Wrocław University of Science and Technology

Scientific Calculations - Task List 3

idk yet

Marcel Jerzyk

November 15, 2019

1 Bisection Method

1.1 Problem

Create a function that solves $f(x) = 0$ using a Bisection Method.

1.2 Implementation

Used third presentation from our lectures as a base for the code implementation (fifth slide).

1.3 Algorithm

Method Bisection works because of a fact that if a function is continuous function in interval $[a, b]$ and if $f(a) * f(b) < 0$ then the function must have a zero place somewhere in said interval. This is how it works:

Check if $f(a) * f(b) < 0$ to proceed:

1. Calculate middle-point $c = \frac{a+b}{2}$
2. Test if $f(a) * f(c) < 0$
   (a) Yes: f have a zero in $[a, c]$
   Replace b for c
   (b) No: f have a zero in $[c, b]$
   Replace a for c

After that we end up with new interval that is two times shorter and have f's zero. We repeat the algorithm until we meet $f(a) * f(c) = 0$ which means that $f(c) = 0$. That c is what we were looking for.

1.4 Code Differences

In *Float32* and *Float64* we must define end condition for our calculations because otherwise ***approximation errors*** can lead to never ending algorithm cycle due to not reaching original end condition. That's why the function has δ and σ to avoid such pathological cases $e < \epsilon$ tests if it's possible to try another iteration of the algorithm and $|e| < \delta \vee |v| < \epsilon$ to approximate if we have an answer with an error that is small enough to satisfy or if f value in r is close enough to zero.

There were few more changes, namely to avoid other numerical problems. ***Middle-point*** is calculated with three steps in code $e = b - a$, $e = \frac{e}{2}$, $c = a + e$ which effectively can we write as $c = a + \frac{(b-a)}{2}$. In numerical calculations, it's best to calculate the new value by adding to the previous one a minor correction. It is so because sometimes algorithm may end up with middle point outside of $[a, b]$.

Another one is usage of ***sign*** to calculate change in function sign to avoid potential underflow or overflow with multiplication in $f(a) * f(c) < 0$.

2 Newton Method

2.1 Problem

Create a function that solves $f(x) = 0$ using a Newton Method.

2.2 Implementation

Used third presentation from our lectures as a base for the code implementation (eighth slide).

2.3 Algorithm

Bisection Method works because of a fact that if a function is continuous function in interval $[a, b]$ and if $f(a) * f(b) < 0$ then the function must have a zero place somewhere in said interval. **The difference** compared to *Bisection Method* is that it must have exactly one zero in the interval where *Bisection Method* could have more (even though the algorithm finds only one). This is how it works:

1. Choose an x_1 (which could be anything, usually a, b, 0 or 1
2. Derive a tangent in $f(x_1)$
3. The abscissa of the intersection of the tangent with the x-axis is a first approximation to the solution (denote is as x_2)
4. Use recursive function $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ to reach satisfactory value

If the first approximation (x_1) is not satisfactory then we select another starting point x_2 and repeat all actions. This process can be repeated until a sufficiently good first approximation of the element is obtained.

2.4 Quit Conditions

We use **maxit** for maximum desired amount of iterations of this algorithm. Quit conditions similarly as in previous algorithm: return when the distance between successive approximations is smaller than δ or the value is smaller than ϵ.

$$|x_{k+1} - x_k| < \delta \vee |f(x_k)| < \epsilon$$

2.5 Conclusions

When the calculated root approximation is close enough to the actual value, we need relatively small amount of iterations to obtain the maximum possible accuracy of the result. This is because, when the assumptions are met, the error decreases squarely with increasing iteration (because of continuously taking derivative). That's why it should be faster than the *Bisection Method.* The problem is that in many cases, when the approximated starting point is far away from the f's root, the convergence may not happen.

3 Secant Method

3.1 Problem

Create a function that solves $f(x) = 0$ using a Secant Method.

3.2 Implementation

Used third presentation from our lectures as a base for the code implementation (twelve slide).

3.3 Algorithm

It is similar to Newton Method but instead of using $f(x)$ derivative we use *differential quotient*:

$$|f'(x_n)| \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

So the *Secant Method equation* is:

$$x_{n+1} = x_n - f(x_n) * \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

3.4 Quit Conditions

These are exactly the same as in the Newton Method.

3.5 Conclusion

To hold $|f(a)| \leq |f(b)|$ condition we swap interval end points a and b. It ensures that the values will not increase in further iterations.

4 Finding the Root of an Equation

4.1 Problem

$$f(x) = sin(x) - (\frac{1}{2} * x)^2$$

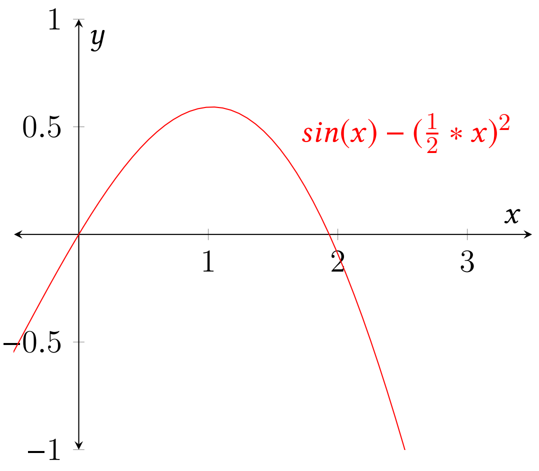
The task is to find root of that equation using previously implemented methods and with given data:

1. Bisection: $[1.5, 2]$, $\delta \wedge \epsilon = \frac{1}{2} * 10^{-5}$
2. Newton: $x_0 = 1.5$, $\delta \wedge \epsilon = \frac{1}{2} * 10^{-5}$
3. Secant: $x_0 = 1$, $x_1 = 2$, $\delta \wedge \epsilon = \frac{1}{2} * 10^{-5}$

4.2 Results

Method	Root	Value	It	Er
Bisection	1.9337539672851562	-2.7027680138402843e-7	16	0
Newton	1.933753779789742	-2.2423316314856834e-8	4	0
Secant	1.933753644474301	1.564525129449379e-7	4	0

Table 1: Comparison of different methods



4.3 Conclusion

As we can see the bisection method needed **16** iterations to find the root of f so it proves that it is indeed slower than *Newton Method* as it needed only **4** iterations (same with Secant Method).

5 Using Bisection Method to find intersection of functions

5.1 Problem

Find intersection points of two functions:

$$f(x) = 3x$$
$$g(x) = e^x$$

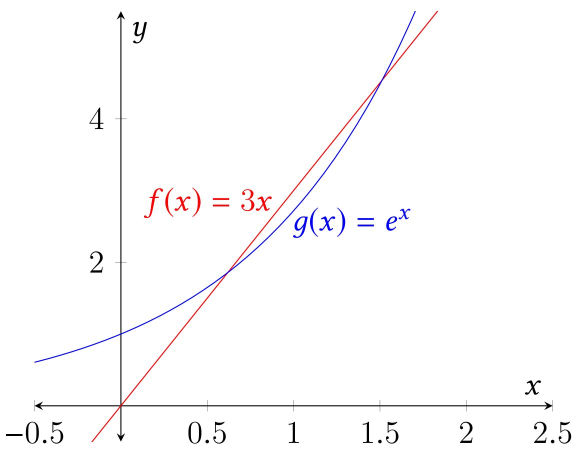
using *Bisection Method* with $\delta \wedge \epsilon = 10^{-4}$.

5.2 Finding Answer

Obviously to finish the task we just need to transform functions into one

$$y = 3x \wedge y = e^x$$
$$3x = e^x$$
$$3x - e^x = 0$$

Then I used drawn both functions to find out in which intervals I should search intersection point (or points - if many).



As seen on the graph - there are two intersection points and they are somewhere between $[0.5, 1]$ and tiny bit above $x = 1.5$.

5.3 Results

Range	Root	It	Er
[0.0, 1.0]	0.619140625	9	0
[0.5, 1.0]	0.619140625	8	0
[1.0, 2.0]	1.5120849609375	13	0
[1.4, 1.6]	1.512109375	9	0

Table 2: Four range tests with $3x - e^x$

5.4 Conclusions

To find those intersection points I had to know the functions variations to get a good grasp where to put the range values for the Bisection Method. Using a wild guess range like $[0, 5]$ produced an Error $(Err = 1)$ because of the *sign(f(a)) == sign(f(b))* condition.

Using something like graph to have a better knowledge of the functions led to conclusions thanks to which the task could be completed by using tighter intervals. Otherwise without a method to find this out the task could be quite difficult to complete. Both tested ranges for both intersection points gave about satisfactory results (~ 0.619 and ~ 1.512) with a difference in amount of the iterations.

Additionally, created "reversed" function to see if something different could happen.

$$e^x - 3x = 0$$

Range	Root	It	Er
[0.0, 1.0]	0.619140625	9	0
[0.5, 1.0]	0.619140625	8	0
[1.0, 2.0]	1.5120849609375	13	0
[1.4, 1.6]	1.512109375	9	0

Table 3: Another test but with $e^x - 3x$

But with no surprise - the results were identical to the $3x - e^x$.

6 Initial Intervals and Starting Approximation

6.1 Problem

The last task on the list is to find zero places of functions:

$$f(x) = e^{1-x} - 1$$
$$g(x) = xe^{-x}$$

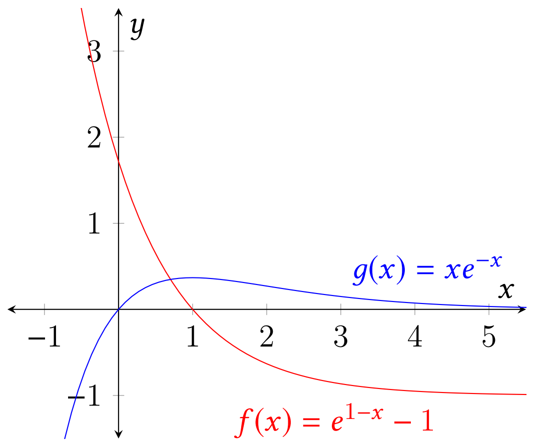
with all of the previously implemented Methods (*Bisection*, *Newton* and *Secant*) and precision equal to $\delta \wedge \epsilon = 10^{-5}$. We shall determine the initial range by ourselves and then test *Newton Method* with $x_0 = [0, \infty)$, for $g(x)$ $x_0 > 1$ and test if we can choose $x_0 = 1$ for $g(x)$.

6.2 Solution

Like in the task before we need to find out the initial interval for *Bisection Method* and for *Newton Method* and *Secant Method* appropriate initial approximation. On the first glance we can already tell that the zero places are 1 for $f(x)$ and 0 for $g(x)$ because

$$f(1) = e^{1-1} - 1 = e^0 - 1 = 1 - 1 = 0$$
$$g(0) = 0 * e^{-0} = 0 * 1 = 0$$

Lets graph these two functions anyway to see visually with what we are dealing with.



6.3 Results

Results are presented on the next page.

Range	**Root**	**Value**	**It**	**Er**
[0.0, 2.0]	1.0	0.0	1	0
[0.0, 1.0]	0.9999923706054688	7.629423635080457e-6	17	0
[1.0, 9.0]	1.0000076293945312	-7.6293654275305656e-6	20	0
[-0.1, 1.7]	1.0000061035156251	-6.103496998699498e-6	15	0
[-3.41, 4.253]	1.0000017776489258	-1.7776473457686848e-6	17	0
[-0.03, 941.5]	0.9999903663992882	9.633647115148136e-6	25	0
[1.0, 9.0^{300}]	1.000007511333226	-7.511305016083547e-6	968	0
[1.98, 2.0]	0.0	0.0	0	1

Table 4: Bisection Method for $f(x)$

X_0	**Root**	**Value**	**It**	**Er**
0.999	0.9999995001666251	4.99833499922886e-7	1	0
1.001	0.9999994998332917	5.001668332837994e-7	1	0
0.0	0.9999984358892101	1.5641120130194253e-6	4	0
1.0	1.0	0.0	0	2
0.5	0.9999999998878352	1.1216494399945987e-10	4	0
-2.0	0.9999999999251376	7.48623385504743e-11	7	0
123.0	NaN	NaN	100	1
9.0^{300}	NaN	NaN	100	1

Table 5: Newton Method for $f(x)$ ($maxit = 10^2$)

X_0, X_1	**Root**	**Value**	**It**	**Er**
0.0, 2.0	1.0000017597132702	-1.7597117218937086e-6	6	0
0.0, 1.0	1.0	0.0	1	0
1.0, 9.0	1.0	0.0	1	0
-0.1, 1.7	1.000000018143681	-1.814368089103624e-8	6	0
-3.41, 4.253	4.1634135543727036	-0.9577188345946179	3	0
-0.03, 941.5	NaN	NaN	100	1
1.0, 9.0^{300}	1.0	0.0	1	0
1.98, 2.0	1.0000001195152606	-1.1951525347164704e-7	7	0

Table 6: Secant Method for $f(x)$ ($maxit = 10^2$)

Range	**Root**	**Value**	**It**	**Er**
[0.0, 2.0]	7.62939453125e-6	7.62933632381113e-6	18	0
[0.0, 1.0]	7.62939453125e-6	7.62933632381113e-6	17	0
[1.0, 9.0]	0.0	0.0	0	1
[-0.1, 1.7]	3.051757812496916e-6	3.051748499285381e-6	16	0
[-3.41, 4.253]	3.4294128417685943e-6	3.4294010809163218e-6	18	0
[-0.03, 941.5]	470.735	1.7185554577015025e-202	1	0
[1.0, 9.0^{300}]	9.36963851942397e285	0.0	1	0
[1.98, 2.0]	0.0	0.0	0	1

Table 7: Bisection Method for $g(x)$

X_0	**Root**	**Value**	**It**	**Er**
0.999	44.755291502626946	1.6363200601993009e-18	2	0
1.001	45.069067725790866	1.2040129462915176e-18	2	0
0.0	0.0	0.0	0	2
1.0	44.911837503175164	1.404098657788036e-18	2	0
0.5	45380.840617460846	0.0	3	0
-2.0	15.14032501373688	4.025089953143983e-6	10	0
123.0	123.0	4.695521542065545e-52	0	2
9.0^{300}	1.873927703884794e286	0.0	0	2

Table 8: Newton Method for $g(x)$ ($maxit = 10^2$)

X_0, X_1	**Root**	**Value**	**It**	**Er**
0.0, 2.0	0.0	0.0	1	0
0.0, 1.0	0.0	0.0	1	0
1.0, 9.0	14.6011702062938	6.655489669233851e-6	8	0
-0.1, 1.7	7.262297938216049e-7	7.26229266412083e-7	5	0
-3.41, 4.253	14.846722384909766	5.293963976127139e-6	14	0
-0.03, 941.5	941.5	0.0	1	0
1.0, 9.0^{300}	1.873927703884794e286	0.0	1	0
1.98, 2.0	14.293972124133456	8.858490276816477e-6	14	0

Table 9: Secant Method for $g(x)$ ($maxit = 10^2$)

.

6.4 Conclusions

Bisection Method will always calculate the correct zero value when it's in the middle of the given range. The result is calculated at a iteration rate depending on the size of the range and how far the zero is from middle cuts (compare for example $[0.0, 1.0]$ and $[0.1, 1.7]$ where the correct answer is on one of the "edges" of the range).

Newton Method sometimes will refuse to give the correct answer if the initial guess is too bad. In $g(x)$ it will try to find it by following the functions convergence to zero which won't be correct without good initial approximation. This is a big concern although on the other hand - it works very fast.

Secant Method failed to work with ($maxit = 10^2$) with given iterations when chosen $X_0 \wedge X_1$ were too far away from each other and chosen intentionally to be tough for the algorithm. It also gave the wrong answers similarly like in Newton Method with "bad" initial x_0 and x_1. To sum it up - if the initial aproximation was incorrectly selected, then it returns the root that is not a zero place. This is due to the function value which is equal to less than the specified accuracy.

Choosing correct range or the initial approximated value is quite crucial in all Methods to achieve low iteration numbers and even - the correct results. It's especially important in the two latter methods where it could produce wrong values with bad initial approximations, because even though its much slower - *Bisection Method* will get the job done in *some* iterations unless the answer is out of range.