

Task 3A - Joseph Britton 2104278

Original Method

Using Original Method:

Elapsed Time: 5.599561 seconds

Total Iterations: 26754

Bisection Method

Using Bisection Method:

Elapsed Time: 1.551018 seconds

Total Iterations: 989898

Newton Method

Using Newton-Raphson Method:

Elapsed Time: 2.969480 seconds

Total Iterations: 53508

Secant Method

Using Secant Method:

Elapsed Time: 2.673030 seconds

Total Iterations: 317812

Results Table

Method Comparison Results:

Method	ElapsedTime	TotalIterations	AverageTolerance
{'Original' }	5.5996	26754	0.024549
{'Bisection' }	1.551	9.899e+05	0.017689
{'Newton-Raphson' }	2.9695	53508	4.0774e-08
{'Secant' }	2.673	3.1781e+05	5.7017e-06

Scoring system (0-5)

- 0: Bad
- 1-2: Poor
- 3: Okay
- 4-5: Good

Table 1 - Root finding methods scored

Solving the function $f(x) = e^x - (1 + x + \frac{x^2}{2})$ with an initial guess of 1, and limits of -2 to 1 for secant and

Solver	Ease-of-use	Efficiency	Accuracy	Total score	Solve method	Comments	Reference
Fzero	5	3	4	12/15	Brent's/Bisection	MATLAB function	(MathWorks, 2025) (Brent, 1973)
Fsolve	4	4	0	8/15	Trust-region-dogleg and trust-region-reflective	MATLAB function (requires optimization toolbox)	(MathWorks, 2025) (Yuan, 2000)
VPAolve	3	3	5	11/15	Variable-precision arithmetic	MATLAB function (requires symbolic math add in)	(MathWorks, 2025) (Schulte, 1994)
lsqnonlin	3	3	0	6/15	Trust-Region-Reflective and Levenberg-Marquardt	MATLAB function (requires optimization toolbox)	(MathWorks, 2025) (Berghen, 2004)
Bisection	3	4	4	11/15	Bisection	Custom Implementation	(Burden, 1985)
Newton-Raphson	3	4	3	10/15	Newton-Raphson	Custom Implementation	(Cajori, 1911)
Secant	3	4	4	11/15	Secant	Custom Implementation	(Papakonstantinou, 2013)

bisection methods. For secant, bisection and Newton-Raphson methods, a tolerance of $1e^{-6}$ was specified.

From the scoring system generated the best performer is shown to be Fzero, hence this technique will be implemented to solve for n5. The analytical solution for this function is 0. Figure 1 shows 5 runs of the code, illustrating a reduction in solve time each run. From the scoring seen in table 1, the chosen method of root finding is Fzero.

Method	Root	Iterations	Time (s)	Percentage Difference (%)
{'fsolve' }	0.099236	14	0.71574	9.9236
{'vpasolve' }	8.6983e-15	NaN	2.9647	8.6983e-13
{'lsqnonlin'}	0.099236	14	0.73451	9.9236
{'fzero' }	-2.4342e-06	73	0.054262	0.00024342
{'Newton' }	1.0197e-05	30	0.050145	0.0010197
{'Bisection'}	8.8215e-06	22	0.10208	0.00088215
{'Secant' }	4.3042e-06	41	0.035721	0.00043042

1

Method	Root	Iterations	Time (s)	Percentage Difference (%)
{'fsolve' }	0.099236	14	0.052339	9.9236
{'vpasolve' }	8.6983e-15	NaN	0.22362	8.6983e-13
{'lsqnonlin'}	0.099236	14	0.053094	9.9236
{'fzero' }	-2.4342e-06	73	0.016254	0.00024342
{'Newton' }	1.0197e-05	30	0.02412	0.0010197
{'Bisection'}	8.8215e-06	22	0.011798	0.00088215
{'Secant' }	4.3042e-06	41	0.070002	0.00043042

2

Method	Root	Iterations	Time (s)	Percentage Difference (%)
{'fsolve' }	0.099236	14	0.049092	9.9236
{'vpasolve' }	8.6983e-15	NaN	0.20394	8.6983e-13
{'lsqnonlin'}	0.099236	14	0.02648	9.9236
{'fzero' }	-2.4342e-06	73	0.002297	0.00024342
{'Newton' }	1.0197e-05	30	0.018028	0.0010197
{'Bisection'}	8.8215e-06	22	0.0048741	0.00088215
{'Secant' }	4.3042e-06	41	0.0094491	0.00043042

3

Method	Root	Iterations	Time (s)	Percentage Difference (%)
{'fsolve' }	0.099236	14	0.057838	9.9236
{'vpasolve' }	8.6983e-15	NaN	0.24603	8.6983e-13
{'lsqnonlin'}	0.099236	14	0.11152	9.9236
{'fzero' }	-2.4342e-06	73	0.010105	0.00024342
{'Newton' }	1.0197e-05	30	0.041253	0.0010197
{'Bisection'}	8.8215e-06	22	0.030869	0.00088215
{'Secant' }	4.3042e-06	41	0.013544	0.00043042

4

Method	Root	Iterations	Time (s)	Percentage Difference (%)
{'fsolve' }	0.099236	14	0.0061433	9.9236
{'vpasolve' }	8.6983e-15	NaN	0.0783	8.6983e-13
{'lsqnonlin'}	0.099236	14	0.0091013	9.9236
{'fzero' }	-2.4342e-06	73	0.0005659	0.00024342
{'Newton' }	1.0197e-05	30	0.023484	0.0010197
{'Bisection'}	8.8215e-06	22	0.0050224	0.00088215
{'Secant' }	4.3042e-06	41	0.006906	0.00043042

5

Figure 1 - Tables of found root, iteration count, time to solve and percentage difference for 5 code runs

References

- Berghen, F., 2004. *Levenberg-Marquardt algorithms vs trust region algorithms*, s.l.: s.n.
- Brent, R., 1973. Chapter 4: An algorithm with guaranteed convergence for finding a zero of a function. In: *Algorithms for minimization without derivatives*. s.l.:s.n.
- Burden, R. F. J., 1985. *Numerical analysis*. 3rd ed. s.l.:PWS Publishers.
- Cajori, F., 1911. Historical note on the Newton-Raphson method of approximation. *The American mathematical monthly*, 18(2), pp. 29-32.
- MathWorks, 2025. *Help Center (Documentation)*. [Online]
Available at: <https://uk.mathworks.com/help/>
[Accessed 16 01 2025].
- Papakonstantinou, J. T. R., 2013. Origin and evolution of the secant method in one dimension. *The American mathematical monthly*, 120(6), pp. 500-518.
- Schulte, M. S. E., 1994. *A variable-precision interval arithmetic processor*. San Francisco, IEEE.
- Yuan, Y., 2000. *A review of trust region algorithms for optimization*, s.l.: s.n.