Chmielewska Urszula

300167

Advisor: dr inż. Andrzej Miękina

Numerical Methods

Solving nonlinear equations

Assignment B nr. #8

I declare that this piece of work, which is the basis for recognition of achieving learning outcomes in the Numerical Methods course, was completes on my own.

Urszula Chmielewska

300167

Theoretical introduction

There are several methods of finding root for a given function. Procedures differ in preformed calculation however the approach in used algorithm is similar. The aim of this project was to compare the following methods:

- Bisection method
- Regula-falsi method
- Secant method
- Newton's method

To fully understand computations presented below it is necessary to get familiarized with each method algorithm.

Bisection method is defined by the formula:

$$x_i = \frac{1}{2}(a_i + b_i)$$

$$[a_{i+1}, b_{i+1}] = \begin{cases} [a_i, x_i] \; if \; f(a_i) * f(x_i) < 0 \\ [x_i, b_i] \; if \; f(a_i) * f(x_i) > 0 \end{cases}$$

for i= 0,1,…

Also it is necessary to state that the iterations can begin if the given function *f(x)* is continuous in interval $[a_0, b_0]$ and if the $f(a_0) * f(b_0) < 0$ condition is satisfied. Appropriately it means that function values for a_0 and for b_0, have to have results with opposite signs.

Regula-falsi method is defined by the formula:

$$x_{i+1} = x_i - \frac{x_i - x_0}{f(x_i) - f(x_0)} f(x_i) \qquad for\ i=1,2,\ldots$$

It is necessary to set proper x_0 and x_1 values. Constant point x_0 cannot have value closer to 0,5 to the exact solution. Point x_1 value changes in each iteration.

Local convergence can be calculated using formula: $C = \frac{f'(x)}{f(x_0)}(x_{exact} - x_0) + 1$

and q=1

Secant method is defined by the formula:

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \qquad for\ i=1,2,\ldots$$

Local convergence can be calculated using formula: $C = \left[\frac{1}{2}\frac{f''(x_{exact})}{f'(x_{exact})}\right]^{\frac{1}{2}(\sqrt{5}-1)}$

and q = $\frac{1}{2}(\sqrt{5}+1) = 1{,}61$

Newton's method is defined by the formula:

$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ *for i=1,2,…*

Local convergence can be calculated using formula: $C = \frac{1}{2}\frac{f''(x_{exact})}{f'(x_{exact})}$

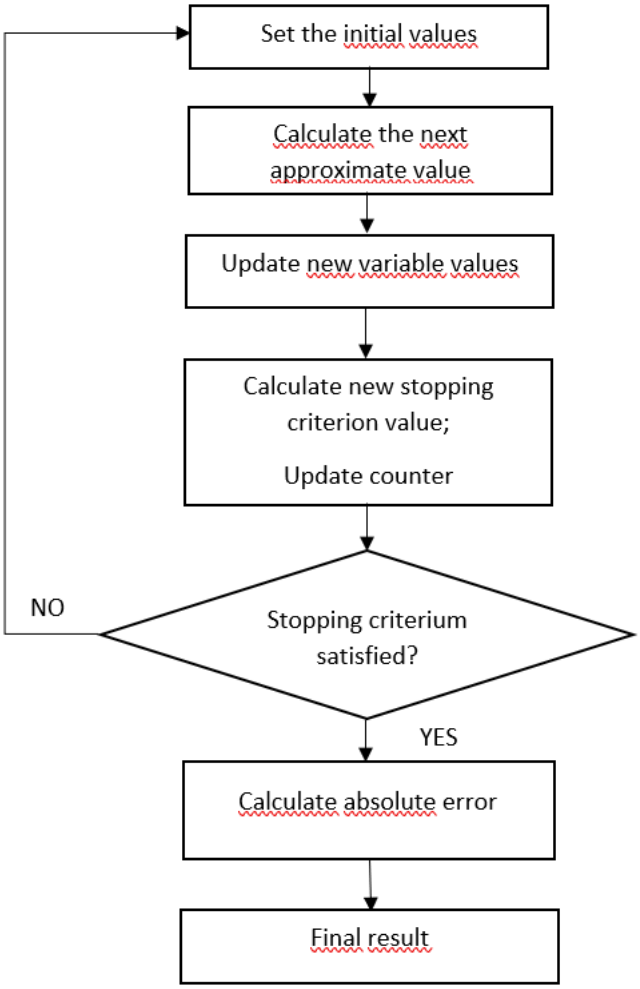
and q=2

To calculate absolute error of computation I used the formula (valid for each method):

$\Delta x = |x_{obtained} - x_{exact}|$

Algorithm

To solve tasks 2,3,4 and 5 in each method I used similar algorithm to perform calculations. The main idea is presented on the below flowchart:



For each method the initial conditions and way of calculating the next approximated solution value was different. In tasks, the presented algorithm was performed for stopping criterions smaller than several Δ (Δ = 10^{-3}, 10^{-4},…,10^{-6}).

MatLab solution – general approach description

In MatLab solution I practice similar approach in method. I decided to make *for* loop with 15 iterations, because there are 15 delta values to be evaluated. In such loop I inserted *while* loop which allows to find the best approximated value according to the formulas, distinctive for each method. Apart from stopping criterion given in the task I also set another stopping condition. It was a maximum iterations number equal to 100, to avoid infinite or too long computations.

After such loop I could evaluate absolute error, for the obtained solution for each delta. During tests I noticed that, especially in secant and Newton's methods, the absolute error or the stopping criterion were equal to 0 for many delta's values. To avoid such situations I replaced it with build in MatLab value *eps*. Due to this replacement I can observe all points on the graphs.

I decided to store solutions, obtained in tasks, in arrays (ex.: *allcounter(i)=counter*). Such approach allowed me to easily plot graphs representing dependences. To plot the graph comparing the iteration number (I) versus Δ *(Fig.1),* I used *semilogx* function. Due to that, x-axis is presented in a logarithmic scale. I update this graph after performed calculations on each method, using *hold on*.

Individual figures for each method, presenting comparison between the absolute errors versus Δ and the final criterion values versus Δ, I decided to plot using *loglog* function. Logarithmic scale gave me a better representation of the obtained result values.

TASK 1.

Solve $f(x) = 0$, where $f(x) = \ln(x) + x + \sin\left(\frac{x^2}{100}\right) - \frac{27}{4}$ for $x \in (0, 10]$

I was supposed to find the solution applying the procedure *fzero*. For this purpose I had to chose a starting point from the given *x* interval.

For starting point $x \leq 2.5$ I obtained this message:

```
Exiting fzero: aborting search for an interval containing a sign change
    because complex function value encountered during search.
(Function value at -0.7 is -7.8018+3.1416i.)
Check function or try again with a different starting value.

x0 =

   NaN
```

For any other *x*, chosen from the interval $x \in (2.5, 10]$, I was able to find the solution, which is:

$x_0 = 4.9178$

In the following tasks, the above solution x_0 is considered as an exact solution.

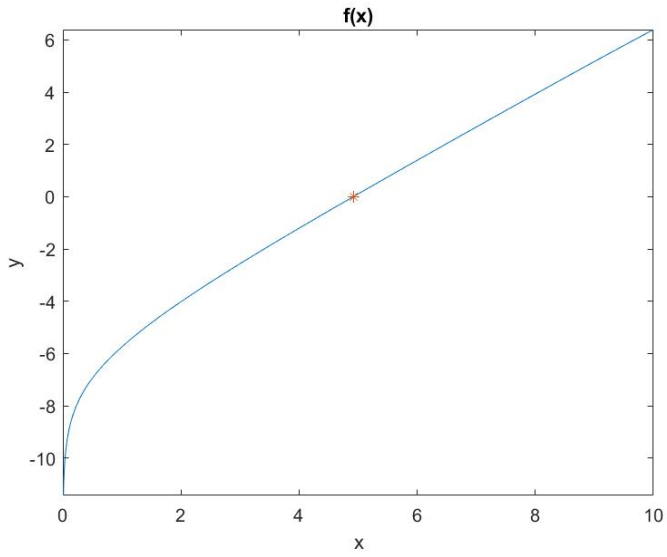


*Fig.11 Function f(x) with marked exact solution**

TASK 2.

Solving the equation using the bisection method, required starting points to initialize the procedure. The same variables were also used as a criterion for terminating the iterations. My starting points where: $a_0 = 10^{-6}$ and $b_0 = 10$.

Iterations stop if the $\frac{1}{2}(b_i - a_i) < \Delta$ condition is completed. I also set the counter condition *(counter<100)* since for very small Δ, it took a great amount of time to find the solution, for which the terminating condition was satisfied. It also reduced the program uptime.

From the undermentioned *Fig. 2* I can conclude that the final criterion value decreased gradually for each delta. On the contrary, absolute error of the solution. Noticeable peak was generated, because for $\Delta = 10^{-9}$, accidentally calculated solution had a value closer to the exact solution.

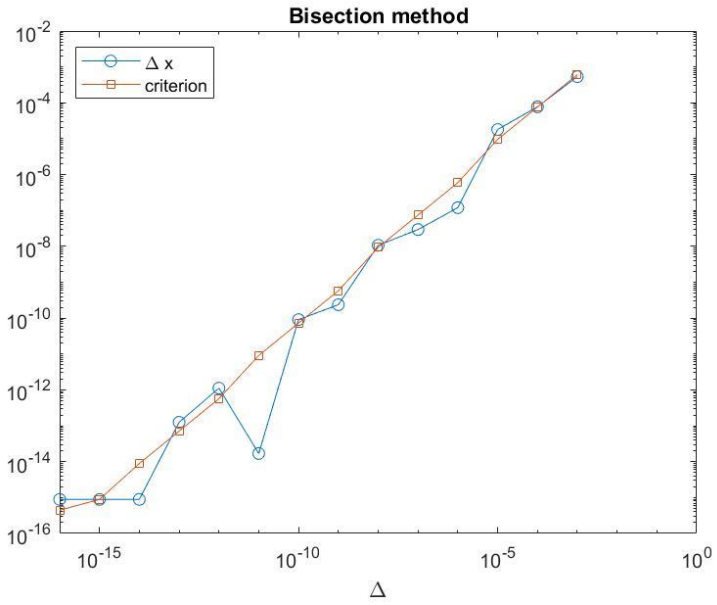


Fig. 2 The actual absolute error versus Δ compared with the criterion value versus Δ

TASK 3.

Regula-falsi method required constant and starting points. To choose the constant point is was necessary to know an exact solution, because its value had to have a value not closer than 0,5 to the exact solution. Chosen parameters for my calculations:

$x_0 = 4$ *(constant point)*

$x_1 = 10$ *(starting point)*

The stopping condition was the $|x_i - x_{i-1}| < \Delta$ unequality. In the MatLab code solution for this task, I had to store an *x* value calculated in a previous iteration. It was necessary for stopping condition calculation.

Presented below *Fig.3*, shows the staircase changes structure of Δx and criterion versus Δ. Mostly for the two following solutions, their absolute error was the same. Also similar behavior of the criterion values is visible.

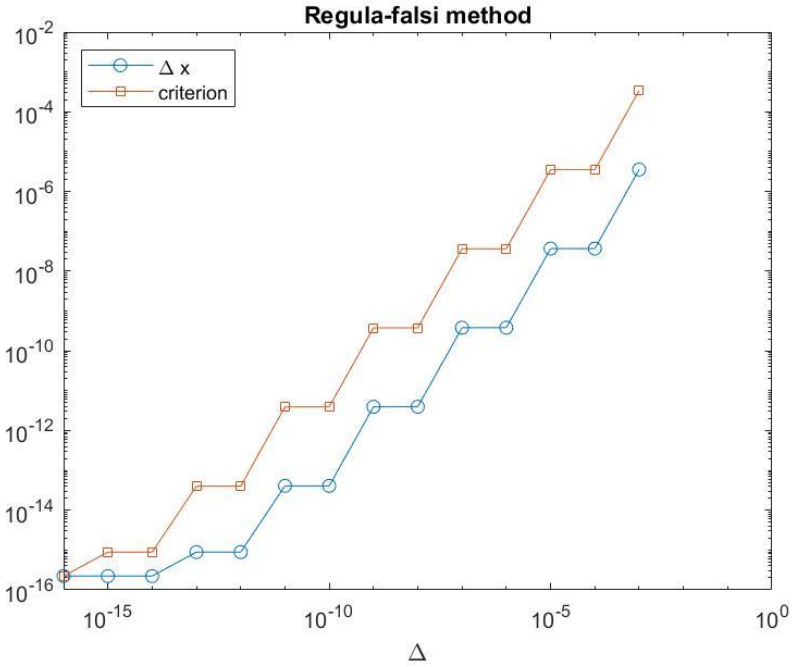


Fig. 3 The actual absolute error versus Δ compared with the criterion value versus Δ

TASK 4.

In the secant method I used the same starting points as in the previous task:

$x_0 = 4$ *(lower starting point)*

$x_1 = 10$ *(upper starting point)*

However in this method both points change their values in each iteration. All used criterions are the same, as presented in the task 3.

From the following figure we can expeditiously conclude that this method, allowed us to find a solution equal to the exact solution already for $\Delta = 10^{-4}$.

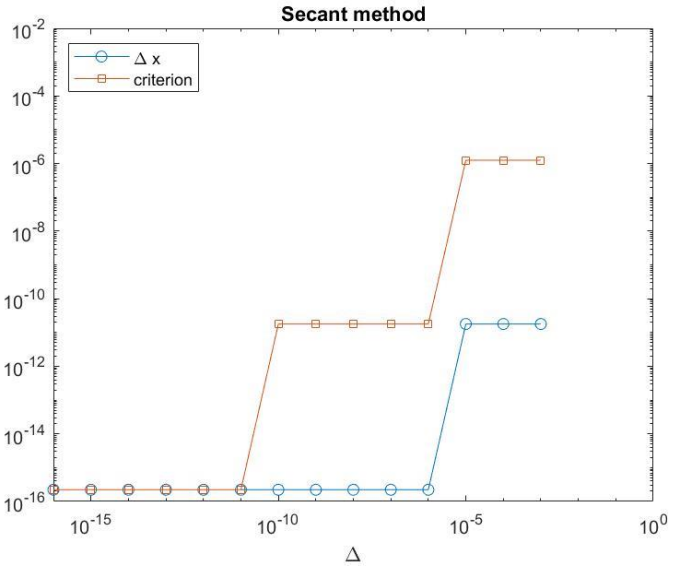


Fig. 4 The actual absolute error versus Δ compared with the criterion value versus Δ

TASK 5.

To solve $f(x) = 0$ using Newton's method, I had to set a starting point $x_0 = 10$. It was also necessary to compute $f'(x)$ analytically.

$$f'(x) = (\ln(x) + x + \sin\left(\frac{x^2}{100}\right) - \frac{27}{4})' = \frac{1}{x} + 1 + \frac{x}{50}\cos\left(\frac{x^2}{100}\right)$$

All terminating criterions were the same as in the task 3. In my MatLab solution I decided to perform the first calculation with Newton's formula before the while loop, because I needed a x_1 value, to calculate the first criterion value to start iteration.

On the *Fig. 5* presented below it is visible that the Newton's method allowed us to find a solution equal to the exact one, already for $\Delta = 10^{-6}$.

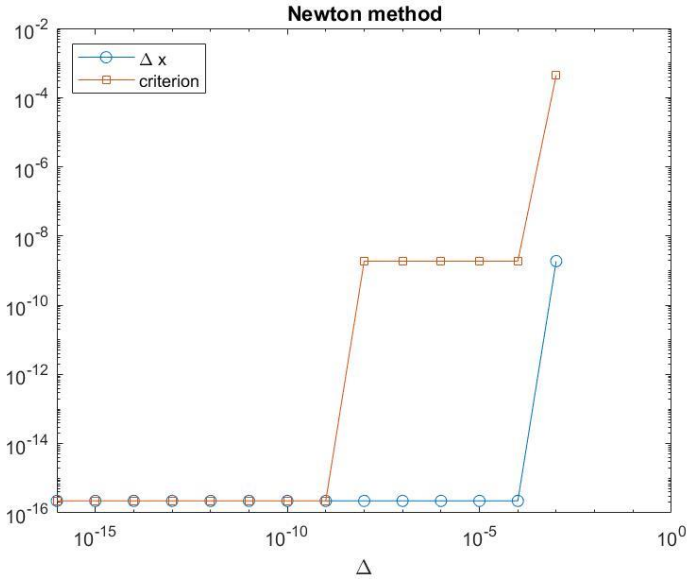


Fig. 5 The actual absolute error versus Δ compared with the criterion value versus Δ

METHODS SUMMARY

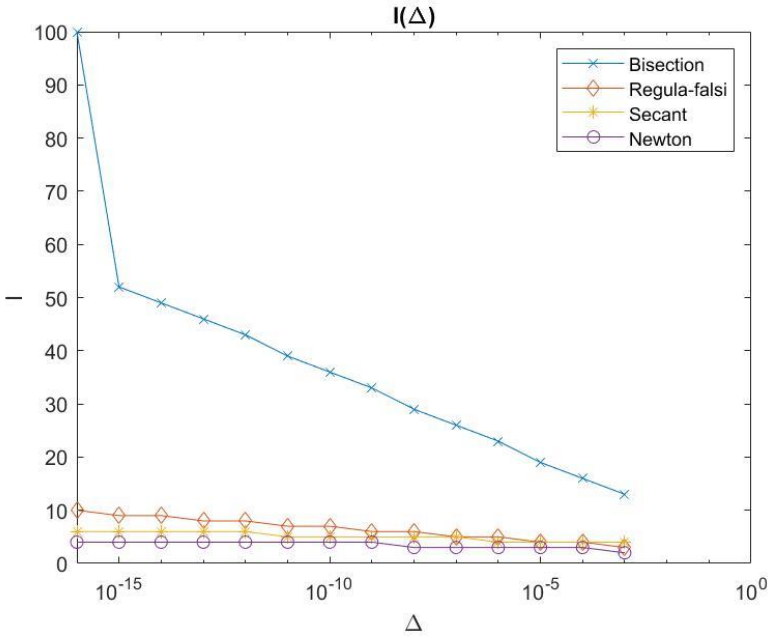


Fig. 1 Number of iterations for each methods versus Δ

Recall

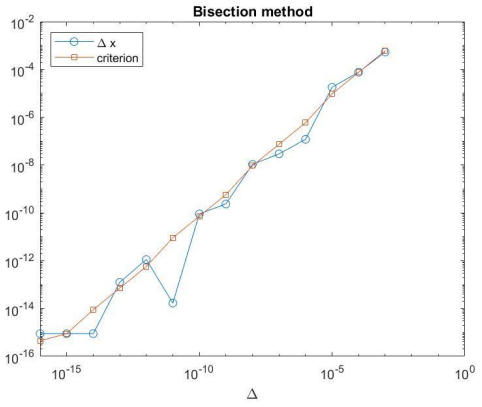


Fig. 2

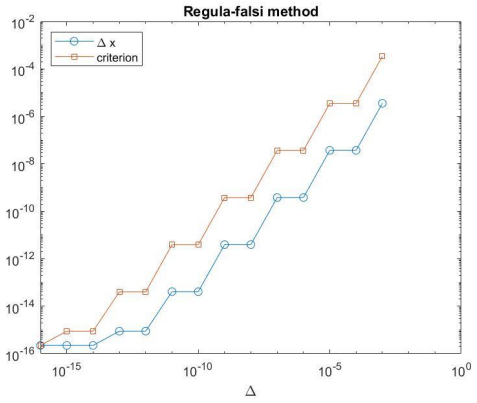


Fig. 3

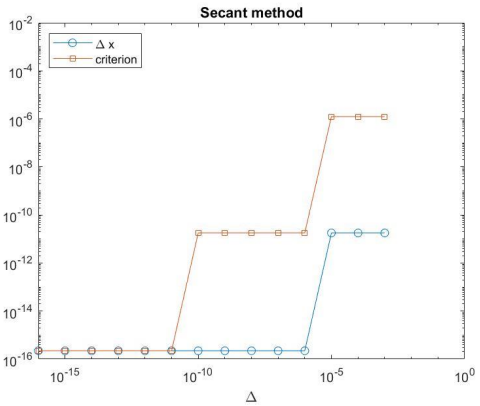


Fig. 4

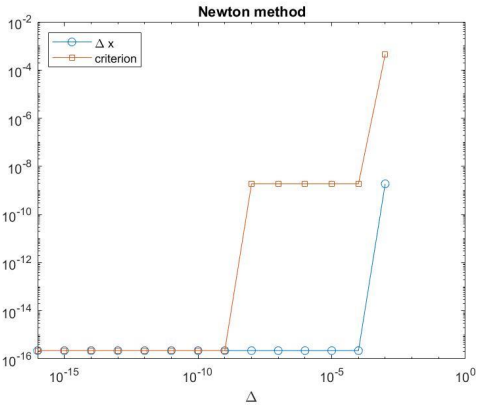


Fig. 5

Fig. 1 summarizes the effectiveness of each tested method. This visualization and figures presented in previous tasks, were necessary to make conclusions about methods optimality.

Due to the x-axis and y-axis limits on the graphs, the same for each method, I could conclude that for the final solution obtained absolute error was the same in regula-falsi, secant and in the Newton's method. Solely in bisection method this error was insignificantly bigger. However it is a result of set condition for maximum iteration number.

On the *Fig. 1* it is also apparent that for the Bisection method graph, iteration number significantly increases with the decreasing Δ. Contrary to the remaining processes; in regula-falsi iteration number, insignificantly but little by little increases. In secant and Newton's method this increase is almost sightless.

After all performed observations I can conclude that the Newton's method is the most efficient approach. It requires the least number of iterations to find solution. It is important case in terms of program fastness. Also the absolute error had the smallest value for the second, given in tasks, Δ value.

TASK 6.

In the previous tasks I presented solutions obtained for optimal starting points. This task is concerned to examine the impact of the initial points on the figures presenting the actual absolute error versus Δ compared with the criterion value versus Δ.

Bisection method

When examining the impact of the initial points it is necessary that this algorithm works properly only is the condition $f(a_0) * f(b_0) < 0$ is satisfied.

I decided to try with points: $a_0 = 10^{-16}$ and $b_0 = 10000$. The graph did not differ a lot. The smallest absolute error value was the same as for the default values in task 2. For this case there were no accidental drops. Values smoothly decrease.

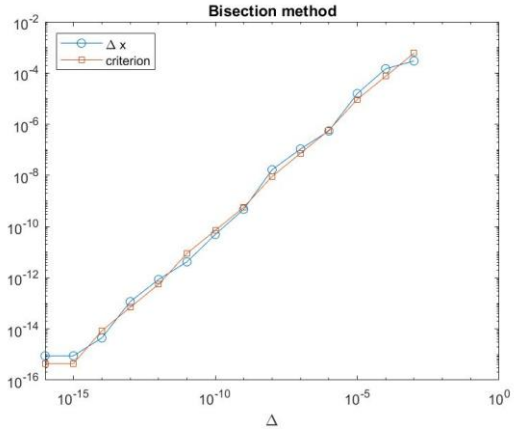


Fig. 6a bisection with $a_0 = 10^{-16}$ and $b_0 = 10000$

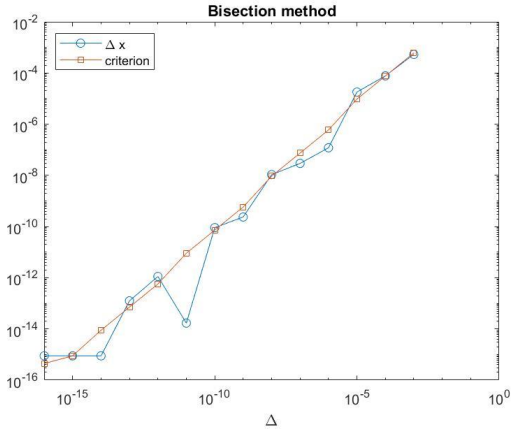


Fig. 2 from task 2

Regula-falsi method

In this method increase of the starting point $x_1$to 10000 does not have a significant impact on the obtained results. However it is different for changes x_0 value to very small, ex.: $x_0 = 10^{-6}$. From the figures presented below I concluded that for such small value, number of iterations increased dramatically and now it is much less efficient than before. I can draw a conclusion that this method is efficient only if the constant point is properly chosen (has a value close enough to the exact solution).

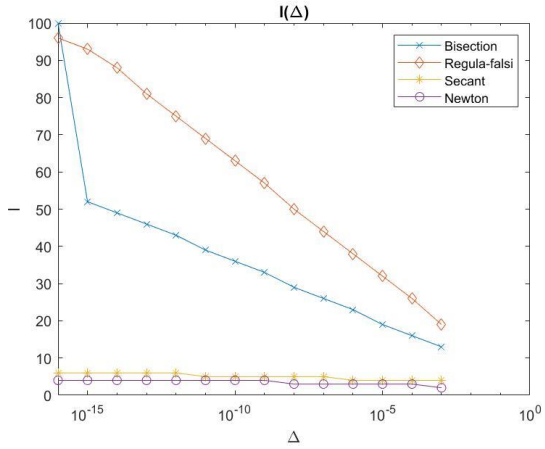


Fig. 6b regula-falsi with $x_0 = 10^{-6}$ and $x_1 = 10$

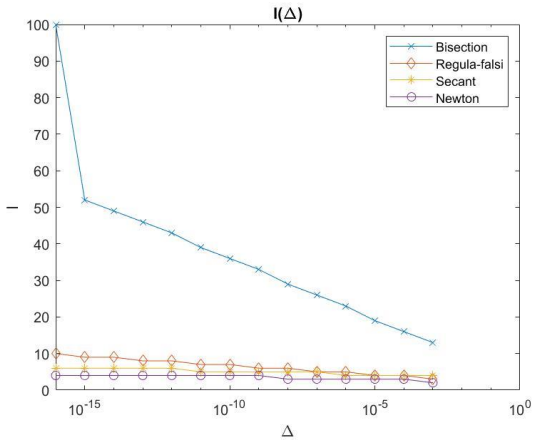


Fig. 3

Orange line now takes much higher values I (iterations) increasing dramatically.

Secant method

I examined the impact of much smaller lower starting point x_0 and much greater upper starting point x_1. For this purpose I took values: $x_0=10^{-26}$ and $x_1=10^9$. With such significant change, I noticed a negative increase of the iterations number (I).

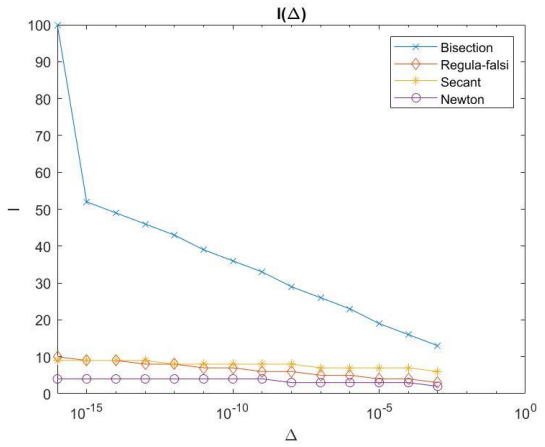


Fig. 6c secant with $x_0 = 10^{-26}$ and $x_1 = 10^9$

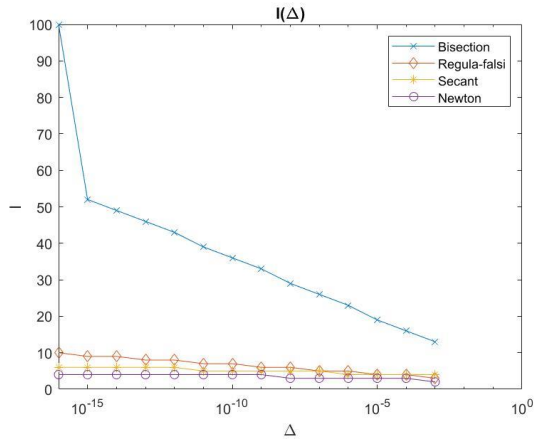


Fig. 1

Yellow line now takes a bit higher values of I for smaller Δ.

There are also visible changes on the second graph. I noticed that for such values the maximum Δx value and maximum criterion value are higher than in the computations made before. However in my opinion this unwanted result does not matter a lot.

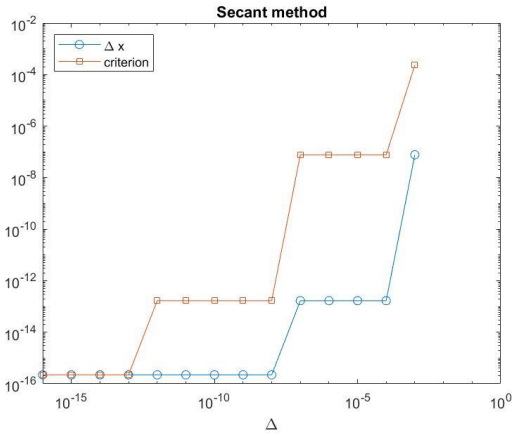


Fig. 6c secant with $x_0 = 10^{-26}$ and $x_1 = 10^9$

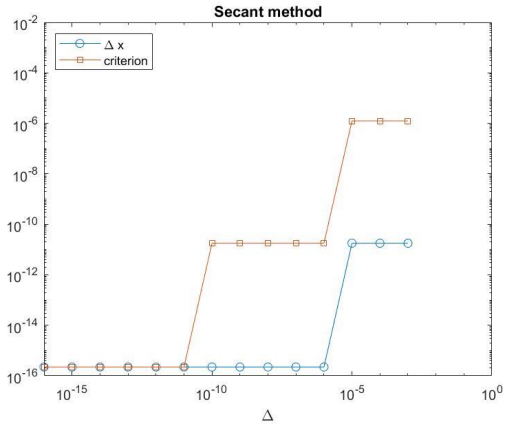


Fig. 4

Newton's method

Changing the starting point x_0=11 causes even faster result receive. However for other values out of the interval given at the beginning $x \in (0, 10]$, this method fails. This method is not efficient with such extreme values.

Below I present some examples of Newton's method failures:

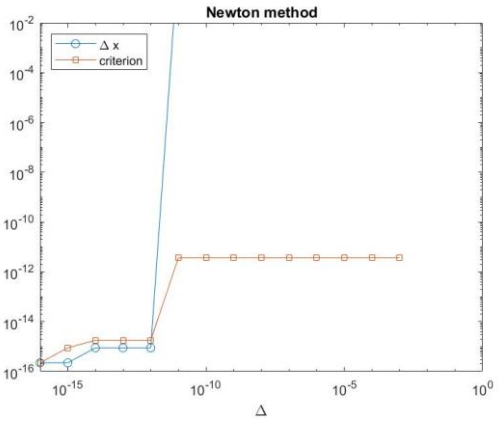


Fig. 6d Newton's with $x_0 = 10^{-13}$

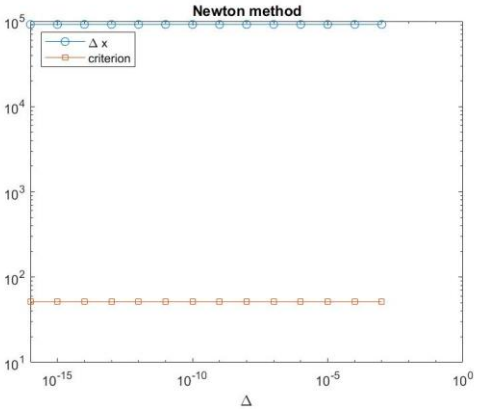


Fig. 6e Newton's with $x_0 = 10^5$

Final conclusions

After performed examination in task 6 I conclude that regula-falsi method is efficient only if constant point is properly chosen. Newton's method failed for much greater or smaller values. From performed computations and observations I conclude that the secant method is the most efficient and preferable.

APPENDIX – SOURCE CODE

```
close all
clear
clc
format long;
%==========================TASK 1==========================
f=@(x)log(x) + x +sin(x.^2/100)-27/4;

%exact solution
xexact=fzero(f,6)

figure(10);
fplot(f,[0,10]);
hold on;
scatter(xexact,0,'*');
xlabel('x');
ylabel('y');
title('f(x)');
hold off;

%==========================TASK 2==========================
%=====================bisection method=====================
for i=1:14

    a=10^(-6);                  %lower number
    b=10;                       %upper number
    criterion2=(b-a)/2;
    delta=10^(-3)/10^(i-1);
    counter=0;

    while (criterion2>delta && counter<100)
        xsolution=(b+a)/2;
        if f(xsolution)<0
            a=xsolution;
        else
            b=xsolution;
        end
        criterion2=(b-a)/2;
        counter=counter+1;
    end
    deltax=abs(xsolution-xexact);

    alldeltax(i)=deltax;
    allcounter(i)=counter;
    alldelta(i)=delta;
    allcriterion2(i)=criterion2;
    allxsolution(i)=xsolution;
end

figure(1);
semilogx(alldelta, allcounter,'-x');
title('I(\Delta)');
ylabel('I');
xlabel('\Delta');
hold on;

figure(2);
loglog(alldelta, alldeltax,'-o');
title('Bisection method');
xlabel('\Delta');
hold on;
loglog(alldelta, allcriterion2,'-s');
legend('\Delta x','criterion', 'Location', 'Northwest');
ylim([10^(-16) 10^(-2)]);
xlim([10^(-16) 10^(0)]);
hold off;

%==========================TASK 3==========================
%=====================Regula-falsi method=====================
for i=1:14

    x0=4;                           %constant point
    x1=10;                          %starting point
```

```
    criterion3=abs(x1-x0);
    delta=10^(-3)/10^(i-1);
    f0=f(x0);                       %constant
    counter=0;

    while (criterion3>delta && counter<100)
        f1=f(x1);                   %changes in each iteration
        xprev=x1;
        x1=x1-((x1-x0)/(f1-f0))*f1; %regula-falsi formula
        criterion3=abs(x1-xprev);
        counter=counter+1;
    end
    deltax=abs(x1-xexact);

    if (criterion3 ==0)
        criterion3=eps;
    end
    if (deltax == 0)
        deltax=eps;
    end

    alldeltax(i)=deltax;
    allcounter(i)=counter;
    allcriterion3(i)=criterion3;
    allxsolution(i)=x1;
end

figure(1);
semilogx(alldelta, allcounter,'-d');

figure(3);
loglog(alldelta, alldeltax,'-o');
title('Regula-falsi method');
xlabel('\Delta');
hold on;
loglog(alldelta, allcriterion3,'-s');
legend('\Delta x','criterion', 'Location', 'Northwest');
ylim([10^(-16) 10^(-2)]);
xlim([10^(-16) 10^(0)]);
hold off;

%============================TASK 4============================
%=========================secant method=======================
for i=1:14

     x0=4;                          %lower starting point
     x1=10;                          %upper starting point

    criterion4=abs(x1-x0);
    delta=10^(-3)/10^(i-1);
    counter=0;
    f0=f(x0);
    f1=f(x1);

    while (criterion4>delta && counter<100)
        x=x1-((x1-x0)*f1/(f1-f0));      %secant formula
        fx=f(x);                        %function value at new approximate solution
        x0=x1;
        f0=f1;
        x1=x;
        f1=fx;
        criterion4=abs(x1-x0);
        counter=counter+1;
    end
    deltax=abs(x1-xexact);

    if (criterion4 ==0)
        criterion4=eps;
    end

    if (deltax == 0)
        deltax=eps;
    end

    alldeltax(i)=deltax;
    allcounter(i)=counter;
    allcriterion4(i)=criterion4;
```

```
    allxsolution(i)=x1;
end

figure(1);
semilogx(alldelta, allcounter,'-*');

figure(4);
loglog(alldelta, alldeltax,'-o');
title('Secant method');
xlabel('\Delta');
hold on;
loglog(alldelta, allcriterion4,'-s');
legend('\Delta x','criterion', 'Location', 'Northwest');
ylim([10^(-16) 10^(-2)]);
xlim([10^(-16) 10^(0)]);
hold off;

%==========================TASK 5==========================
%=======================Newton's method====================
for i=1:14

    df=@(x) (1./50).*x.*cos(x.^2./100)+(1./x)+1;
    x0=10;                              %starting point

    %first calculation
    x1=x0-f(x0)/df(x0);              %newton's formula
    criterion5=abs(x1-x0);
    delta=10^(-3)/10^(i-1);
    counter=0;
    f0=f(x0);
    f1=f(x1);

    while (criterion5>delta && counter<100)
        x=x1-f(x1)/df(x1);      %newton's formula
        x0=x1;
        x1=x;
        criterion5=abs(x1-x0);
        counter=counter+1;
    end
    deltax=abs(x1-xexact);

    if (criterion5 ==0)
        criterion5=eps;
    end

    if (deltax == 0)
        deltax=eps;
    end

    alldeltax(i)=deltax;
    allcounter(i)=counter;
    allcriterion5(i)=criterion5;
    allxsolution(i)=x1;
end

figure(1);
semilogx(alldelta, allcounter, '-o');
legend('Bisection','Regula-falsi', 'Secant', 'Newton');
hold off;

figure(5);
loglog(alldelta, alldeltax,'-o');
title('Newton method');
xlabel('\Delta');
hold on;
loglog(alldelta, allcriterion5,'-s');
legend('\Delta x','criterion', 'Location', 'Northwest');
ylim([10^(-16) 10^(-2)]);
xlim([10^(-16) 10^(0)]);
hold off;
```