things are not as simple as they look because the iteration doesn't always converge, as we can see in Figure 8 (c) and (d). The reason for this and the condition for the convergence of this method is that the absolute value of the first derivative of g(x) has to be smaller than 1 for each x over the interval that we are searching for the root ($|g'(x)| < 1$). With the mean-value theorem, we can prove that the first derivative of g(x) is the rate of error in each iteration ($\varepsilon_{i+1} = g'(x) * \varepsilon_i$). That's why if $|g'(x)| > 1$, the error increases in each iteration, and the method deviates.

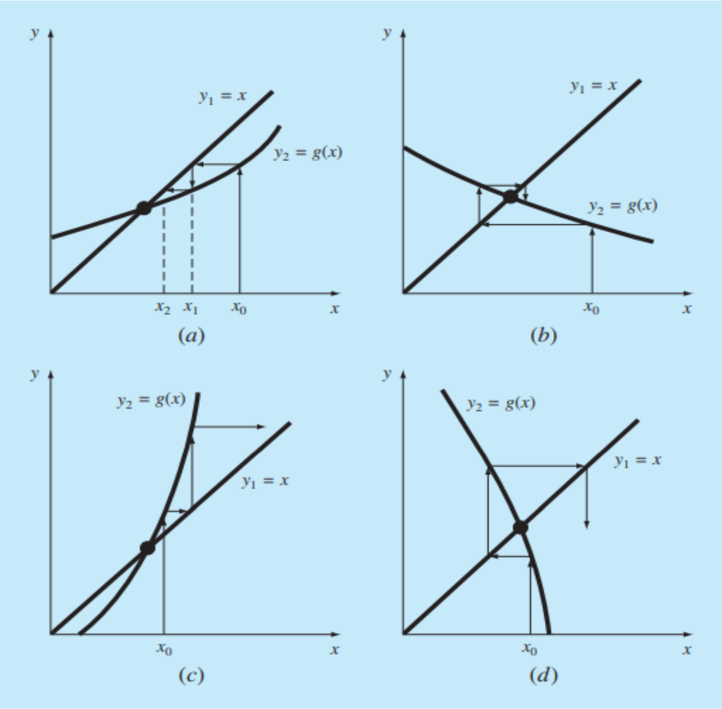


Figure 8. Iteration cobwebs depicting convergence (a and b) and divergence (c and d) of simple fixed-point iteration. Graphs (a) and (c) are called monotone patterns, whereas (b) and (d) are called oscillating or spiral patterns.

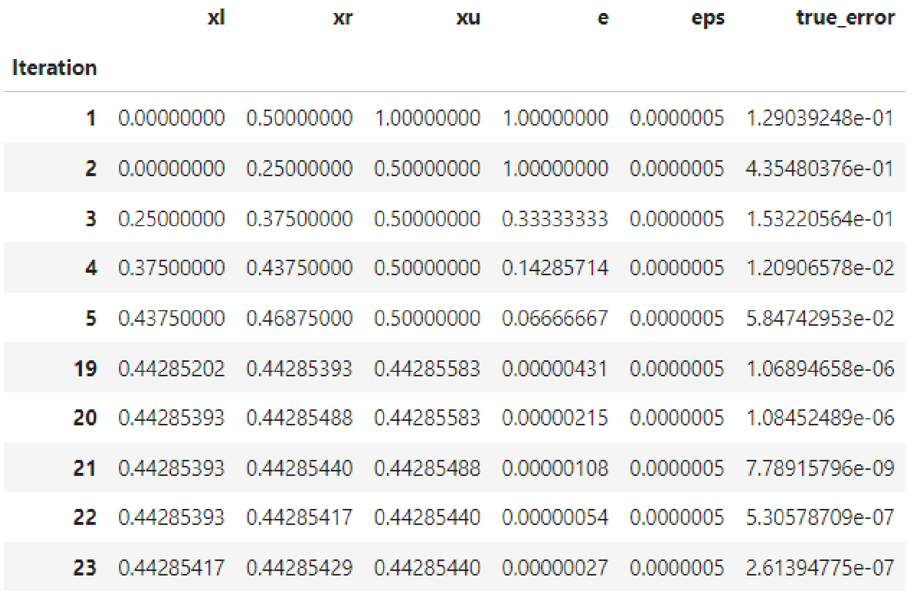
Results and Discussion

The results shown below have been computed using the Python programming language. The code for the three numerical methods discussed earlier, along with their implementation and application to our problem, is provided with this work as a Jupyter Notebook file (.ipynb).

We begin with the bisection method, using the interval [0, 1] as our initial interval and aiming to find the root with 6 significant figures. After running the algorithm, we obtained Table 1, which shows that the algorithm converges after 23 iterations. In the first plot of Figure 9, we can see how the true and approximate errors decrease towards zero, as they would theoretically approach zero as the number of iterations tends to infinity. In the second plot, we can see that the algorithm stops immediately after the Scarborough criterion is met, ensuring the desired accuracy.

It's crucial to note that the theoretical number of iterations (function 4), provides a value close to 21. However, as Table 1 demonstrates, the actual number of iterations is 23. This discrepancy arises because, as explained earlier, our root lies within the interval (-1, 1), where the theoretical iteration number converges too slowly to achieve the desired significant figures. This phenomenon becomes more pronounced as the root approaches zero. For instance, if a root is equal to 0.00123 and we determine the theoretical iteration number, it will provide an iteration number that converges to the root 0.00... However, this is not what we requested. The Scarborough criterion ensures that the answer will be at least 0.0012…

Table 1. Results from the bisection method for the function $f(x) = e^x + x - 2$ with an initial interval of [0, 1] and a 6 significant figures root. This table displays the first and last 5 rows of the data.



Iteration	xl	xr	xu	e	eps	true_error
1	0.00000000	0.50000000	1.00000000	1.00000000	0.0000005	1.29039248e-01
2	0.00000000	0.25000000	0.50000000	1.00000000	0.0000005	4.35480376e-01
3	0.25000000	0.37500000	0.50000000	0.33333333	0.0000005	1.53220564e-01
4	0.37500000	0.43750000	0.50000000	0.14285714	0.0000005	1.20906578e-02
5	0.43750000	0.46875000	0.50000000	0.06666667	0.0000005	5.84742953e-02
19	0.44285202	0.44285393	0.44285583	0.00000431	0.0000005	1.06894658e-06
20	0.44285393	0.44285488	0.44285583	0.00000215	0.0000005	1.08452489e-06
21	0.44285393	0.44285440	0.44285488	0.00000108	0.0000005	7.78915796e-09
22	0.44285393	0.44285417	0.44285440	0.00000054	0.0000005	5.30578709e-07
23	0.44285417	0.44285429	0.44285440	0.00000027	0.0000005	2.61394775e-07

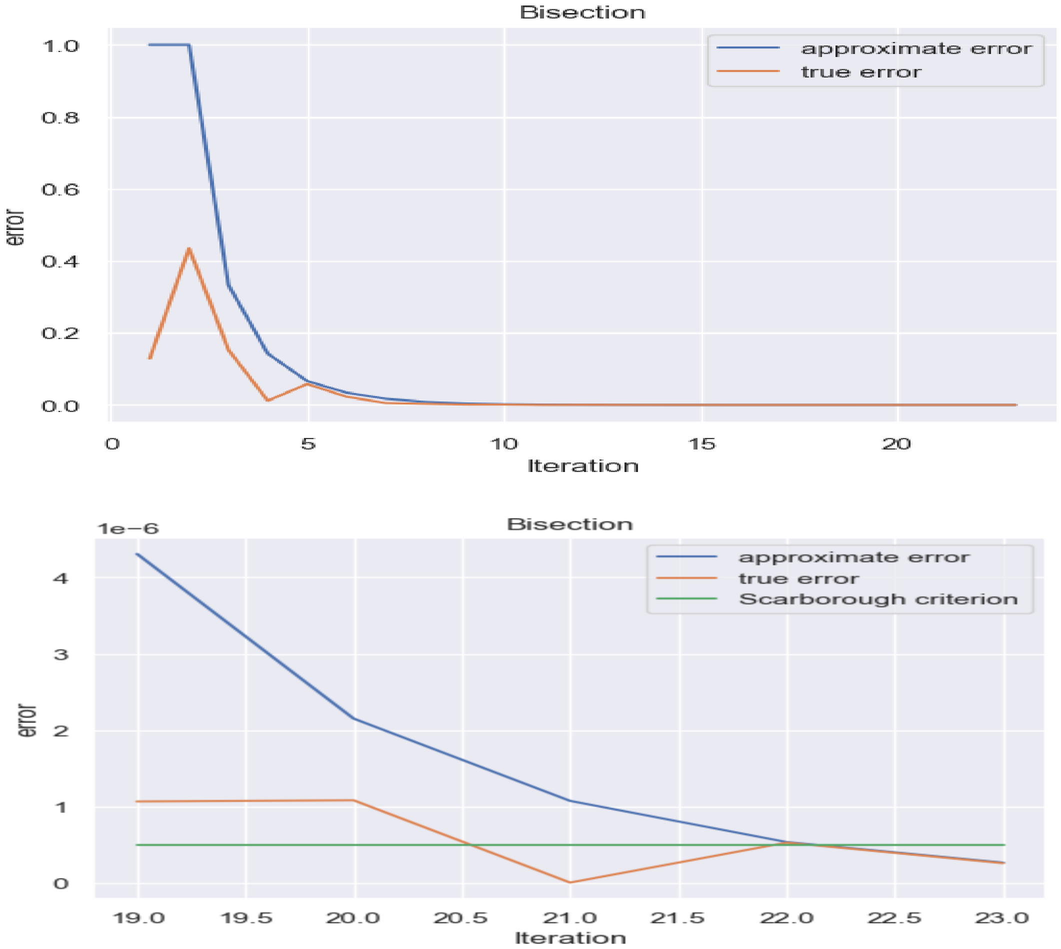


Figure 9. Plots of the true and approximate error with bisection method applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures. The second plot is zoomed in on the last five iterations.

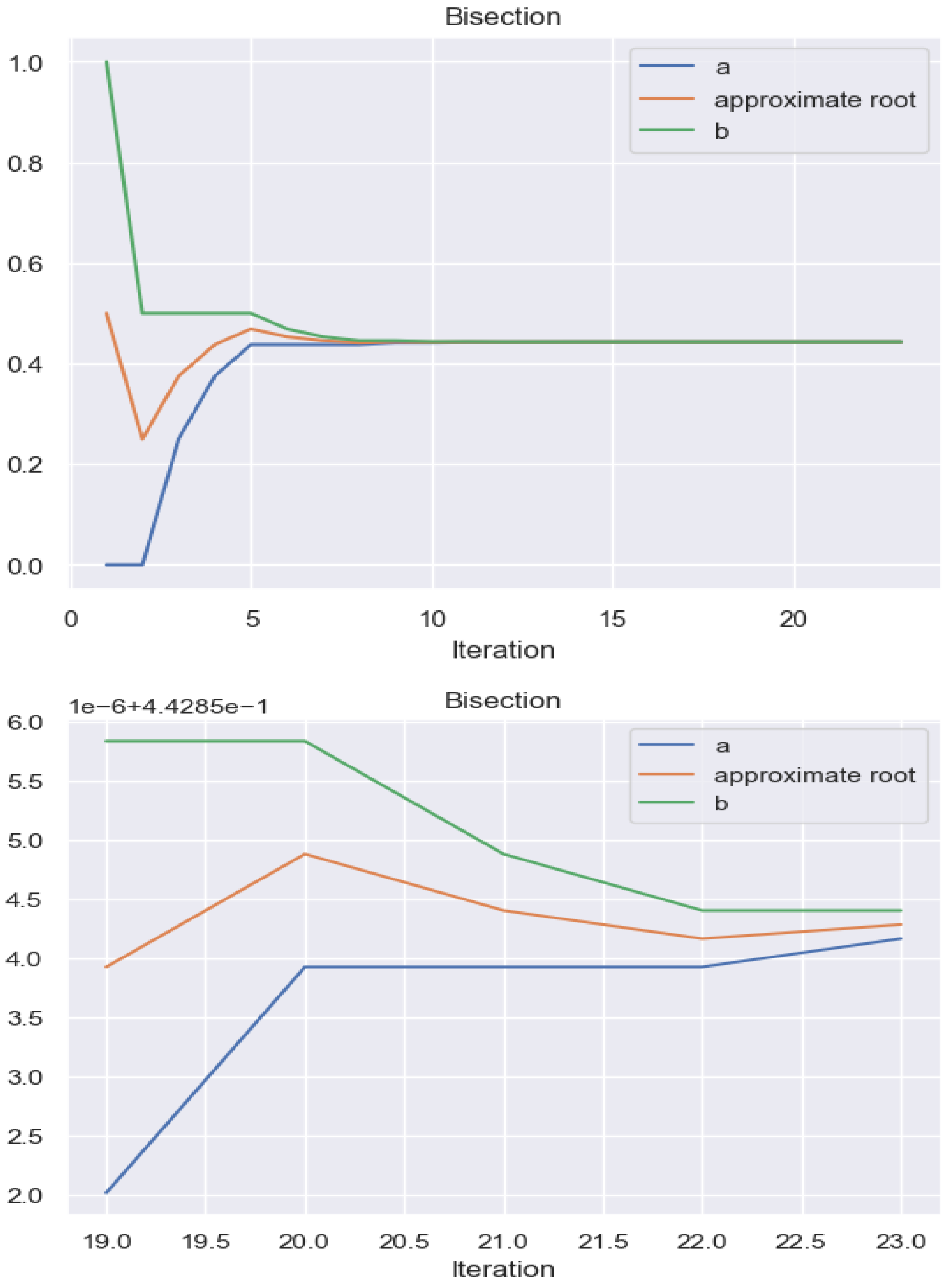
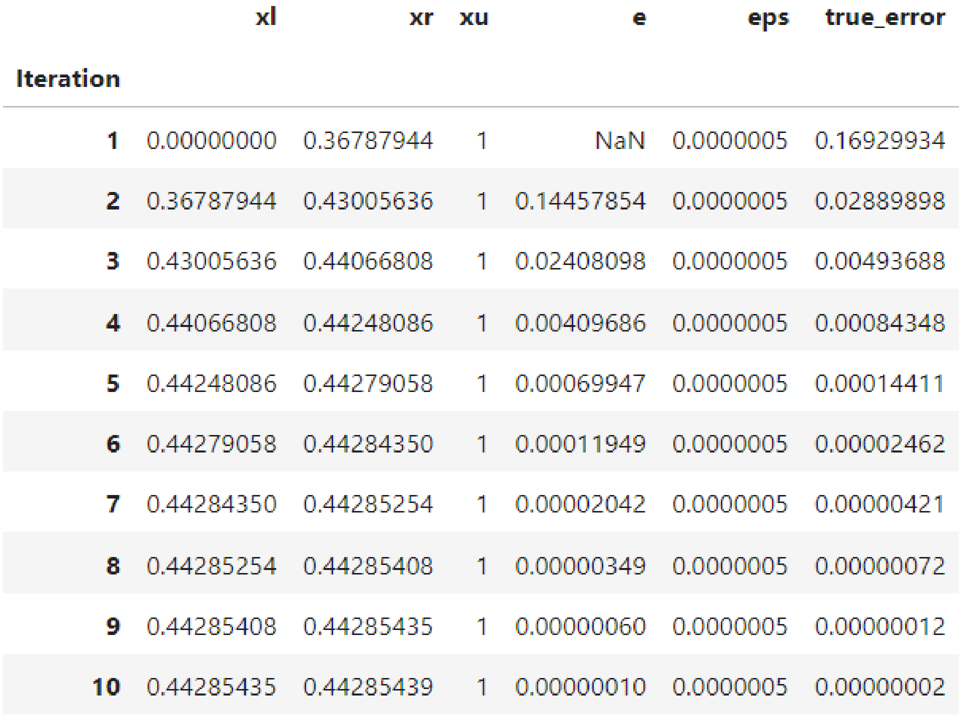


Figure 10. Plots of the approximate root and the interval [a, b] with bisection method applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures. The second plot is zoomed in on the last five iterations.

The regula falsi method converges much faster than bisection, as shown in Table 2, requiring only 10 iterations. However, Figure 12 reveals that one of the endpoints remains stable for all iterations, which is the problem that the Illinois algorithm aims to solve. Table 3 shows that the number of iterations has decreased further and is now 6. The problem is solved in Figure 14, where the stable endpoint at the classical regula falsi method changes after staying stable three times in a row.

Table 2. Results from the Regula Falsi method for the function $f(x) = e^x + x - 2$ with an initial interval of [0, 1] and a 6 significant figures root.



	xl	xr	xu	e	eps	true_error
Iteration						
1	0.00000000	0.36787944	1	NaN	0.0000005	0.16929934
2	0.36787944	0.43005636	1	0.14457854	0.0000005	0.02889898
3	0.43005636	0.44066808	1	0.02408098	0.0000005	0.00493688
4	0.44066808	0.44248086	1	0.00409686	0.0000005	0.00084348
5	0.44248086	0.44279058	1	0.00069947	0.0000005	0.00014411
6	0.44279058	0.44284350	1	0.00011949	0.0000005	0.00002462
7	0.44284350	0.44285254	1	0.00002042	0.0000005	0.00000421
8	0.44285254	0.44285408	1	0.00000349	0.0000005	0.00000072
9	0.44285408	0.44285435	1	0.00000060	0.0000005	0.00000012
10	0.44285435	0.44285439	1	0.00000010	0.0000005	0.00000002

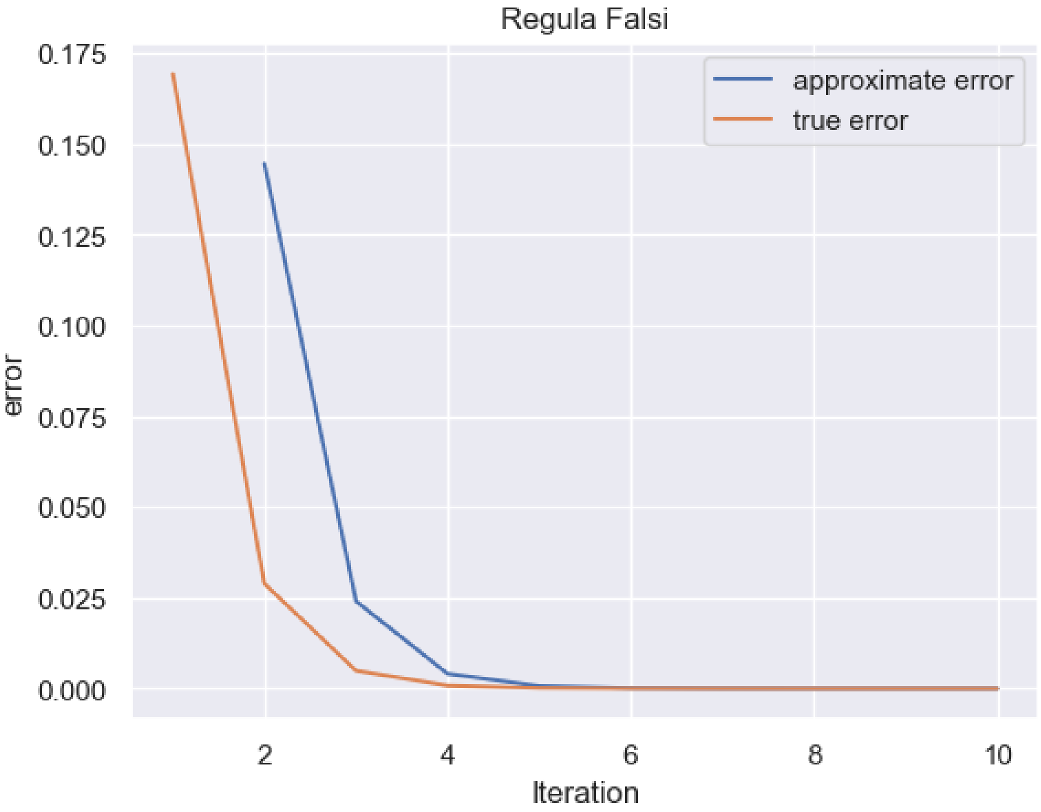


Figure 11. Plots of the true and approximate error with regula falsi method applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures.

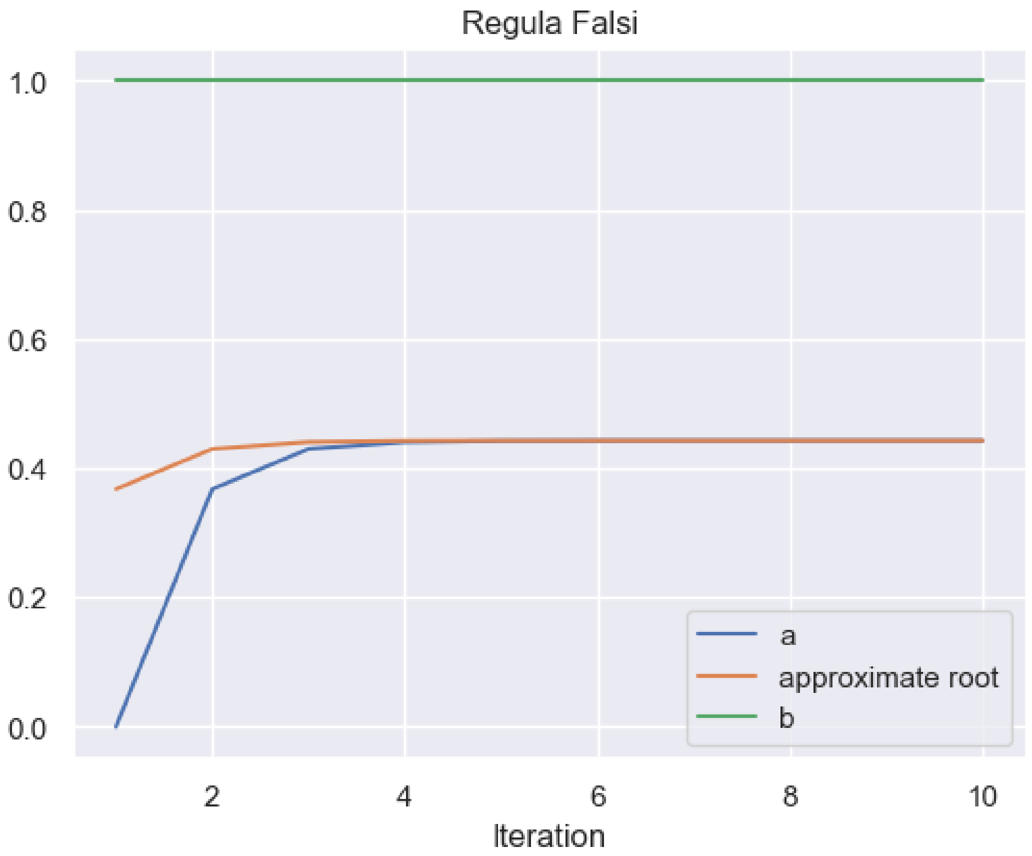
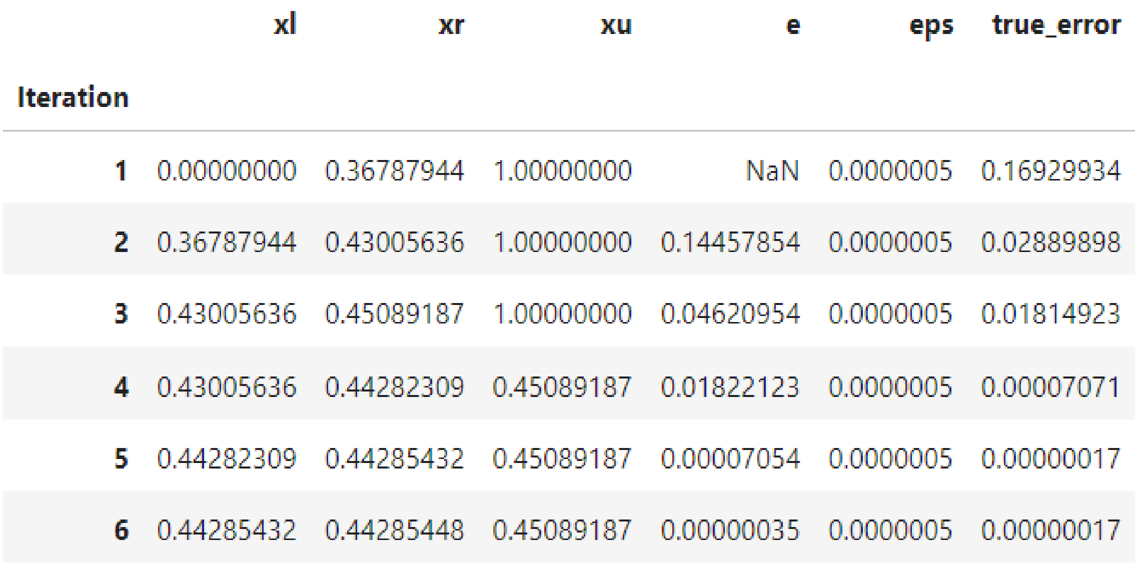


Figure 12. Plots of the approximate root and the interval [a, b] with regula falsi method applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures.

Table 3. Results from the modified regula falsi method (rescaling = 0.5 and counter limit = 2) for the function $f(x) = e^x + x - 2$ with an initial interval of [0, 1] and a 6 significant figures root.



	xl	xr	xu	e	eps	true_error
Iteration						
1	0.00000000	0.36787944	1.00000000	NaN	0.0000005	0.16929934
2	0.36787944	0.43005636	1.00000000	0.14457854	0.0000005	0.02889898
3	0.43005636	0.45089187	1.00000000	0.04620954	0.0000005	0.01814923
4	0.43005636	0.44282309	0.45089187	0.01822123	0.0000005	0.00007071
5	0.44282309	0.44285432	0.45089187	0.00007054	0.0000005	0.00000017
6	0.44285432	0.44285448	0.45089187	0.00000035	0.0000005	0.00000017

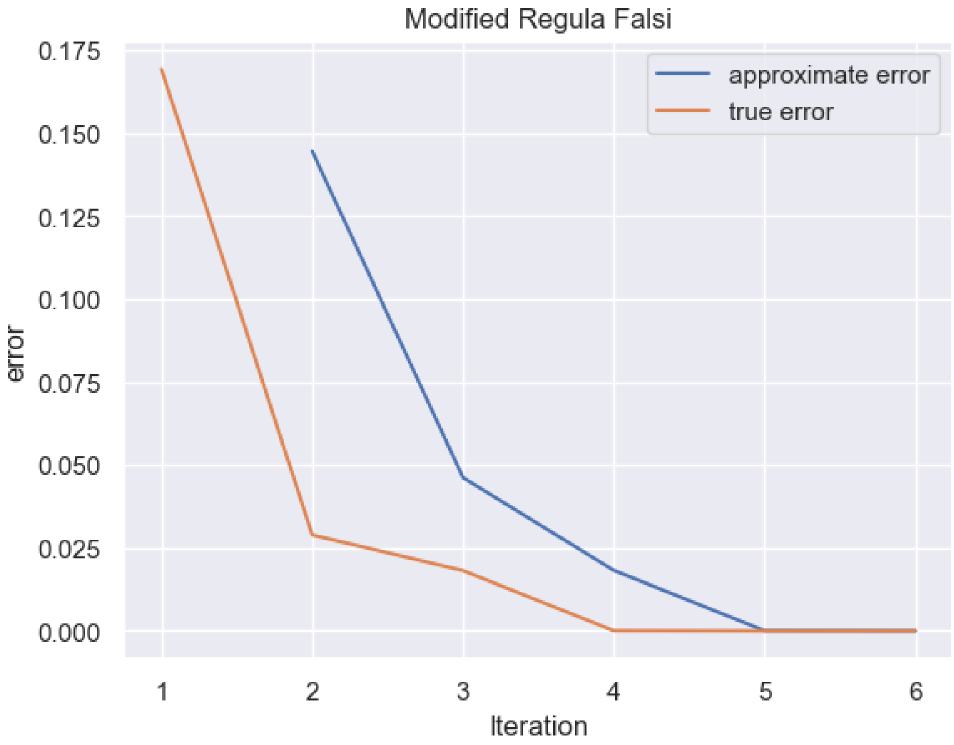


Figure 13. Plots of the true and approximate error with modified regula falsi method (rescaling = 0.5 and counter limit = 2) applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures.

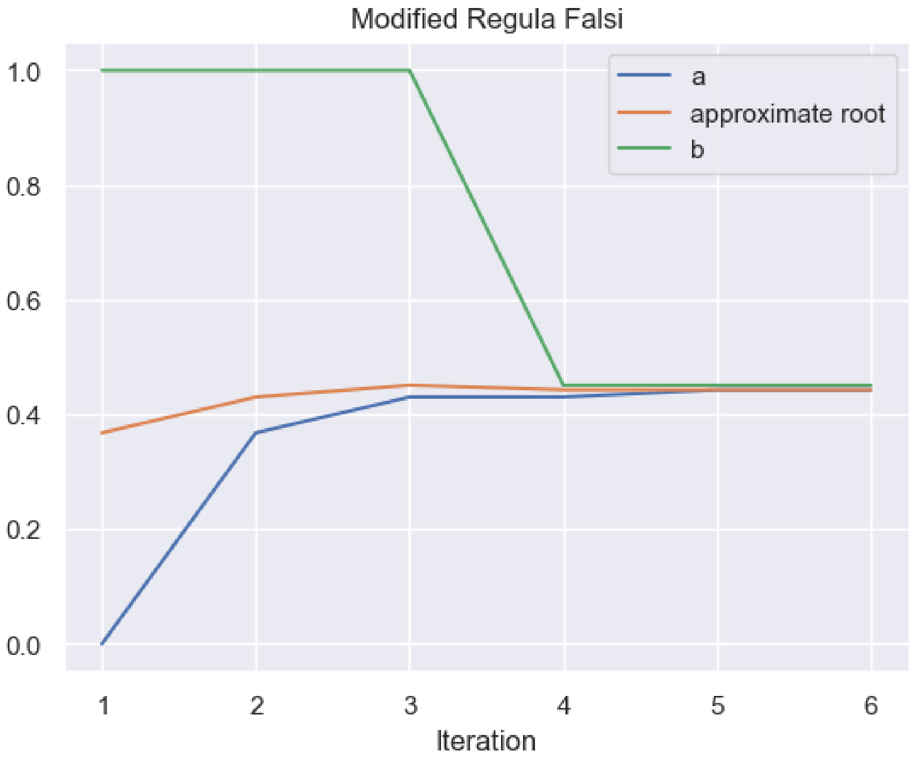


Figure 14. Plots of the approximate root and the interval [a, b] with modified regula falsi method (rescaling = 0.5 and counter limit = 2) *applied to the function $f(x) = e^x + x - 2$ starting with an initial interval of [0, 1] and aiming for a root with 6 significant figures.*

To use the fixed-point iteration method, we need to determine the function g(x). In this work, we analyze two obvious forms of g(x), functions (7) and (9). To ensure that the fixed-point iteration will work, there is a condition, $|g'(x)| < 1$ for every x in the neighborhood of the root. If we solve the inequality for function (8), we find that it is true for $x < 0$, which doesn't contain the region close to the root. If we solve the inequality for function (10), we find that it is true for $x \in (-\infty, 1) \cup (3, +\infty)$, which contains the neighborhood of the root. Therefore, we will use function (10) with initial guess $x_r = 0$.

$$e^x + x - 2 = 0 \Leftrightarrow x = 2 - e^x$$

$$g(x) = 2 - e^x \tag{7}$$

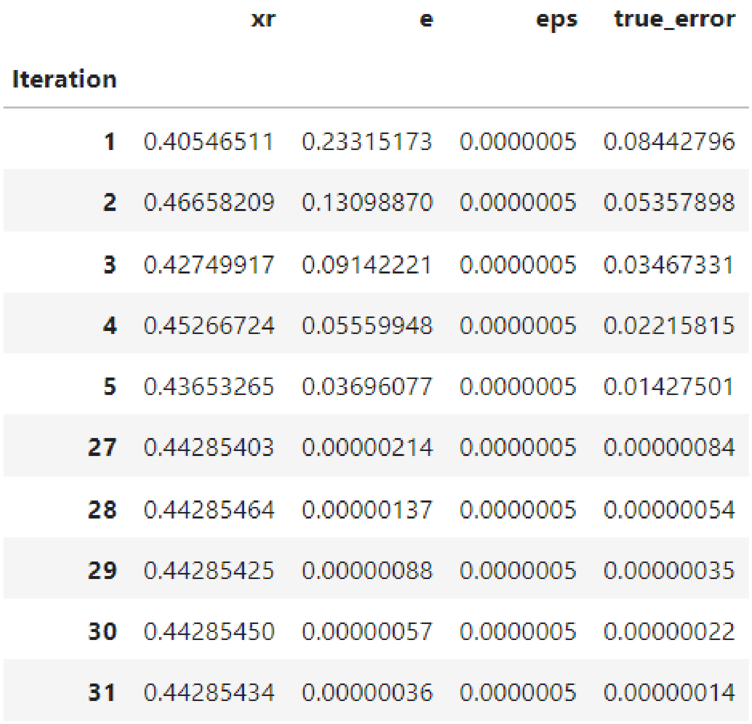
$$g'(x) = -e^x \tag{8}$$

$$e^x + x - 2 = 0 \Leftrightarrow e^x = 2 - x \Leftrightarrow x = \ln(2 - x)$$

$$g(x) = \ln(2 - x) \tag{9}$$

$$g'(x) = \frac{1}{x - 2} \tag{10}$$

Table 4. Results from the fixed-point iteration method for the function $f(x) = e^x + x - 2$ with an initial guess $x_r = 0.5$ and a 6 significant figures root. This table displays the first and last 5 rows of the data.



Iteration	xr	e	eps	true_error
1	0.40546511	0.23315173	0.0000005	0.08442796
2	0.46658209	0.13098870	0.0000005	0.05357898
3	0.42749917	0.09142221	0.0000005	0.03467331
4	0.45266724	0.05559948	0.0000005	0.02215815
5	0.43653265	0.03696077	0.0000005	0.01427501
27	0.44285403	0.00000214	0.0000005	0.00000084
28	0.44285464	0.00000137	0.0000005	0.00000054
29	0.44285425	0.00000088	0.0000005	0.00000035
30	0.44285450	0.00000057	0.0000005	0.00000022
31	0.44285434	0.00000036	0.0000005	0.00000014

Table 4 shows that the fixed-point iteration method is the slowest of all, requiring 31 iterations to converge. This is because, as we mentioned earlier, the first derivative of g(x) is the rate of the error. However, because |g'(0.5)|=0.6667, which is too big, the convergence is too slow. Another g(x) with a value of g'(xr) closer to zero would have a significantly faster convergence. In Figure 16, we can see the approximation root to do oscillation, as in case b at Figure 8 around the true root converging slowly.

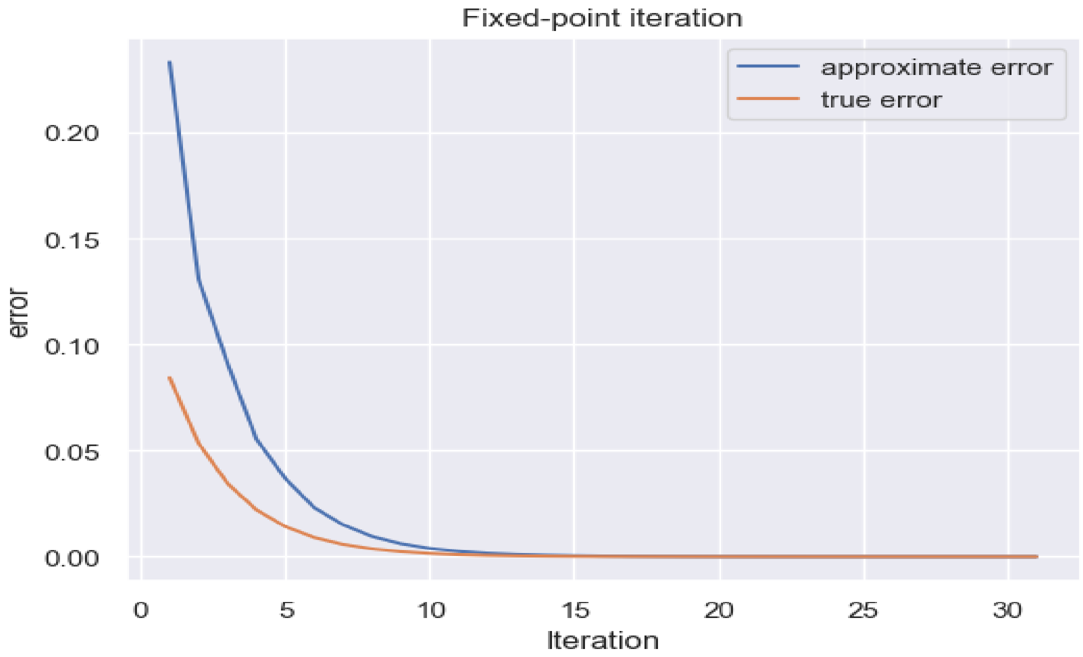


Figure 15. Plots of the true and approximate error with fixed-point iteration method applied to the function $f(x) = e^x + x - 2$ starting with an initial guess $x_r = 0.5$ and aiming for a root with 6 significant figures.

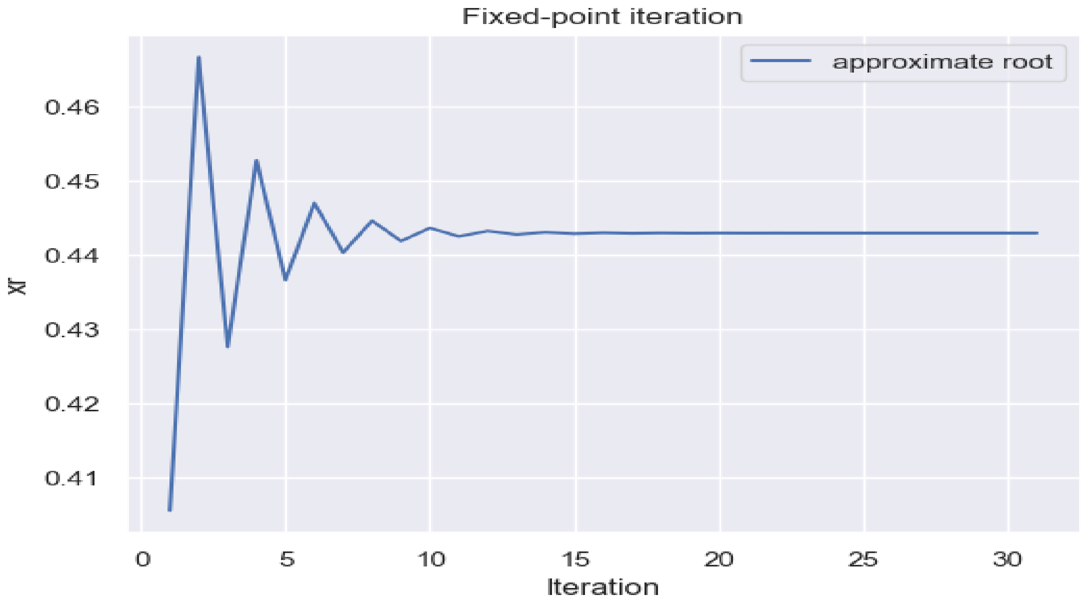


Figure 16. Plots of the approximate root with fixed-point iteration method applied to the function $f(x) = e^x + x - 2$ starting with an initial guess $x_r = 0.5$ and aiming for a root with 6 significant figures.

Conclusion

The rank of the methods used in this work, based on their results, is shown in Table 5. It’s incredible the fact that the faster method (Regula Falsi), which converges in just 10 iterations, can improves by such a simple idea and gives us a result that is 40% better. This can inspire us to believe that we can improve or create better methods that can solve such problems or even more complex ones with desired accuracy and less computational time. The fixed-point iteration method can be a very good choice if g’(x) is close enough to zero, but finding a g(x) with that condition can be a very painful process, which I don’t support.

Table 5. Rank the efficiency of 4 numerical methods in solving the equation $e^x + x - 2 = 0$ with 6 significant figures.

Rank	**Method**	**Iterations**
1	Modified Regula Falsi	6
2	Regula Falsi	10
3	Bisection	23
4	Fixed-Point Iteration	31

Bibliography

1. **Chapra, Steven C. and Raymond P. Canale.** *Numerical Methods for Engineers.* SEVENTH EDITION. United States of America : McGraw-Hill, 2015.

2. **Lehtonen, Jussi.** The Lambert W function in ecological and evolutionarymodels. *Methods in Ecology and Evolution.* 2016, Vol. 7, 9.