Seth Goodman
APSC 607
Fall 2017

Project 01

Overview

This project explored different methods of approximating the roots of six different functions. Functions were examined over one or more non overlapping ranges, where unique roots exist, using the Bisection, Newton's, and Secant methods with varying tolerances. The outputs for each function and range are detailed in the *Results* portion of this report.

All computations were performed using MATLAB. Derivations used for approximation with Newton's method were performed first by hand (see Appendix) using basic derivative rules[1] for each term in the equation. All derivatives were confirmed using Wolfram Alpha[2] as well as MATLAB's differentiation function[3] from the Symbolic Math Toolbox.

Along with this report, the zip file it came in includes all code and outputs. The M files containing code run in MATLAB can be found alongside the report in the root directory and corresponding outputs in the *output* subdirectory.

Code:

- *main.m* - Primary script which runs all code. Includes definitions of the function examined and other variable definitions such as tolerance and ranges. Manages plotting and table outputs.
- *bisection.m* - Function implementing Bisection method
- *newton.m* - Function implementing Newton's method
- *secant.m* - Function implementing Secant method
- *modified_newton.m* - Function implementing modified version of Newton's method for dealing with multiple roots (Note: not used in core code and not fully tested)

The code for the Bisection, Newton's, and Secant methods are largely standardized (except where input parameters vary slightly based on specifics of method) and accept the function to examine, tolerance, number of iterations and key parameter such as starting point. In addition, each function accepts options which define how to plot the results and whether to print additional outputs when run (such as for examining results for each iteration or debugging). Each function produces similar plots, print statements (when enabled), and outputs the same four values: number of iterations run, root approximation at last iteration run, difference/error at final iteration, and a status code indicating success/error (e.g., failure to converge).

---

[1] https://www.mathsisfun.com/calculus/derivatives-rules.html
[2] https://www.wolframalpha.com/
[3] https://www.mathworks.com/help/symbolic/differentiation.html

Outputs:

- For each function-range combination (11 total) four figures are produced. The first is a plot of the true function over the specified range with the zero line and exact root value displayed. For each approximation method there is a figure containing two plots: root approximation at each iteration and difference/error at each iteration. The results for each of the three tolerance values are plotted side by side in each figure for comparison. The final values (approximation and difference) for each tolerance are included the legends.
- A final table containing results from all function-range-method-tolerance combinations is output as a CSV file.

All code and figures found in the zip file can also be accessed via GitHub

- https://github.com/sgoodm/apsc607/tree/master/project_01

Results

The results for the six functions examined in this project are broken down into sections based on the function, and subsections based on the range for the specified function. Each subsection for the eleven function-range combinations contains a plot of the true function over the range (with some padding for improved visualization), figures showing the outcome of approximation using each method (Bisection, Newton's, Secant), and finally a table which summarizes the results. Additional figures may be included to illustrate particular aspects of how methods perform or situations unique to a particular function.

The first subsection will explain the format of the figures, provide information about how they were generated, and go over the behavior of the methods in some detail. Subsequent subsections will only include noteworthy remarks specific to that function-range (when needed) rather than providing unnecessary or repetitive details that were either already mentioned or are evident from figures and table(s) provided.

Function A

$$e^x + 2^{-x} + 2 * cos(x) - 6 = 0$$

A1) Range *1 : 2*

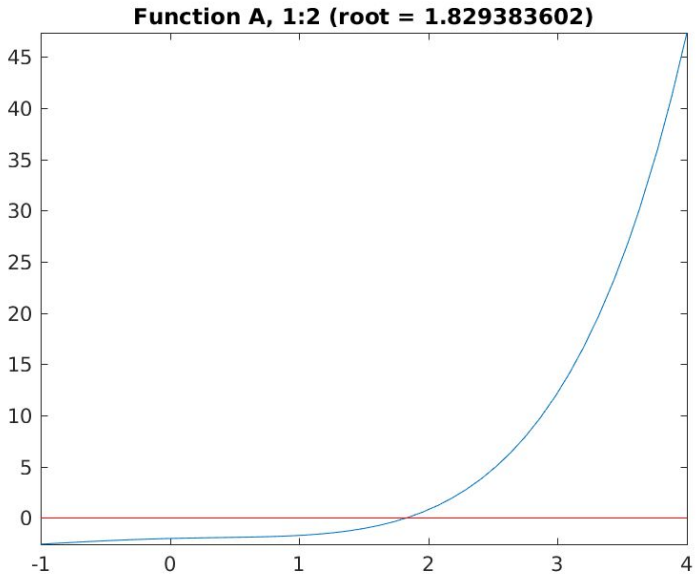

Figure A1-1

Figure A1-1 is a plot of the true function over the specified range (1 to 2) generated using *fplot* function in MATLAB, along with a zero line to identify where roots exist. The actual root which exists within this range was calculated using the *fzero* function in order to validate the results of the approximation methods.

In each of the following figures, the results for all three tolerance settings are plotted alongside each other. As indicated by the legends, the largest tolerance (10E-3) appears as a dark blue dashed line[4], the middle tolerance (10E-5) appears as light blue dashed line, and the smallest tolerance (10E-8) appears as pink circular points.

The number of iterations required to reach the specified tolerance is indicated in the legends, as are the final approximation values (value at last iteration) and the final difference/error value (difference at last iteration; hereafter referred to as *difference value*, rather than *error*).

The equation for the difference value varies slightly between methods. For the Bisection method, the difference is defined as half the difference between the points *a* and *b* which are being bisected in the current iteration (i.e., the difference between *a* and *b* in the *next* iteration). For Newton's method, the difference is the absolute difference between *p* and *p0*. And similarly, for the Secant method, it is the absolute difference between *p* and *p1*.[5]

Bisection Method

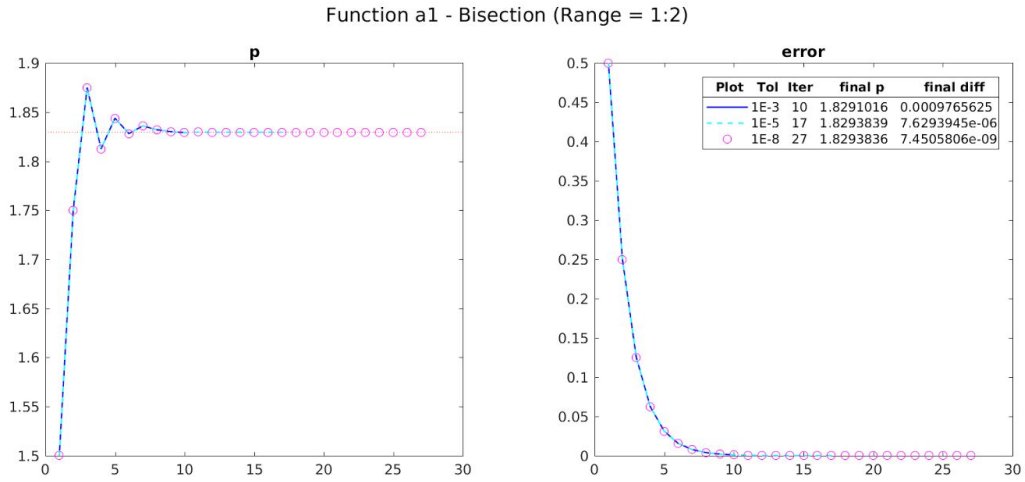

Figure A1-2 - Bisection method

[4] Technically this is a solid blue line, but the overlaid light blue dashed line makes the solid dark blue line appear dashed.
[5] See code (https://github.com/sgoodm/apsc607/tree/master/project_01) for variable references

Newton's Method

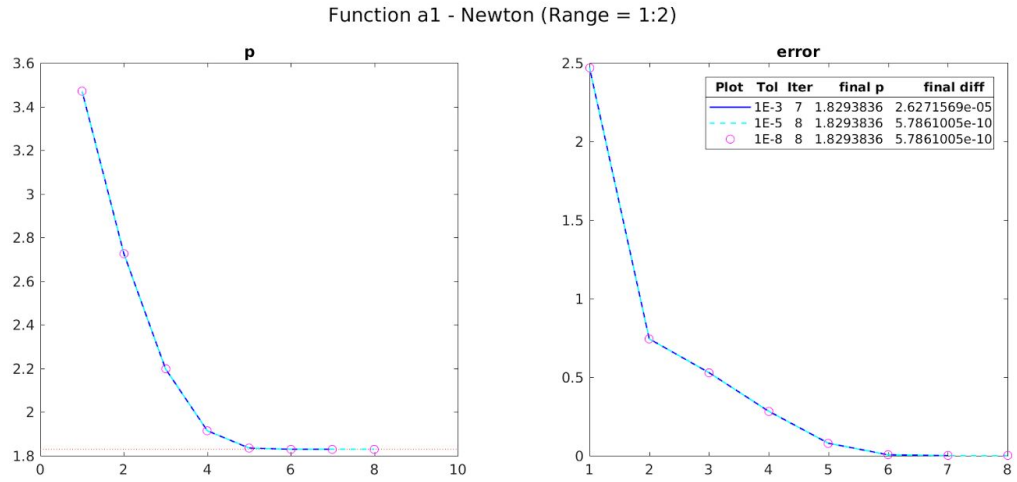

Figure A1-3 - Newton's method with default starting point

For Newton's method, the initial starting point used is the range minimum. A better starting point can generally be used to reduce the number of iterations needed for convergence. The range minimum (*p0* = 1, figure A1-3) was compared to a value of approximately 90% of the result from the Bisection method (*p0* = 1.65, figure A1-4) in table A1-1.

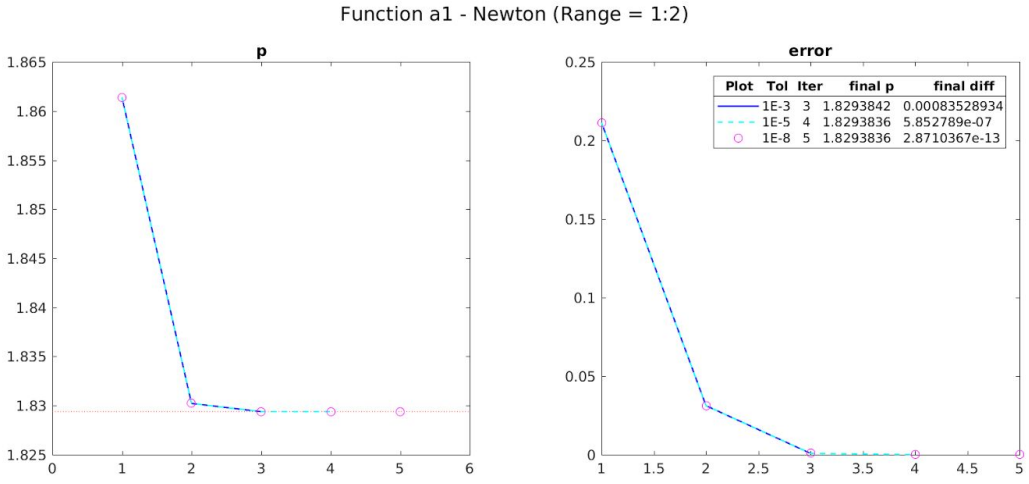

Figure A1-4 - Newton's method with improved starting point

Table A1-1 - Comparison of starting points for Newton's method

Method	Tolerance	p0	N	p	Max_Error
newton	0.001	1	7	1.8293836	2.63E-05
newton	0.00001	1	8	1.8293836	5.79E-10

newton	0.00000001	1	8	1.8293836	5.79E-10
newton	0.001	1.65	3	1.8293842	0.0008352893
newton	0.00001	1.65	4	1.8293836	5.852789E-7
newton	0.00000001	1.65	5	1.8293836	2.8710367E-13

Improving the starting point for Newton's method reduced the number of iterations required to achieve the specified tolerance by about 50%. In situations where significant iterations (and run times) are needed to reach an approximation within the desired tolerance, improving the starting point estimate would likely be beneficial. For most of the cases examined in this project, sufficient iterations can be run very quickly and using the range minimum as a starting point for Newton's method is sufficient.

Two of the examples explored later in this report (C2 and E1) will deal with an additional scenario where adjusting the starting point is necessary due to convergence on a root outside the specified range when using the range minimum.

Secant Method

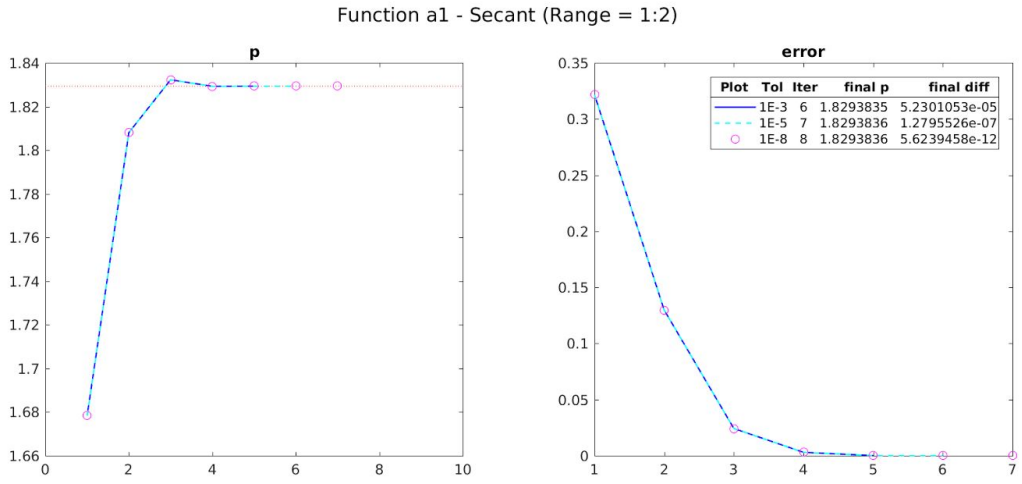

Figure A1-5 - Secant method

Summary of Results

Table A1-2 - Comparison of methods

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	1	2	10	1.8291016	0.0009765625
bisection	0.00001	1	2	17	1.8293839	7.63E-06

bisection	0.00000001	1	2	27	1.8293836	7.45E-09
newton	0.001	1	2	3	1.8293842	0.00083528934
newton	0.00001	1	2	4	1.8293836	5.852789E-07
newton	0.00000001	1	2	5	1.8293836	2.8710367E-13
secant	0.001	1	2	6	1.8293835	5.23E-05
secant	0.00001	1	2	7	1.8293836	0.000000128
secant	0.00000001	1	2	8	1.8293836	5.62E-12

While the Bisection method performed only slightly worse than the other methods at the largest tolerance, as tolerance decreased the Newton's method (using the adjusted starting point of 1.65, mentioned above) and the Secant method were able to converge notably faster. This can be attributed to the linear convergence of the Bisection method compared to the quadratic convergence of Newton's method and the Secant method. Although Newton's method in particular can provide highly accurate approximations with fewer iterations, it does so by assuming:

$$(p - p_0)^2 << |p - p_0|$$

For the terms as seen in the below equation[6] for the first Taylor polynomial:

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + \frac{(p - p_0)^2}{2} f''(\xi(p)),$$

Which reduces to the simplified equation for Newton's Method:

$$p \approx p_0 - \frac{f(p_0)}{f'(p_0)} \equiv p_1.$$

This results in an accurate approximation that requires few iteration, given an accurate starting point. In addition to necessitating an accurate starting point estimation, Newton's method also requires calculating the derivative of the main function (and in certain cases dealing with multiple roots, the second derivative) which can become computationally intensive for complicated functions (not an issue that will be explored in this project).

The final difference value is smallest for the smallest tolerance when using Newton's method, while requiring the fewest iterations. Going beyond the tolerance range for this project, to values of 1E-10, 1E-12, and 1E-15 we can see that Newton's method (as well as the Secant method)

[6] Burden, R., Faires, J., Numerical Analysis. 2010

will effectively converge on a zero difference value (though this may simply be such a small number that the computation displays it as zero[7].

Table A1-3 - Comparison of method with smaller tolerances

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	1.00E-10	1	2	34	1.8293836	5.82E-11
bisection	1.00E-12	1	2	40	1.8293836	9.09E-13
bisection	1.00E-15	1	2	50	1.8293836	8.88E-16
newton	1.00E-10	1	2	5	1.8293836	2.87E-13
newton	1.00E-12	1	2	5	1.8293836	2.87E-13
newton	1.00E-15	1	2	6	1.8293836	0
secant	1.00E-10	1	2	8	1.8293836	5.62E-12
secant	1.00E-12	1	2	9	1.8293836	0
secant	1.00E-15	1	2	9	1.8293836	0

As seen in the table above, Newton's method required only a single additional iteration with a reasonably accurate starting point (1.65) to reach a tolerance of 1E-15 (as did the Secant method) compared to a tolerance of 1E-8, while the Bisection method require 23 additional iterations.

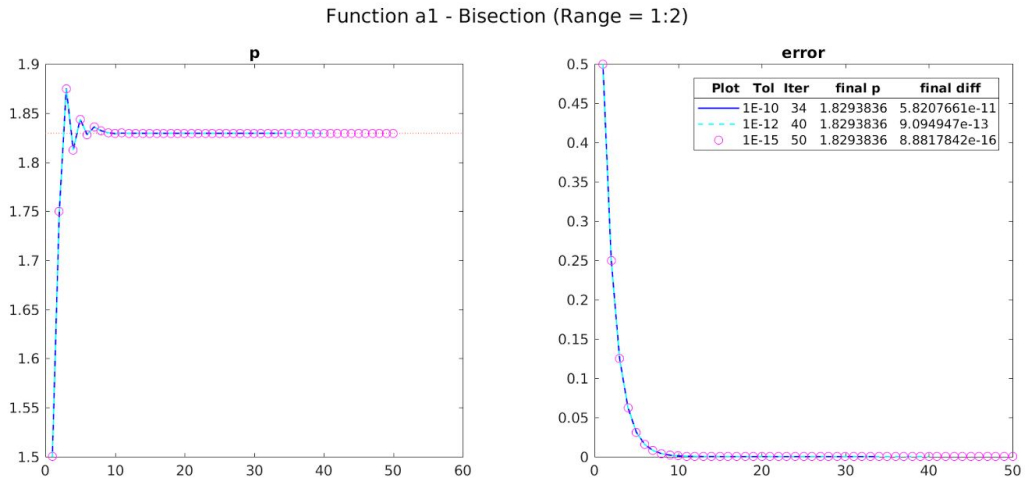

Figure A1-6 - Bisection method used with smaller tolerances

[7] See following link on MATLAB precision limitions (general limitations of floating point representations apply) https://www.mathworks.com/help/fixedpoint/ug/limitations-on-precision.html

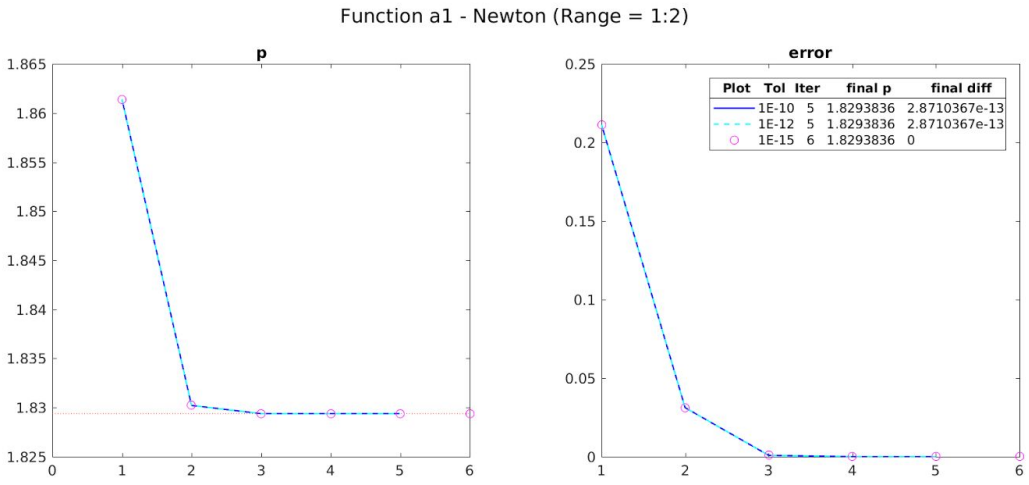

Figure A1-7 - Newton's method used with smaller tolerances

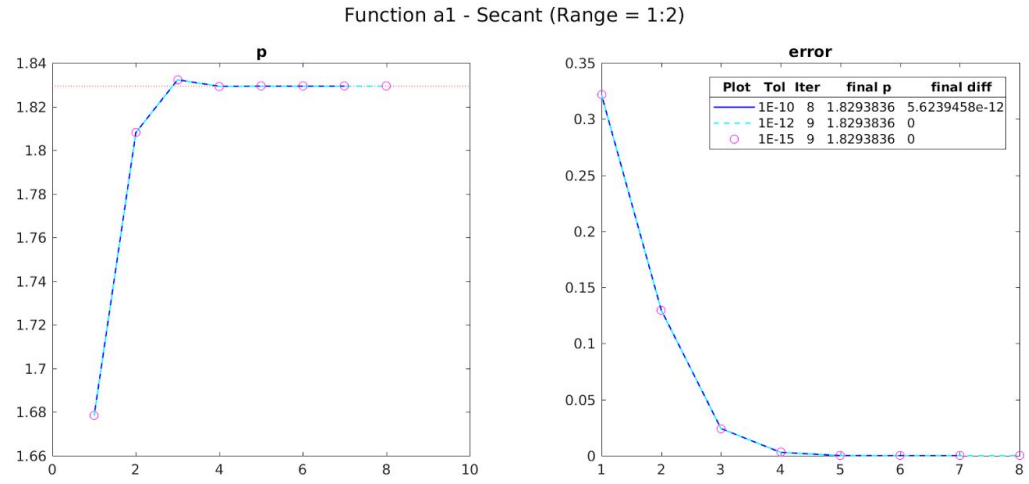

Figure A1-8 - Secant method used with smaller tolerances

Function B

$$log(x-1)+cos(x-1)=0$$

B1) Range *1.3 : 2*

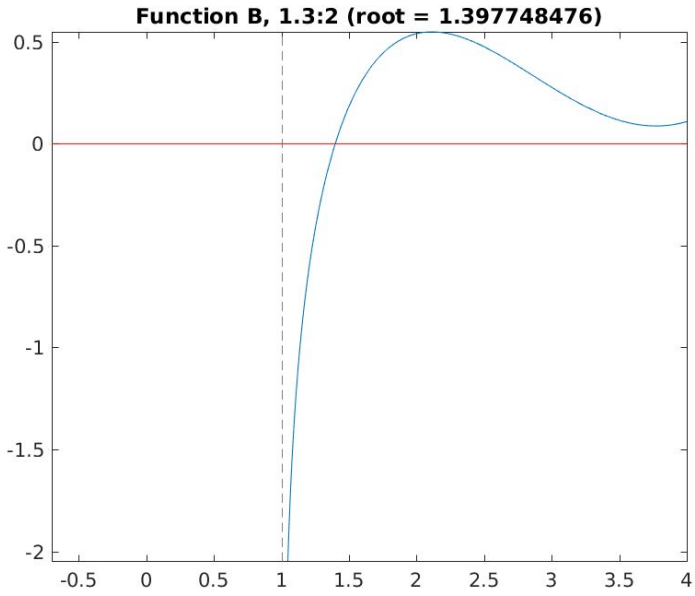

Figure B1-1

Bisection Method

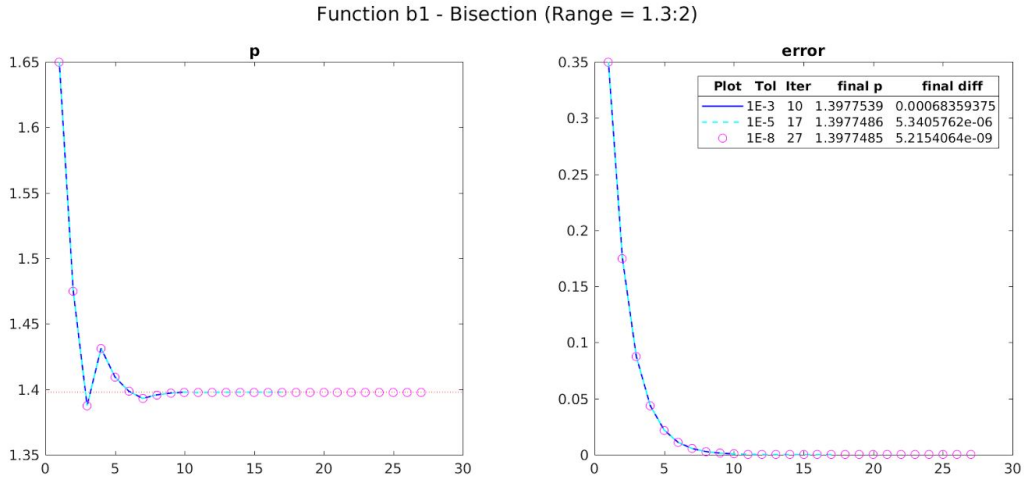

Figure B1-2

Newton's Method

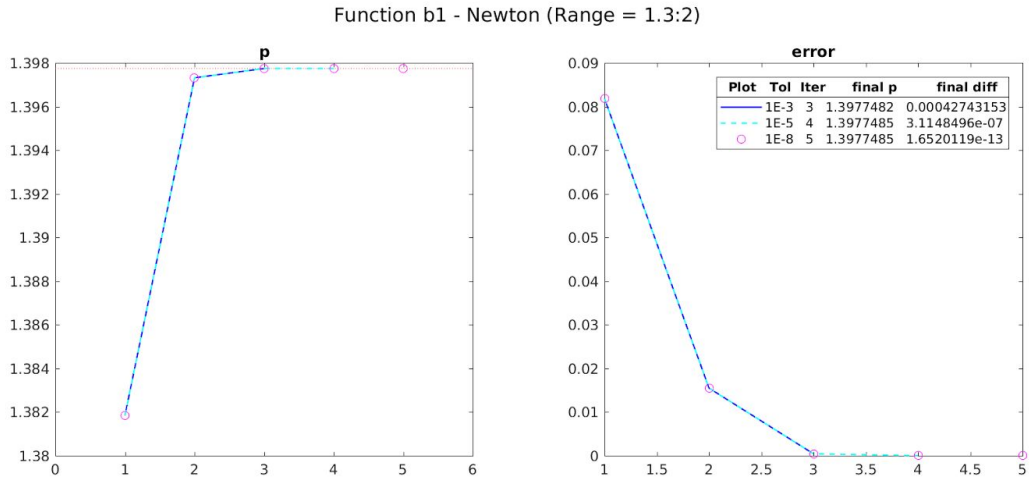

Figure B1-3

Secant Method

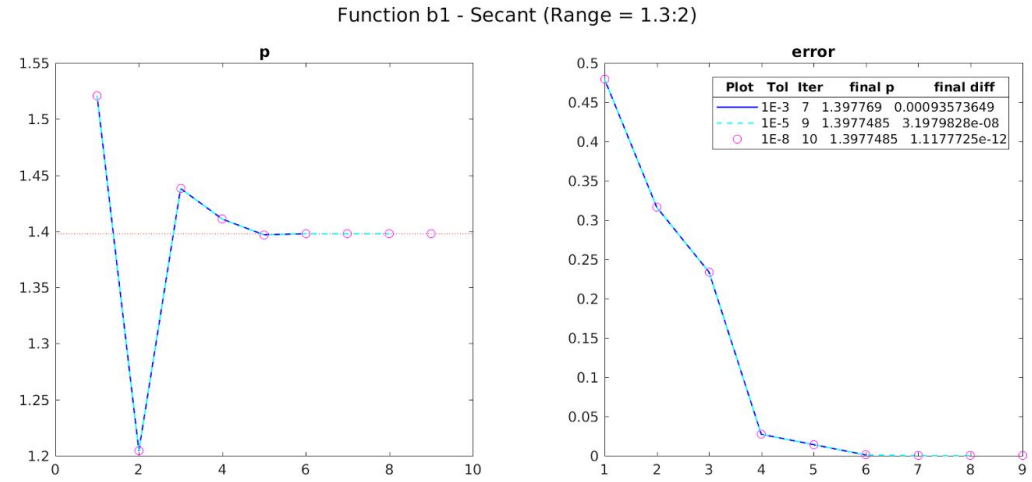

Figure B1-4

Summary of Results

Table B1-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	1.3	2	10	1.3977539	0.0006835938

bisection	0.00001	1.3	2	17	1.3977486	5.34E-06
bisection	0.00000001	1.3	2	27	1.3977485	5.22E-09
newton	0.001	1.3	2	3	1.3977482	0.0004274315
newton	0.00001	1.3	2	4	1.3977485	3.11E-07
newton	0.00000001	1.3	2	5	1.3977485	1.65E-13
secant	0.001	1.3	2	7	1.397769	0.0009357365
secant	0.00001	1.3	2	9	1.3977485	0.000000032
secant	0.00000001	1.3	2	10	1.3977485	1.12E-12

Function C

$$2 * x * cos(2 * x) - (x - 2)^2 = 0$$

C1) Range *2 : 3*

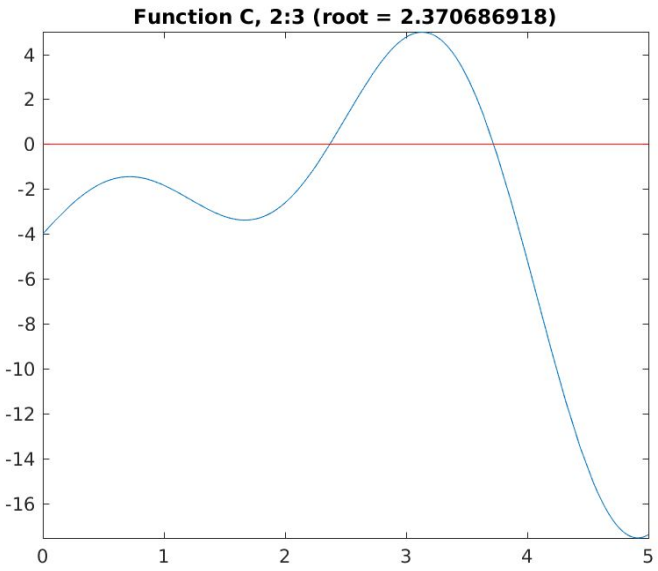

Figure C1-1

Bisection Method

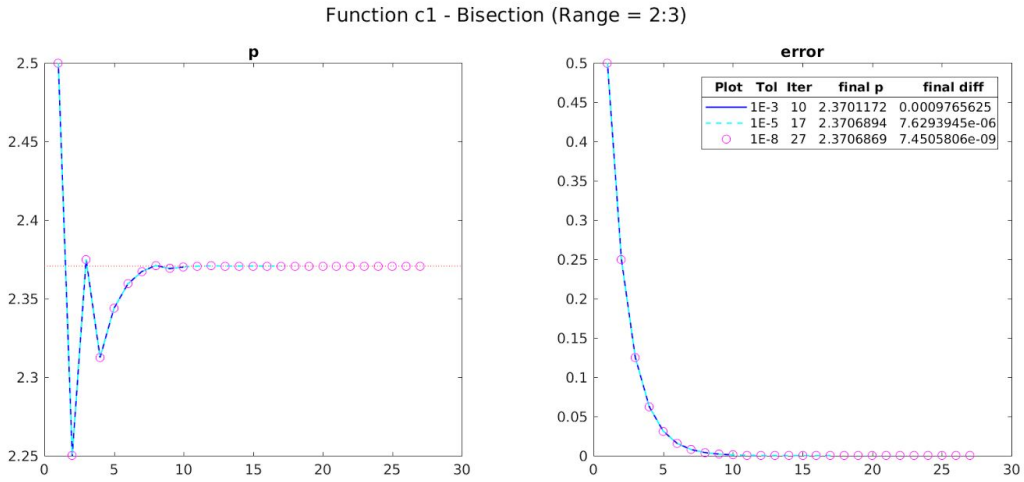

Figure C1-2

Newton's Method

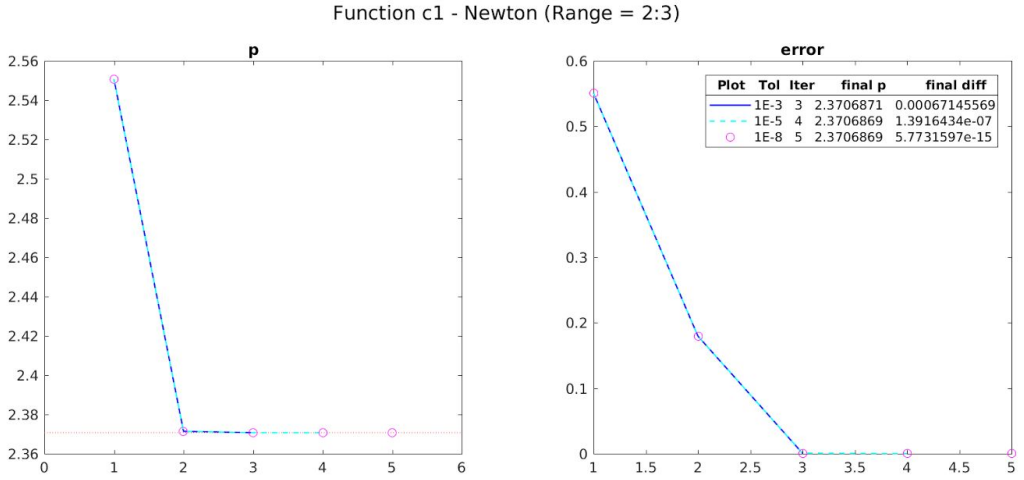

Figure C1-3

Secant Method

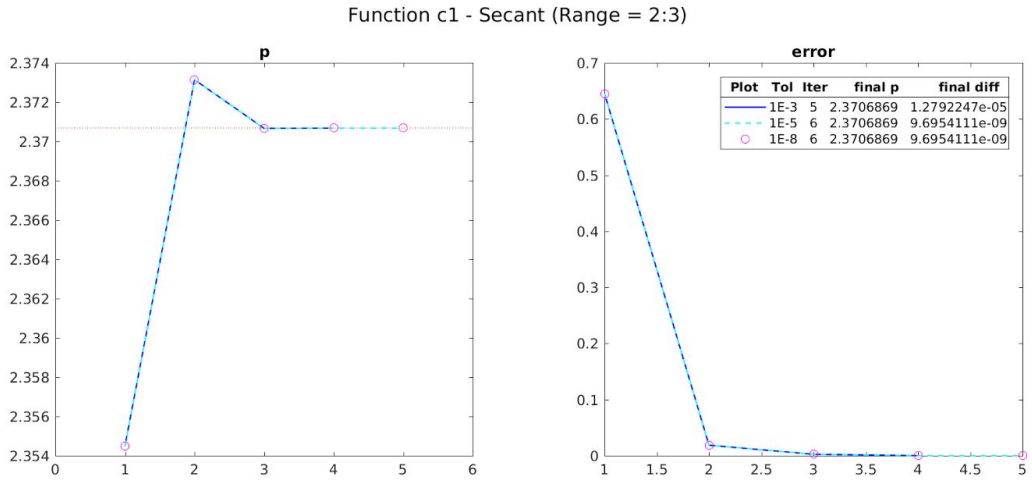

Figure C1-4

Summary of Results

Table C1-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	2	3	10	2.3701172	0.0009765625
bisection	0.00001	2	3	17	2.3706894	7.63E-06
bisection	0.00000001	2	3	27	2.3706869	7.45E-09
newton	0.001	2	3	3	2.3706871	0.0006714557
newton	0.00001	2	3	4	2.3706869	1.39E-07
newton	0.00000001	2	3	5	2.3706869	5.77E-15
secant	0.001	2	3	5	2.3706869	1.28E-05
secant	0.00001	2	3	6	2.3706869	9.70E-09
secant	0.00000001	2	3	6	2.3706869	9.70E-09

C2) Range *3 : 4*

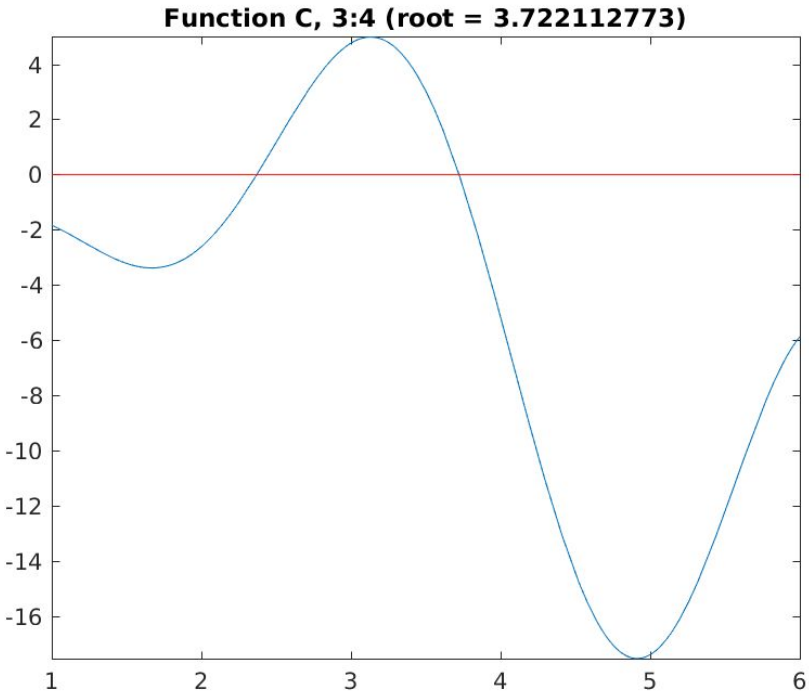

Figure C2-1

Bisection Method

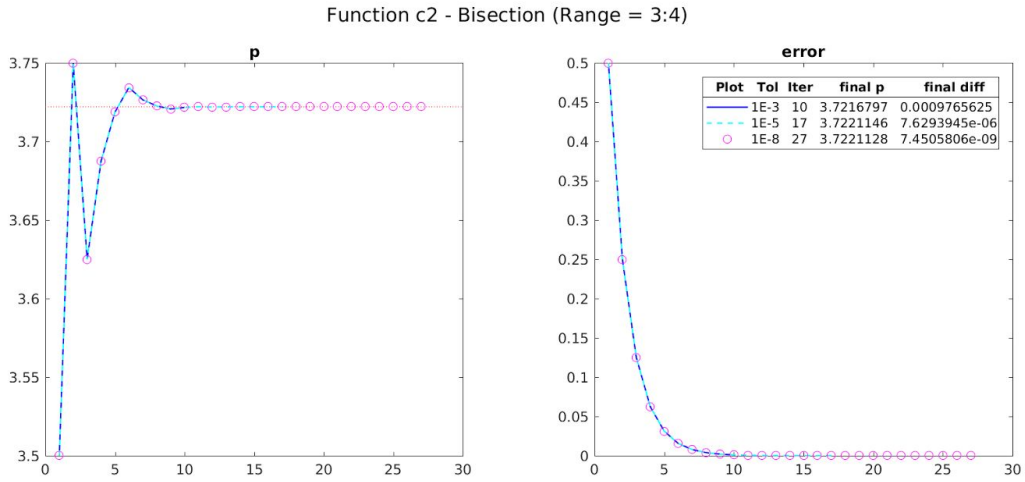

Figure C2-2

Newton's Method

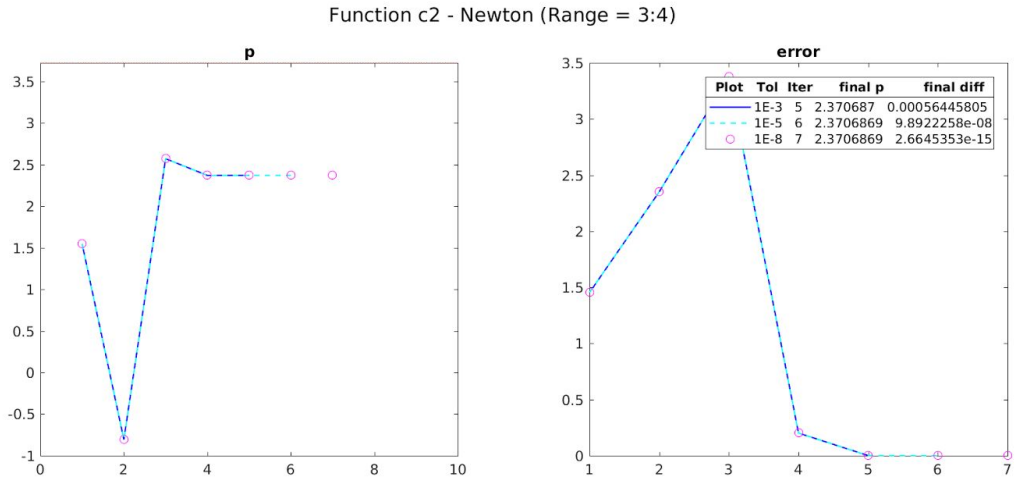

Figure C2-3

As seen in figure C2-3 for Newton's method above, this is an instance where the initial starting point for Newton's method results in convergence on the wrong root. Due to a local maximum within the specified range, and roots on either side of the maximum, a starting point on one side of the local maximum is likely to converge on the root on the same side of the local maximum. When the starting point is set to the range minimum, zero, it converges on the root of 2.37 which as they are both to left of the local maximum. By adjusting the starting point to be on right of the local maximum, the same as our desired root, Newton's method will converge on the desired root of 3.72, as seen in figure C2-4.

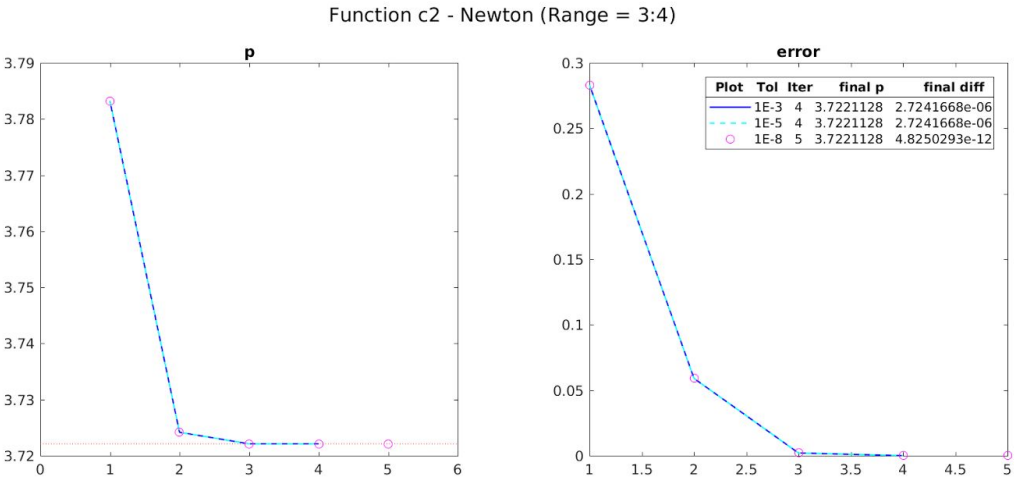

Figure C2-4

Secant Method

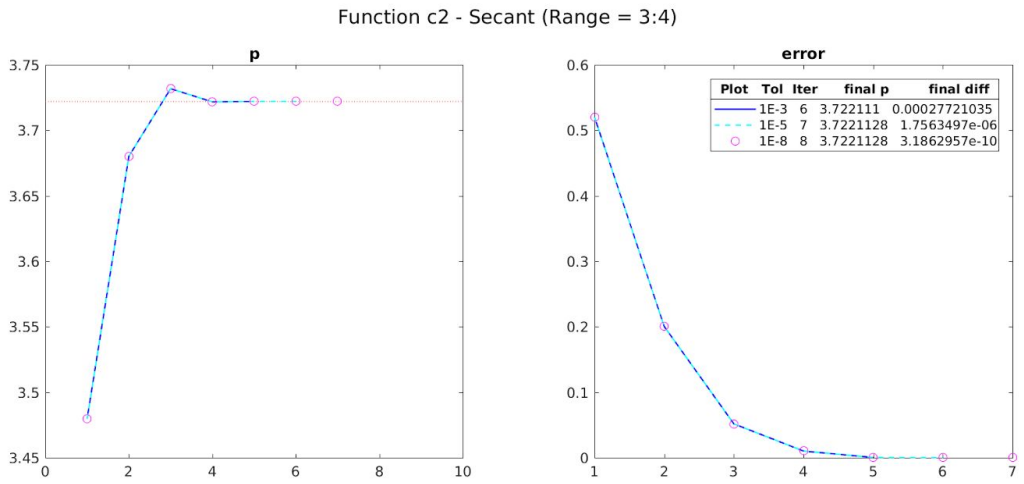

Figure C2-5

Summary of Results

Table C2-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	3	4	10	3.7216797	0.0009765625
bisection	0.00001	3	4	17	3.7221146	7.63E-06
bisection	0.00000001	3	4	27	3.7221128	7.45E-09
newton	0.001	3	4	4	3.7221128	2.72E-06
newton	0.00001	3	4	4	3.7221128	2.72E-06
newton	0.00000001	3	4	5	3.7221128	4.83E-12
secant	0.001	3	4	6	3.722111	0.0002772104
secant	0.00001	3	4	7	3.7221128	1.76E-06
secant	0.00000001	3	4	8	3.7221128	3.19E-10

Function D

$$(x-2)^2 - log(x) = 0$$

D1) Range *1 : 2*

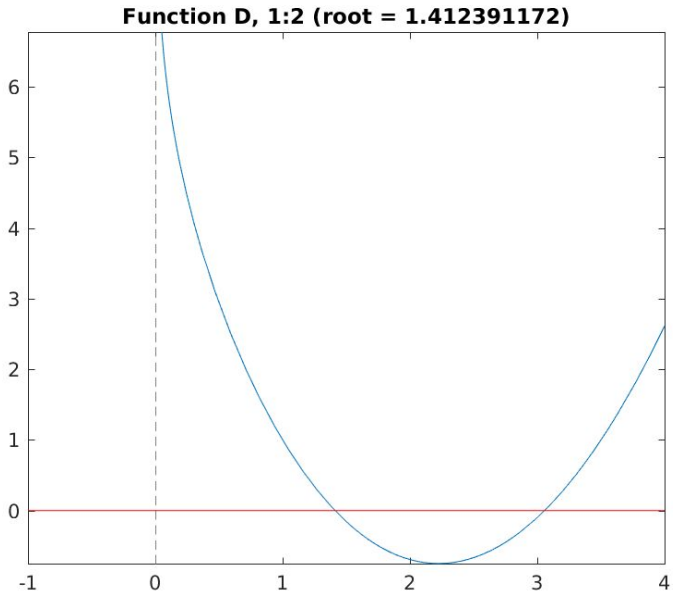

Figure D1-1

Bisection Method

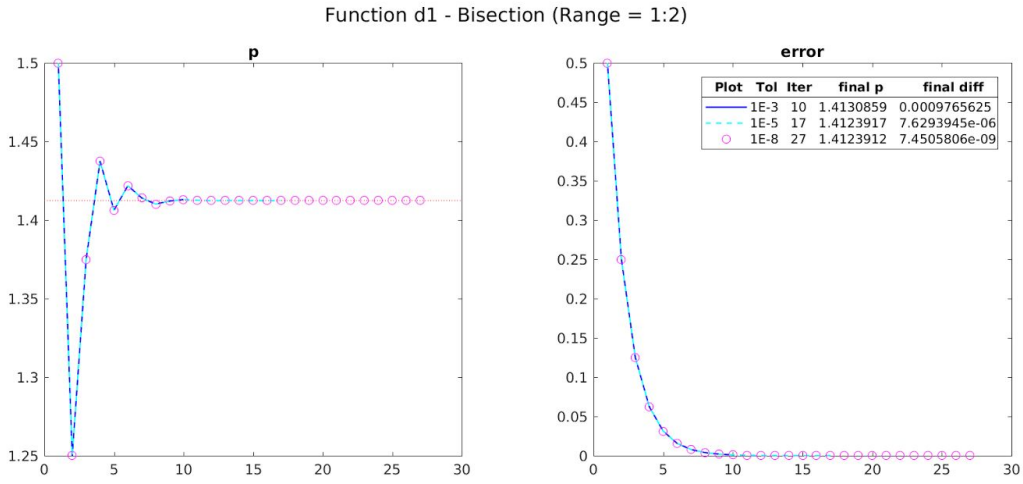

Figure D1-2

Newton's Method

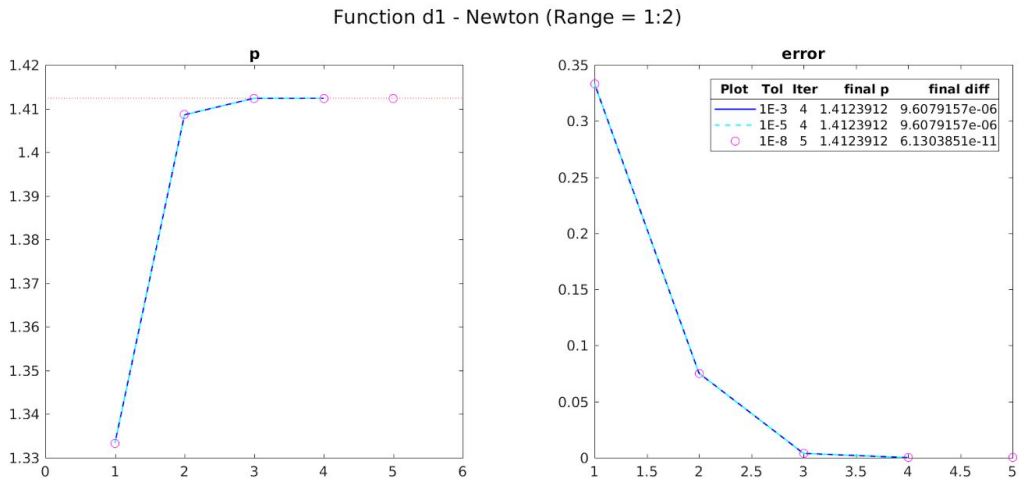

Figure D1-3

Secant Method

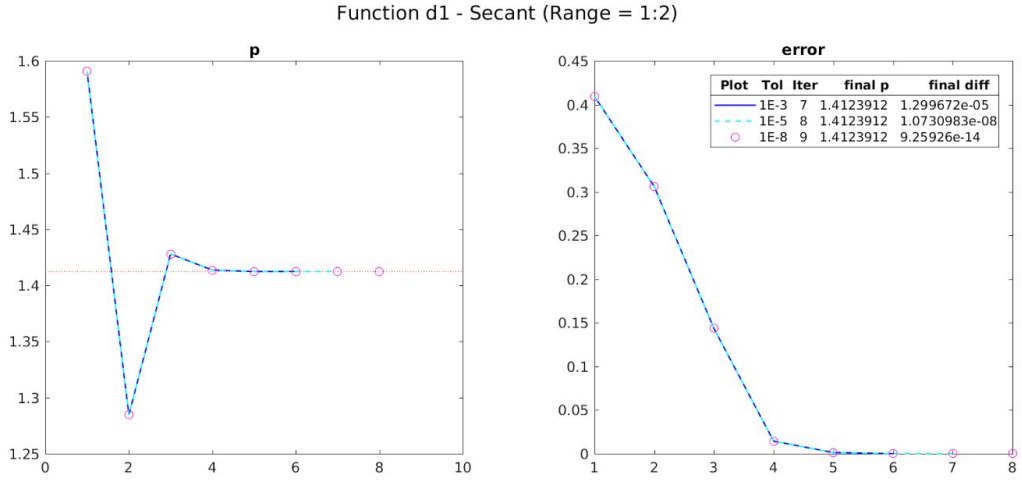

Figure D1-4

Summary of Results

Table D1-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	1	2	10	1.4130859	0.0009765625
bisection	0.00001	1	2	17	1.4123917	7.63E-06
bisection	0.00000001	1	2	27	1.4123912	7.45E-09
newton	0.001	1	2	4	1.4123912	9.61E-06

newton	0.00001	1	2	4	1.4123912	9.61E-06
newton	0.00000001	1	2	5	1.4123912	6.13E-11
secant	0.001	1	2	7	1.4123912	1.30E-05
secant	0.00001	1	2	8	1.4123912	1.07E-08
secant	0.00000001	1	2	9	1.4123912	9.26E-14

D2) Range *e : 4*

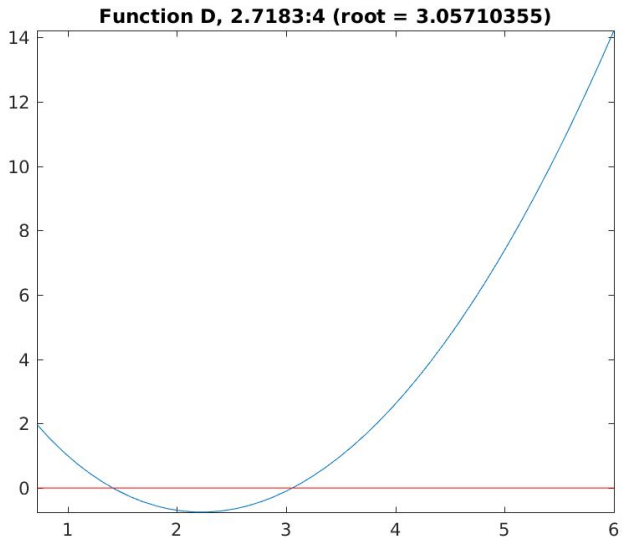

Figure D2-1

Bisection Method

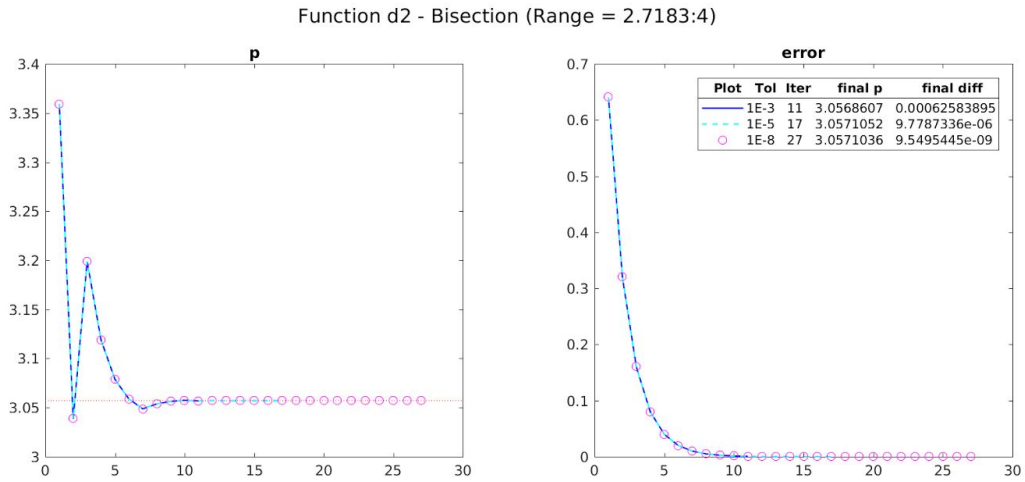

Figure D2-2

Newton's Method

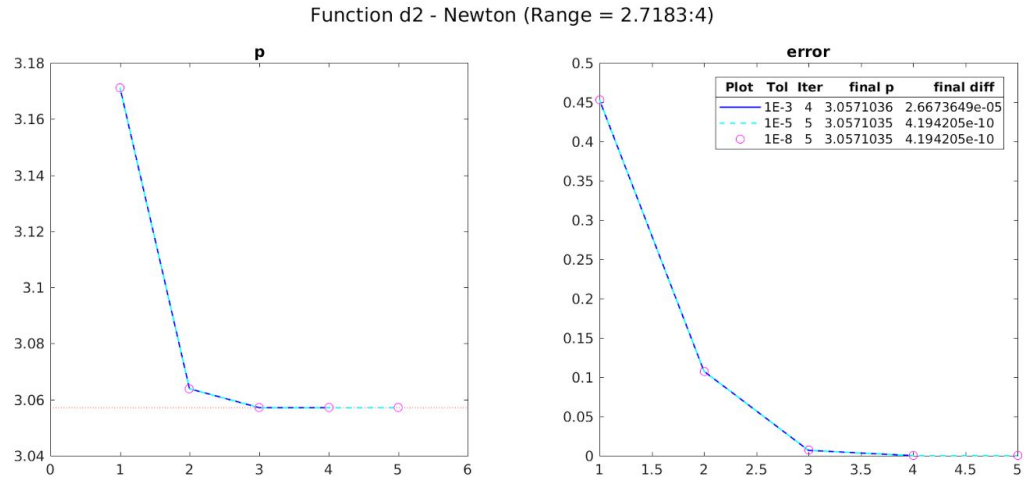

Figure D2-3

Secant Method

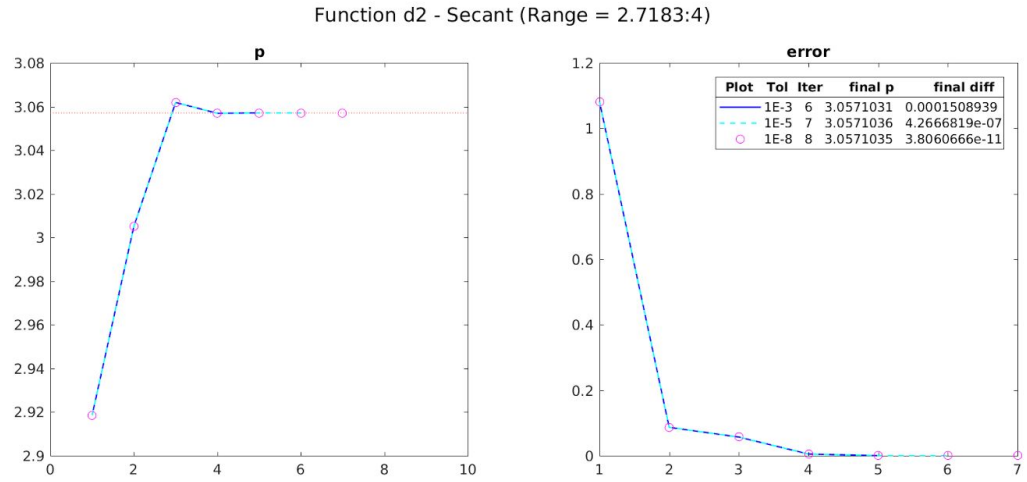

Figure D2-4

Summary of Results

Table D2-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	e	4	11	3.0568607	0.000625839
bisection	0.00001	e	4	17	3.0571052	9.78E-06
bisection	0.00000001	e	4	27	3.0571036	9.55E-09

newton	0.001	e	4	4	3.0571036	2.67E-05
newton	0.00001	e	4	5	3.0571035	4.19E-10
newton	0.00000001	e	4	5	3.0571035	4.19E-10
secant	0.001	e	4	6	3.0571031	0.0001508939
secant	0.00001	e	4	7	3.0571036	4.27E-07
secant	0.00000001	e	4	8	3.0571035	3.81E-11

Function E

$$e^x - 3 * x^2 = 0$$

E1) Range *0 : 1*

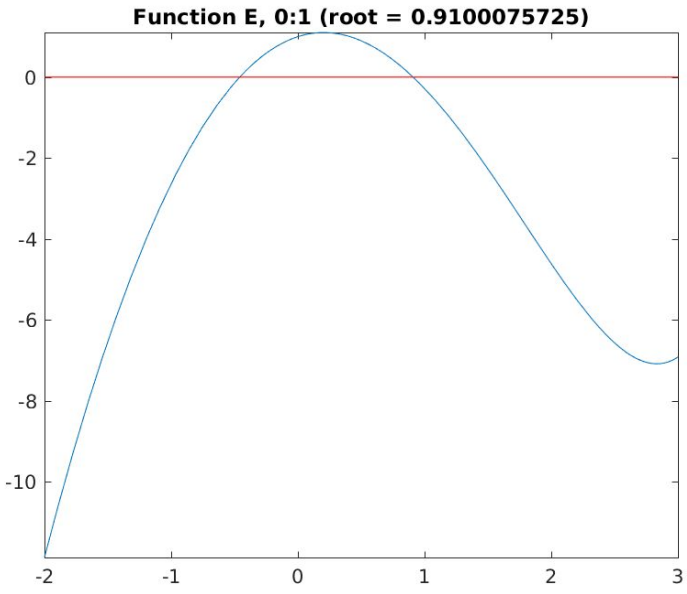

Figure E1-1

Bisection Method

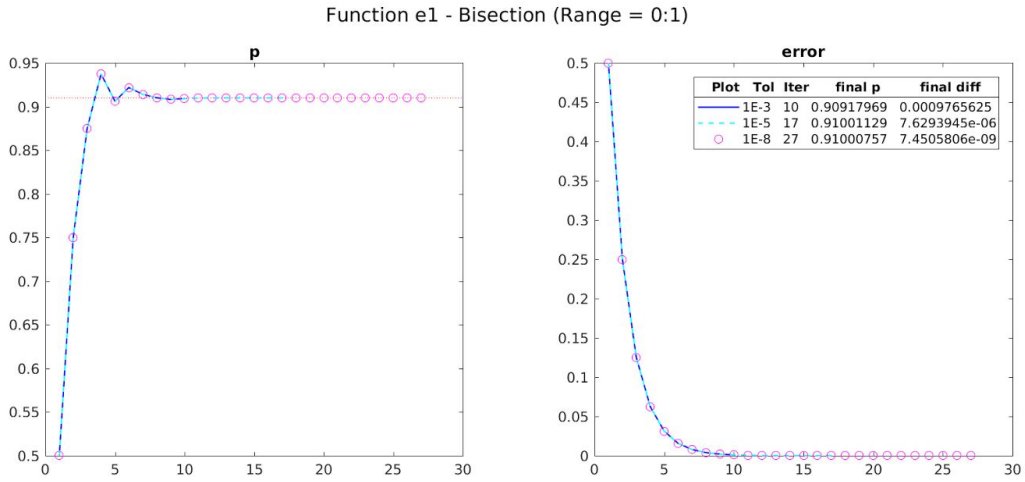

Figure E1-2

Newton's Method

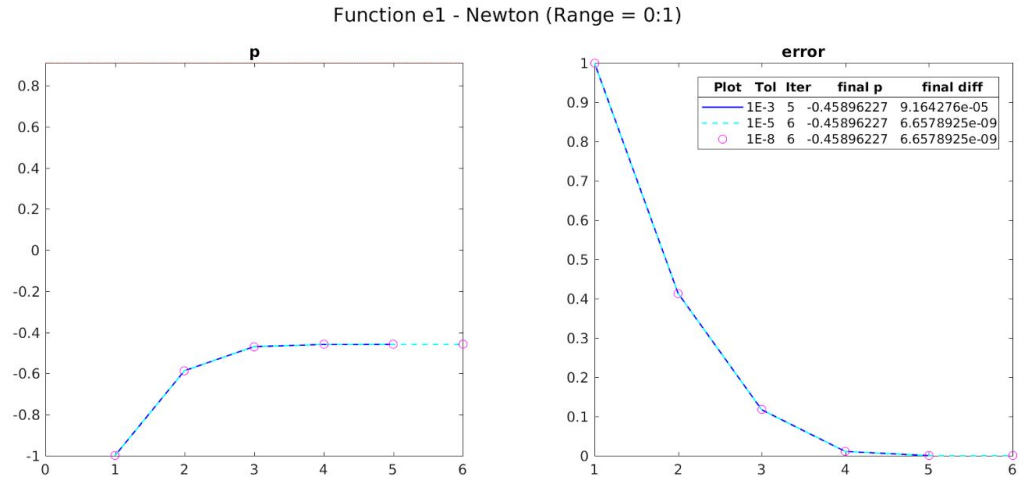

Figure E1-3

As seen in figure E1-3 for Newton's method, this is the second instance where the initial starting point for Newton's method results in convergence on the wrong root. Due to a local maximum within the specified range, and roots on either side of the maximum, a starting point on one side of the local maximum is likely to converge on the root on the same side of the local maximum. When the starting point is set to the range minimum, zero, it converges on the root of -0.45 which as they are both to left of the local maximum. By adjusting the starting point to be on right of the local maximum, the same as our desired root, Newton's method will converge on the desired root of 0.91, as seen in figure E1-4.

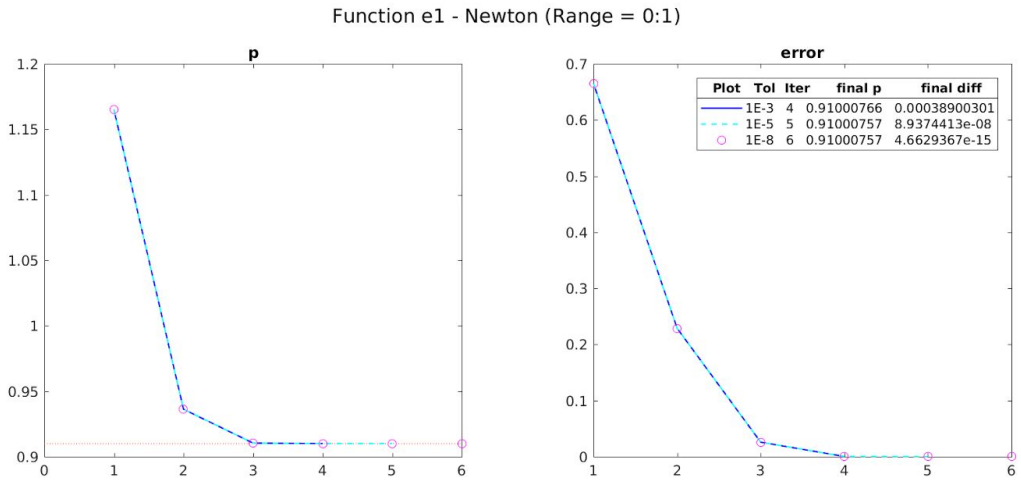

Figure E1-4

Secant Method

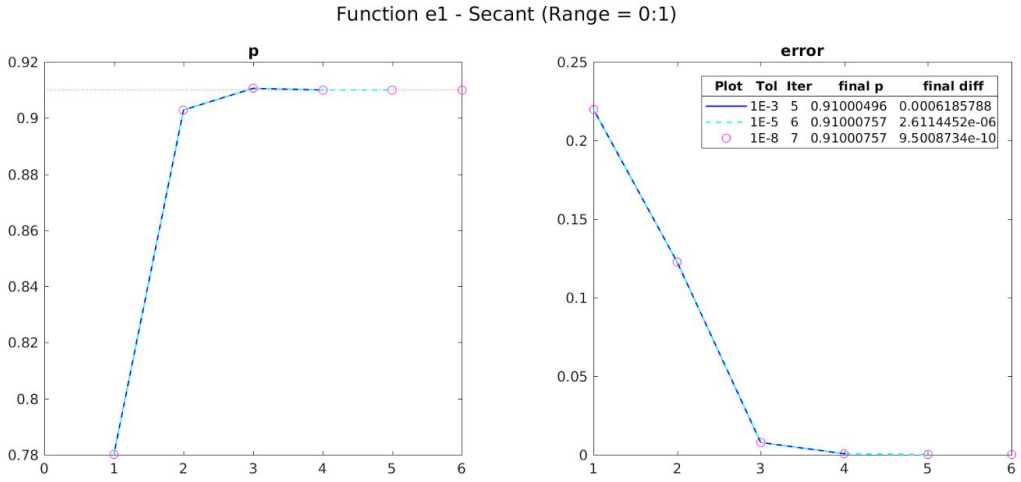

Figure E1-5

Summary of Results

Table E1-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	0	1	10	0.90917969	0.0009765625
bisection	0.00001	0	1	17	0.91001129	7.63E-06

bisection	0.00000001	0	1	27	0.91000757	7.45E-09
newton	0.001	0	1	4	0.91000766	0.000389003
newton	0.00001	0	1	5	0.91000757	8.94E-08
newton	0.00000001	0	1	6	0.91000757	4.66E-15
secant	0.001	0	1	5	0.91000496	0.0006185788
secant	0.00001	0	1	6	0.91000757	2.61E-06
secant	0.00000001	0	1	7	0.91000757	0.000000001

E2) Range *3 : 5*

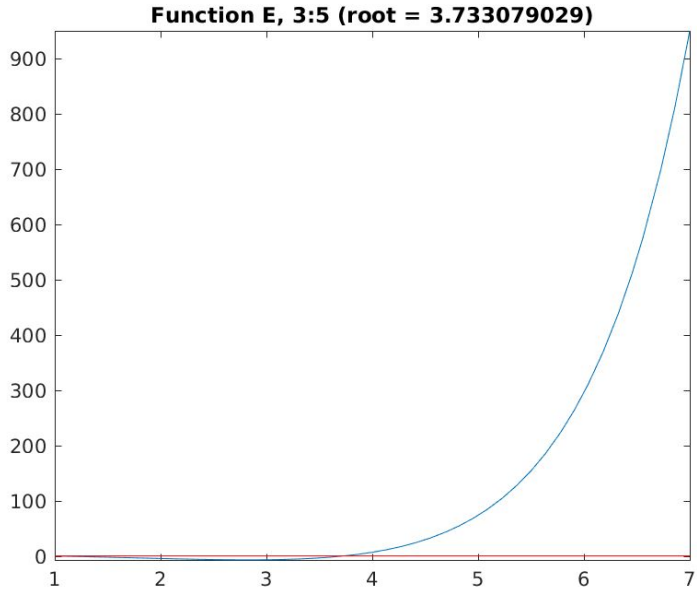

Figure E2-1

Bisection Method

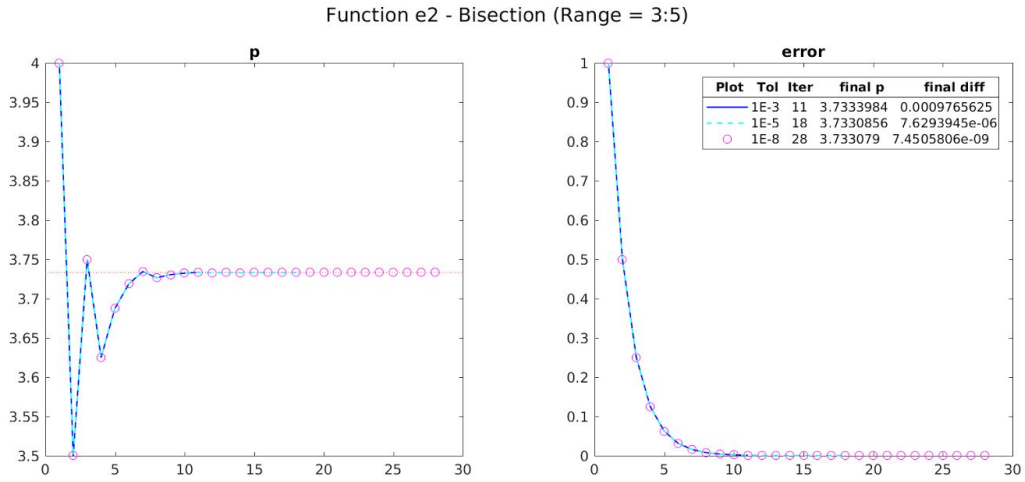

Figure E2-2

Newton's Method

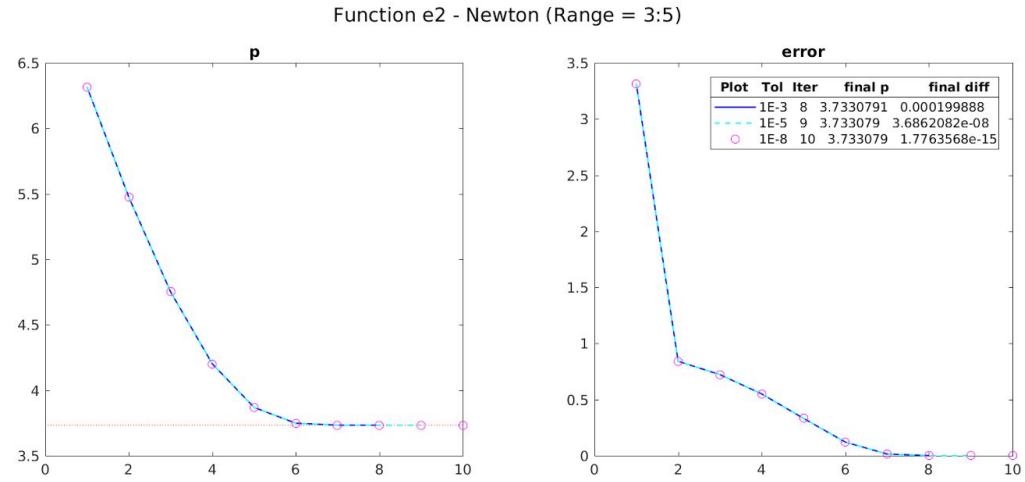

Figure E2-3

Secant Method

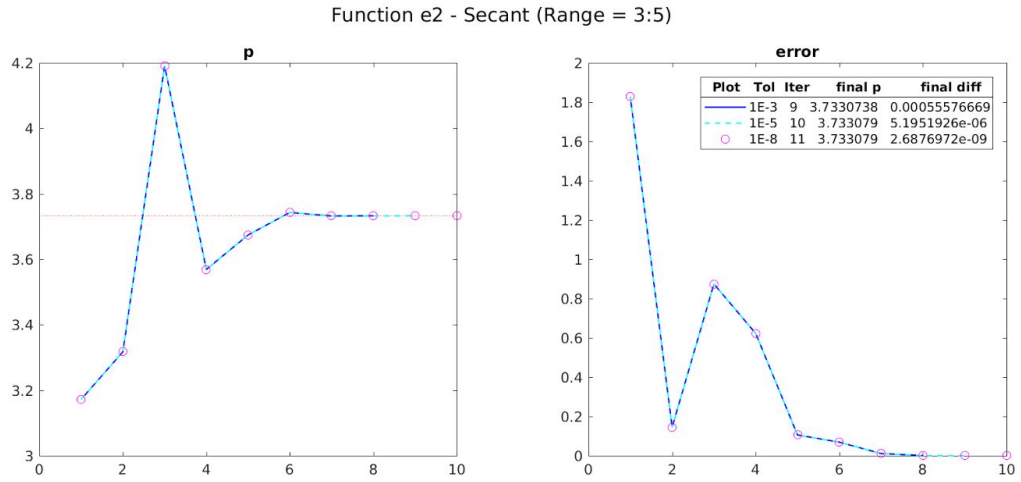

Figure E2-4

Summary of Results

Table E2-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	3	5	11	3.7333984	0.0009765625
bisection	0.00001	3	5	18	3.7330856	7.63E-06
bisection	0.00000001	3	5	28	3.733079	7.45E-09
newton	0.001	3	5	8	3.7330791	0.000199888
newton	0.00001	3	5	9	3.733079	3.69E-08
newton	0.00000001	3	5	10	3.733079	1.78E-15
secant	0.001	3	5	9	3.7330738	0.0005557667
secant	0.00001	3	5	10	3.733079	5.20E-06
secant	0.00000001	3	5	11	3.733079	2.69E-09

Function F

$$sin(x) - e^{-x} = 0$$

F1) Range *0 : 1*

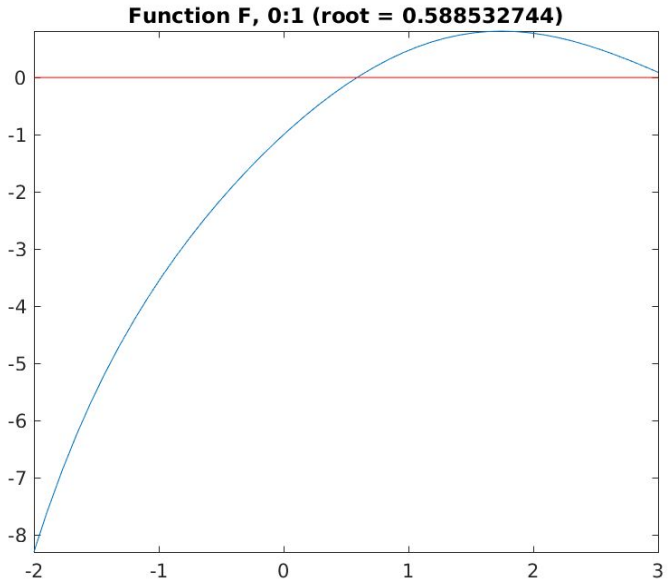

Figure F1-1

Bisection Method

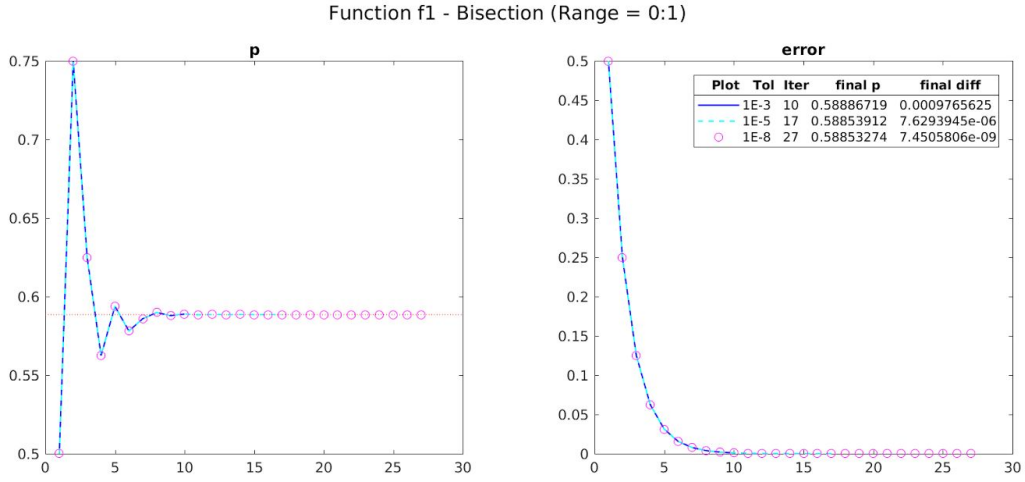

Figure F1-2

Newton's Method

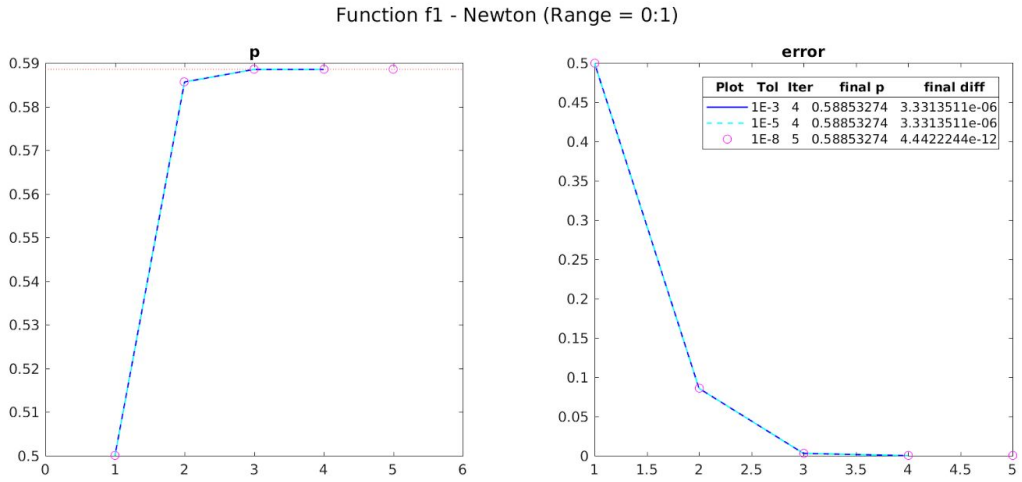

Figure F1-3

Secant Method

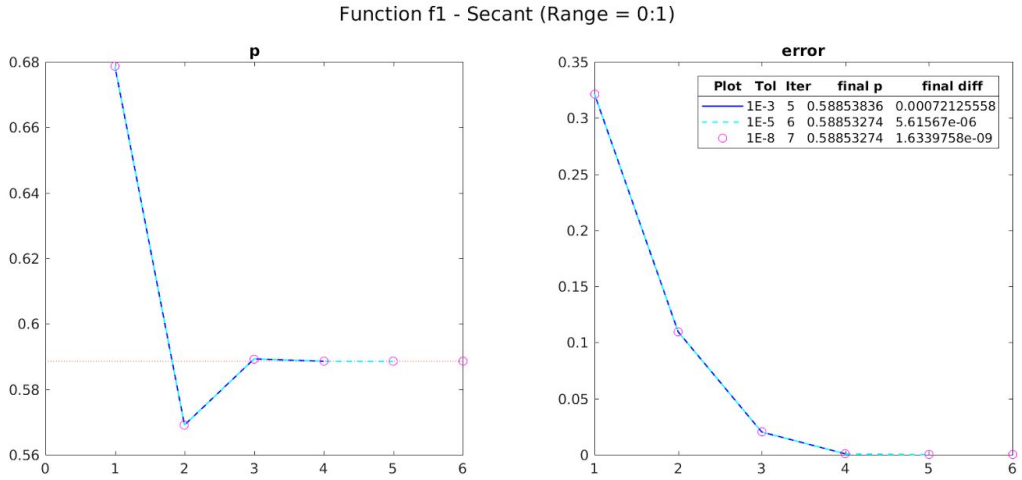

Figure F1-4

Summary of Results

Table F1-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	0	1	10	0.58886719	0.0009765625
bisection	0.00001	0	1	17	0.58853912	7.63E-06
bisection	0.00000001	0	1	27	0.58853274	7.45E-09
newton	0.001	0	1	4	0.58853274	3.33E-06

newton	0.00001	0	1	4	0.58853274	3.33E-06
newton	0.00000001	0	1	5	0.58853274	4.44E-12
secant	0.001	0	1	5	0.58853836	0.0007212556
secant	0.00001	0	1	6	0.58853274	5.62E-06
secant	0.00000001	0	1	7	0.58853274	1.63E-09

F2) Range *3 : 4*

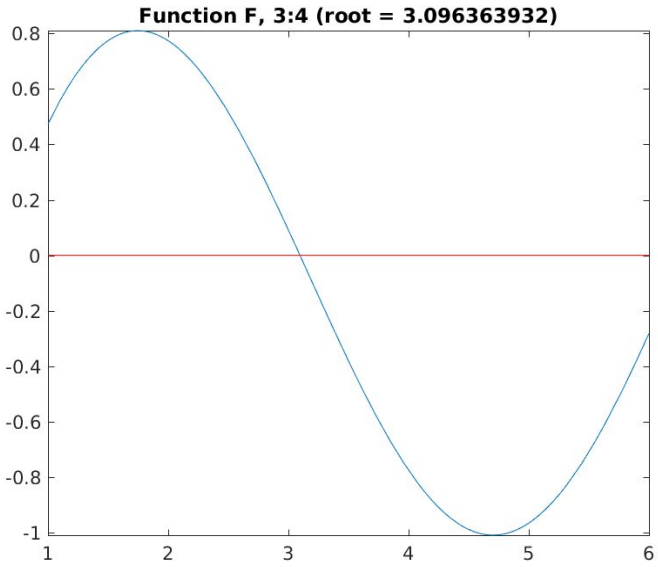

Figure F2-1

Bisection Method

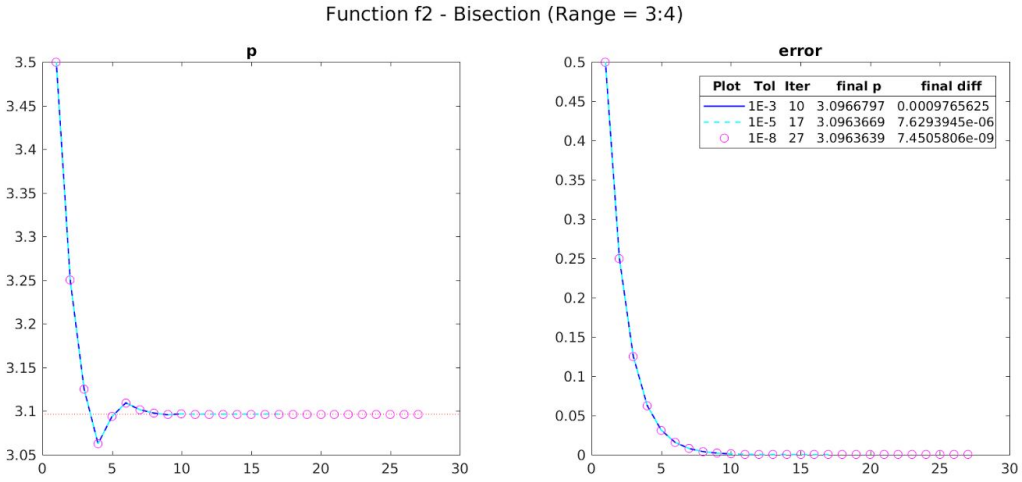

Figure F2-2

Newton's Method

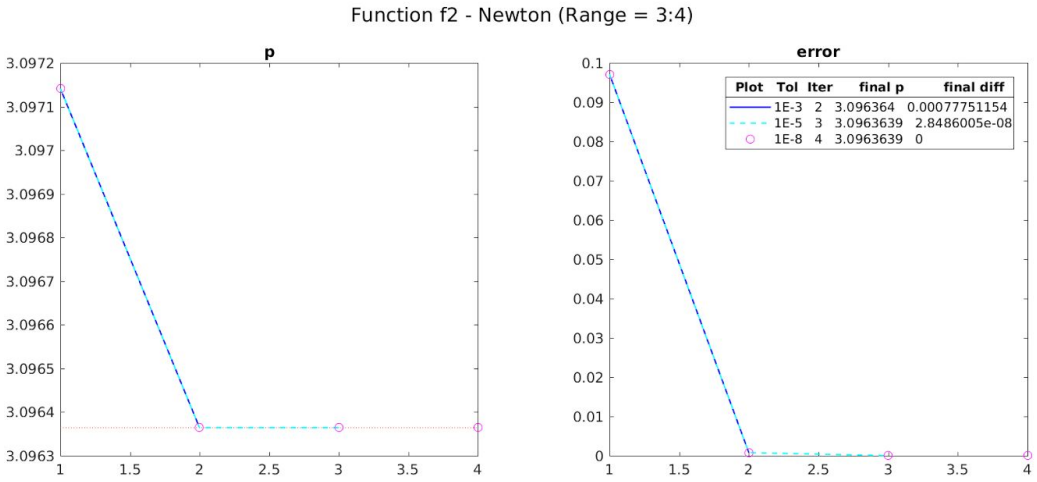

Figure F2-3

Secant Method

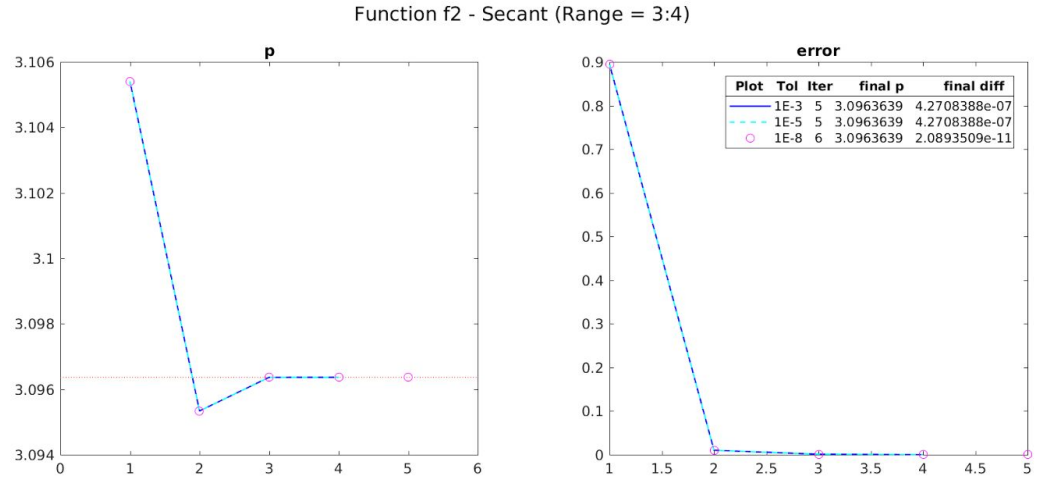

Figure F2-4

Summary of Results

Table F2-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	3	4	10	3.0966797	0.0009765625
bisection	0.00001	3	4	17	3.0963669	7.63E-06
bisection	0.00000001	3	4	27	3.0963639	7.45E-09
newton	0.001	3	4	2	3.096364	0.0007775115

newton	0.00001	3	4	3	3.0963639	2.85E-08
newton	0.00000001	3	4	4	3.0963639	0
secant	0.001	3	4	5	3.0963639	4.27E-07
secant	0.00001	3	4	5	3.0963639	4.27E-07
secant	0.00000001	3	4	6	3.0963639	2.09E-11

F3) Range *6 : 7*

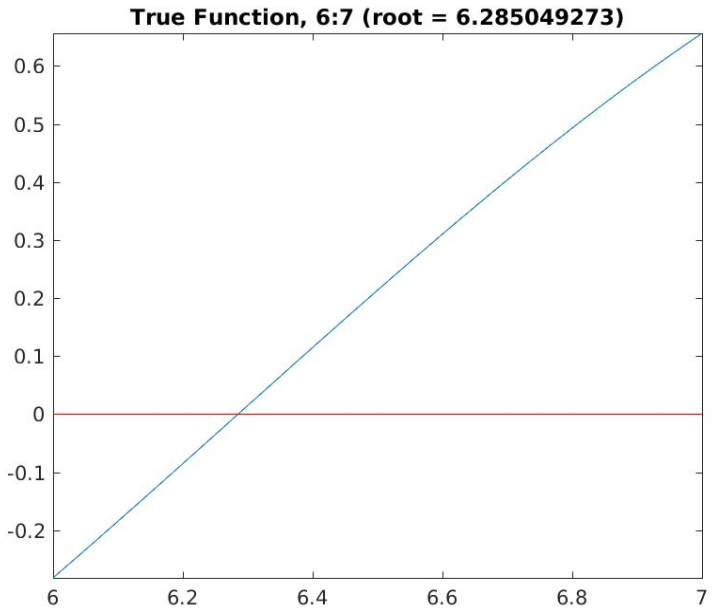

Figure F3-1

Bisection Method

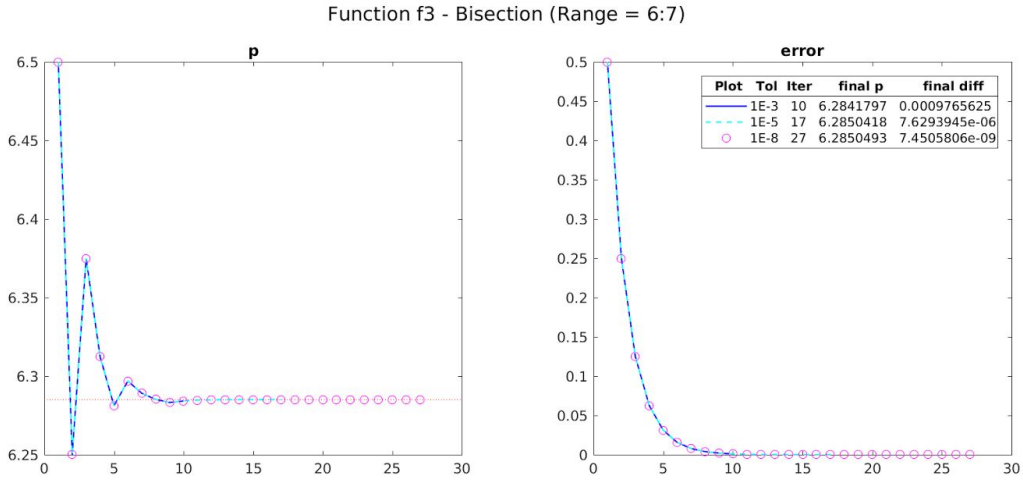

Figure F3-2

Newton's Method

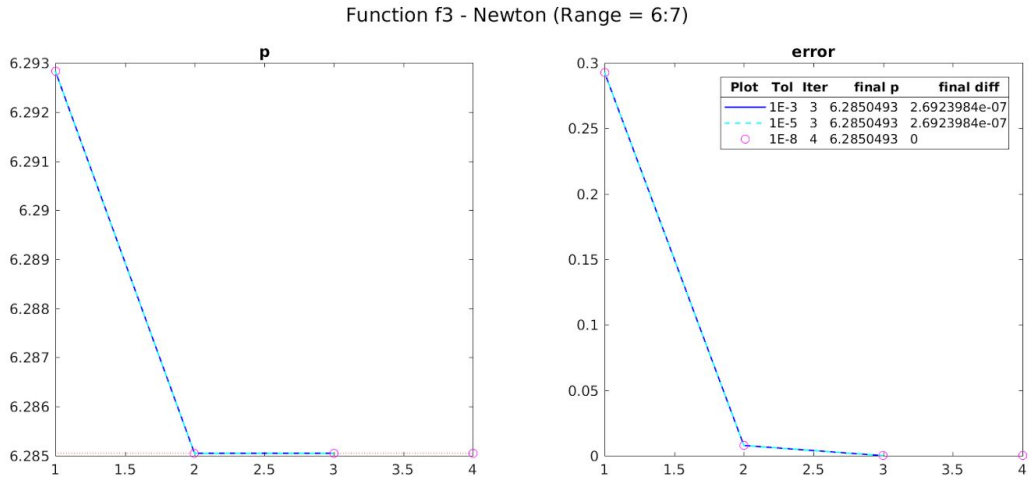

Figure F3-3

Secant Method

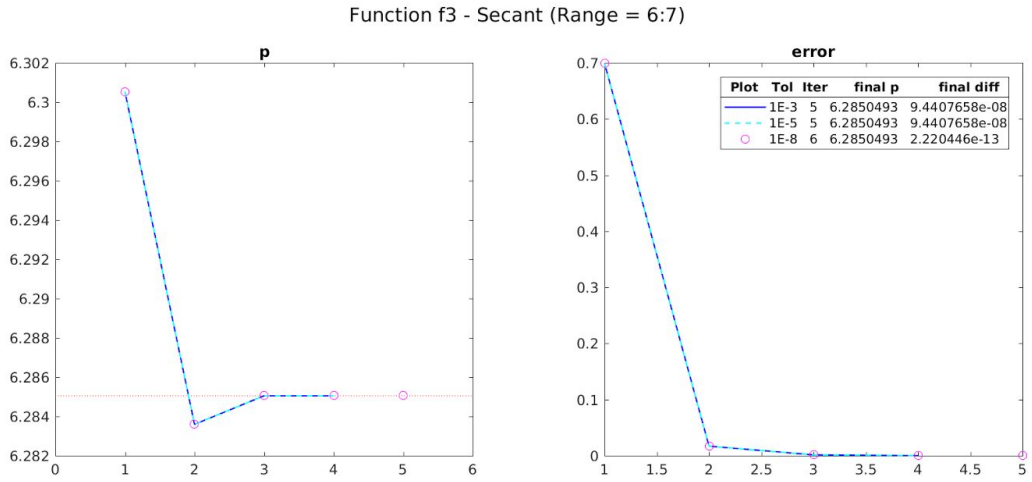

Figure F3-4

Summary of Results

Table F3-1

Method	Tolerance	rmin	rmax	N	p	Max_Error
bisection	0.001	6	7	10	6.2841797	0.0009765625
bisection	0.00001	6	7	17	6.2850418	7.63E-06
bisection	0.00000001	6	7	27	6.2850493	7.45E-09
newton	0.001	6	7	3	6.2850493	2.69E-07
newton	0.00001	6	7	3	6.2850493	2.69E-07

newton	0.00000001	6	7	4	6.2850493	0
secant	0.001	6	7	5	6.2850493	9.44E-08
secant	0.00001	6	7	5	6.2850493	9.44E-08
secant	0.00000001	6	7	6	6.2850493	2.22E-13

Appendix

Manual derivations

A) $f(x) = e^x + 2^{-x} + 2\cos(x) - 6 = 0$

$2^{-x} \rightarrow 1/2^x$

$$\frac{0 - \log(2) \cdot 2^x}{(2^x)^2}$$

$$f'(x) = e^x - 2^{-x} \cdot \log(2) - 2\sin(x)$$

B) $f(x) = \log(x-1) + \cos(x-1)$

$$f'(x) = 1/(x-1) - \sin(x-1)$$

C) $f(x) = 2x \cdot \cos(2x) - (x-2)^2$

$$f'(x) = 2\cos(2x) - 2x\sin(2x)^{\cdot 2} - 2(x-2)$$

$-4x\sin(2x)$

D) $f(x) = (x-2)^2 - \log(x)$

$$f'(x) = 2(x-2) - 1/x$$

E) $f(x) = e^x - 3x^2$

$$f'(x) = e^x - 6x$$

F) $f(x) = \sin(x) - e^{-x}$

$$f'(x) = \cos(x) + e^{-x}$$