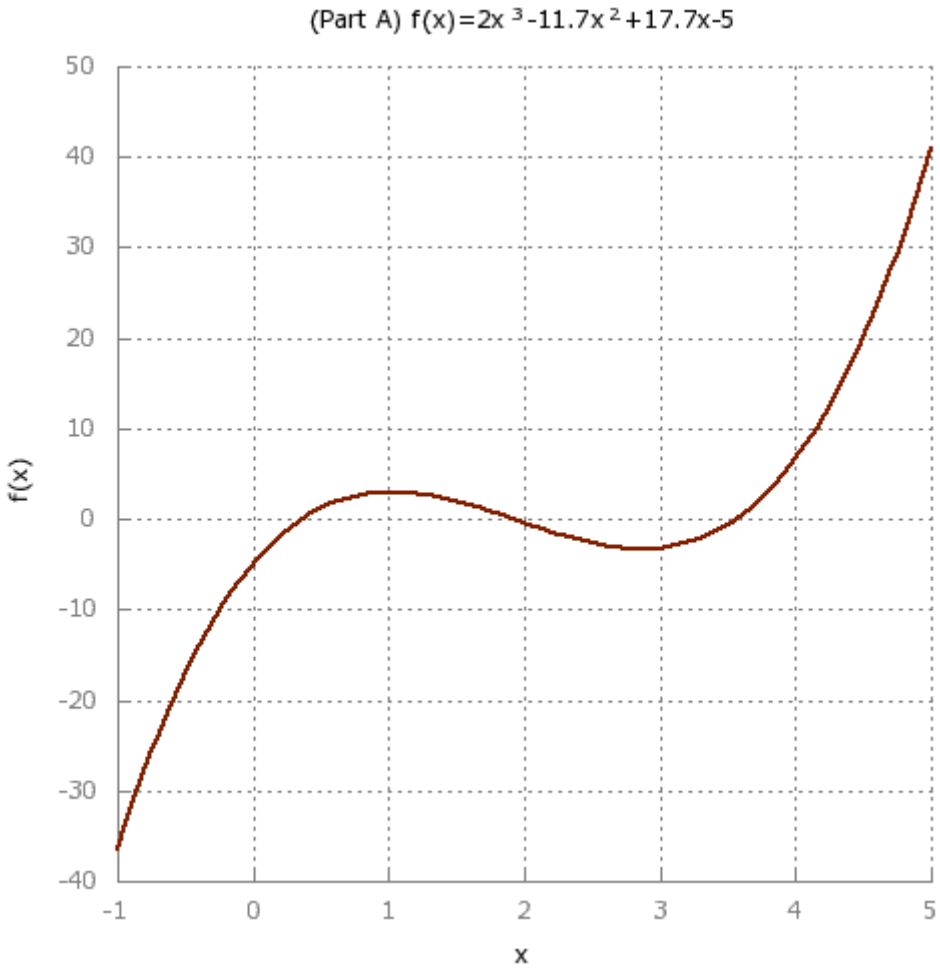
Rachel Chiang

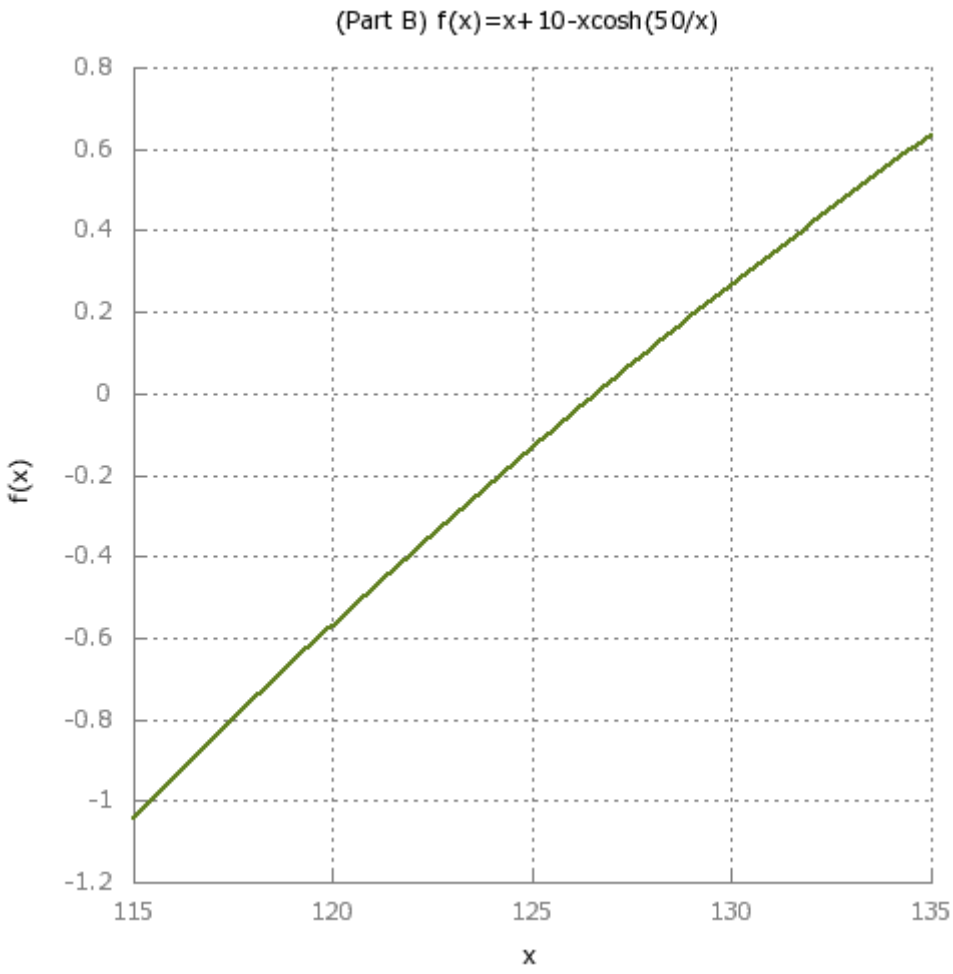
Project 2: Finding Roots

CS 301.01 (F17)

Project 2: Finding Roots of Non-Linear Equations Using Five Numerical Methods

This project implements five methods for locating roots: Bisection, Newton-Raphson, Secant, False-Position, and Modified Secant. Two functions are tested on each of these methods: (a) $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$, with three roots in the interval [0,4], and (b) $f(x) = x + 10 - xcosh(\frac{50}{x})$, with one root in the interval [120,130]. For the former function, the true roots were as follows: 0.365098, 1.92174, and 3.56316, and for the latter, 126.632436. In all of the methods, ε_a was set as 0.01%. The two functions are graphed below.





I. Bisection Method

The bisection method is a very straightforward way to find the root and will always locate a root. However, this method generally takes many iterations to achieve high precision and requires initial values that bracket some root—that is, the two initial values, a and b, must yield functional values of opposite signs. At each iteration, this method calculates the subsequent approximation for the root, c, by slowly converging through the midpoints of a and b:

$$c = \frac{a + b}{2}$$

The approximation c will replace one of the two variables, a or b, depending on which one has the same sign, which effectively ensures that the root is being bracketed at each iteration.

Function A: Roots for $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

Directly below are the iterations and their approximations for locating the three roots for the function of Part A. Included in the tables are the iteration number n, the current value x, its functional value $f(x)$, the percentage relative error ($\%rErr$), and the percentage true error ($\%tErr$).

Steps for approximating the root, $x \cong 0.365092$				
n	x	fx	%rErr	%tErr
0	0.666667	2.19259	-	-
1	0.333333	-0.32593	100	8.70031
2	0.5	1.175	33.3333	36.9495
3	0.416667	0.488426	20	14.1246
4	0.375	0.097656	11.1111	2.71215
5	0.354167	-0.10998	5.88235	2.99408
6	0.364583	-0.00513	2.85714	0.140967
7	0.369792	0.046521	1.40845	1.28559
8	0.367188	0.02076	0.70922	0.572312
9	0.365885	0.007832	0.355872	0.215673
10	0.365234	0.001355	0.178253	0.037353
11	0.364909	-0.00189	0.089206	0.051807
12	0.365072	-0.00027	0.044583	0.007227
13	0.365153	0.000545	0.022287	0.015063
14	0.365112	0.00014	0.011145	0.003918
15	0.365092	-6.26E-05	0.005573	0.001654

(Above) Table for approximating the root, beginning with $a = 0$ and $b = 1.33333$.

Steps for approximating the root, $x \cong 1.92171$				
n	x	fx	%rErr	%tErr
0	2	-0.4	-	-
1	1.66667	1.25926	20	13.273
2	1.83333	0.449074	9.09091	4.60034
3	1.91667	0.025926	4.34783	0.263997
4	1.95833	-0.18712	2.12766	1.90418
5	1.9375	-0.08057	1.07527	0.82009
6	1.92708	-0.02731	0.540541	0.278047
7	1.92187	-0.00069	0.271003	0.007025
8	1.91927	0.012622	0.135685	0.128486
9	1.92057	0.005969	0.067797	0.060731
10	1.92122	0.002642	0.033887	0.026853

	11	1.92155	0.000978	0.016941	0.009914
	12	1.92171	0.000147	0.00847	0.001445

(Above) Table for approximating the root, beginning with $a = 1.33333$ and $b = 2.66667$.

Steps for approximating the root, $x \cong 3.56348$				
n	x	fx	%rErr	%tErr
0	3.33333	-1.92593	-	-
1	3.66667	1.19259	9.09091	2.90491
2	3.5	-0.625	4.7619	1.77258
3	3.58333	0.215741	2.32558	0.566164
4	3.54167	-0.22121	1.17647	0.60321
5	3.5625	-0.00693	0.584795	0.018523
6	3.57292	0.103347	0.291545	0.273821
7	3.56771	0.047943	0.145985	0.127649
8	3.5651	0.020439	0.073046	0.054563
9	3.5638	0.006736	0.036536	0.01802
10	3.56315	-0.0001	0.018272	0.000251
11	3.56348	0.003316	0.009135	0.008884

(Above) Table for approximating the root, beginning with $a = 2.66667$ and $b = 4$.

Function B: Roots for $f(x) = x + 10 - xcosh(\frac{50}{x})$

Steps for approximating the root, $x \cong 126.631$				
n	x	fx	%rErr	%tErr
0	125	-0.13405	-	-
1	127.5	0.06979	1.96078	0.685104
2	126.25	-0.03108	0.990099	0.302005
3	126.875	0.019612	0.492611	0.19155
4	126.563	-0.00567	0.246914	0.055228
5	126.719	0.006988	0.123305	0.068161
6	126.641	0.000663	0.06169	0.006467
7	126.602	-0.0025	0.030855	0.02438
8	126.621	-0.00092	0.015425	0.008957
9	126.631	-0.00013	0.007712	0.001245

(Above) Table for approximating the root, beginning with $a = 120$ and $b = 130$.

II. Newton-Raphson Method

The Newton-Raphson method is an open method. It only needs one initial value and does not need to bracket the root (not that it could with only one value). This method uses tangents to find the roots and is derived from the Taylor's theorem. Each subsequent approximation is calculated as follows:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Notably, utilization of this method requires that the derivative of the function can be found and the slope must never reach zero. Another disadvantage is that the Newton's method may not find an x-intercept even if it exists. Despite these caveats, Newton's method is widely used because it both can find roots that are close to the true values and can calculate the roots very quickly.

Function A: Roots for $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

Steps for approximating the root, $x \cong 0.365098$				
n	x	fx	%rErr	%tErr
0	0	-5	-	-
1	0.282486	-0.88856	100	22.6274
2	0.359293	-0.05812	21.3774	1.5899
3	0.365066	-0.00032	1.58136	0.008673
4	0.365098	-9.68E-09	0.008739	6.64E-05

(Above) Table for approximating the root, beginning with $x = 0$.

Steps for approximating the root, $x \cong 1.92174$				
n	x	fx	%rErr	%tErr
0	1.33333	2.54074	-	-
1	2.23007	-1.5331	40.211	16.0441
2	1.89997	0.111165	17.3739	1.13295
3	1.92176	-0.00012	1.13423	0.00129
4	1.92174	-9.65E-11	0.001241	4.85E-05

(Above) Table for approximating the root, beginning with $x = 1.33333$.

Steps for approximating the root, $x \cong 3.56316$				
n	x	fx	%rErr	%tErr
0	2.66667	-3.07407	-	-

1	1.15483	2.91723	130.915	67.5898
2	3.36285	-1.73074	65.6592	5.62178
3	3.61507	0.571394	6.9771	1.45698
4	3.56547	0.024335	1.39114	0.064936
5	3.56317	5.16E-05	0.064775	0.000161
6	3.56316	2.34E-10	0.000138	2.32E-05

(Above) Table for approximating the root, beginning with $x = 2.66667$.

Function B: Roots for $f(x) = x + 10 - xcosh(\frac{50}{x})$

Steps for approximating the root, $x \cong 126.632$				
n	x	fx	%rErr	%tErr
0	125	-0.13405	-	-
1	126.611	-0.00177	1.27208	0.017257
2	126.632	-3.17E-07	0.017254	3.06E-06
3	126.632	1.07E-14	3.09E-06	3.16E-08

(Above) Table for approximating the root, beginning with $x = 125$.

III. Secant Method

The secant method is another open method and is also very similar to the Newton-Raphson method, as its derivation also draws from the Taylor's theorem. However, unlike the Newton-Raphson method, the secant method requires two points, which is similar to the bisection method—except, as it is an open method, it does not need to bracket a root. The secant method may be useful as a fallback if the function given is not differentiable or is not easily differentiable. In each iteration, the next approximation for a root is calculated as follows:

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

It may also seem that division by zero would be a problem, but this would only occur if $x_i = x_{i-1}$, which would not be calculated unless $f(x_i) = 0$, but that is the solution anyway. This method is slightly similar to the bisection method in another way; it is somewhat choosey on which variable

between x_i and x_{i-1} will be replaced by x_{i+1}, and, however, it is governed by this rule: $|f(x_{i-1})| > |f(x_i)|$.

Function A: Roots for $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

Steps for approximating the root, $x \cong 0.365105$				
n	x	fx	%rErr	%tErr
0	1.33333	2.54074	-	-
1	0	-5	-	-
2	0.884086	2.88553	-	-
3	0.560575	1.59784	57.7105	0.535411
4	0.424818	0.561106	31.9567	0.163571
5	0.381954	0.165134	11.2221	0.046169
6	0.369743	0.046039	3.30268	0.012722
7	0.366369	0.012641	0.920771	0.003482
8	0.365445	0.003456	0.252814	0.000952
9	0.365193	0.000944	0.069123	0.00026
10	0.365124	0.000258	0.018877	7.16E-05
11	0.365105	7.03E-05	0.005154	2.00E-05

(Above) Table for approximating the root, beginning with $x_0 = 1.33333$ and $x_1 = 0$.

Steps for approximating the root, $x \cong 1.92173$				
n	x	fx	%rErr	%tErr
0	1.33333	2.54074	-	-
1	2.66667	-3.07407	-	-
2	1.93668	-0.07635	-	-
3	1.91808	0.01869	0.969317	0.001903
4	1.92261	-0.00443	0.23529	0.000451
5	1.92153	0.001056	0.055822	0.000107
6	1.92179	-0.00025	0.013319	2.61E-05
7	1.92173	6.00E-05	0.003173	5.62E-06

(Above) Table for approximating the root, beginning with $x_0 = 1.33333$ and $x_1 = 2.66667$.

Steps for approximating the root, $x \cong 3.56311$				
n	x	fx	%rErr	%tErr
0	2.66667	-3.07407	-	-
1	4	6.6	-	-
2	3.09035	-3.01157	-	-
3	3.37537	-1.64348	8.44404	0.052703
4	3.4999	-0.62593	3.55812	0.017754

5	3.54322	-0.20552	1.22261	0.005596
6	3.55701	-0.06416	0.387803	0.001725
7	3.56128	-0.01972	0.119766	0.000528
8	3.56259	-0.00603	0.036678	0.000161
9	3.56299	-0.00184	0.011204	4.90E-05
10	3.56311	-0.00056	0.00342	1.48E-05

(Above) Table for approximating the root, beginning with $x_0 = 2.66667$ and $x_1 = 4$.

Function B: Roots for $f(x) = x + 10 - x\cosh(\frac{50}{x})$

Steps for approximating the root, $x \cong 126.633$				
n	x	fx	%rErr	%tErr
0	130	0.265497	-	-
1	120	-0.56825	-	-
2	126.816	0.014817	-	-
3	126.642	0.000807	0.136765	7.87E-05
4	126.633	4.39E-05	0.007441	4.28E-06

(Above) Table for approximating the root, beginning with $x_0 = 130$ and $x_1 = 120$.

IV. False-Position Method

The false-position method, also known as the regula-falsi method, is a bracketing method that mixes components of the bisection method and the secant method. It is similar to the bisection method in that it requires two initial values, a and b, which bracket a root, which means that it will guarantee returning a root. However, unlike the bisection method, it gains speed by using a more intelligent approximation calculation method, using the same slope approximation method as the secant method. The root approximations c are calculated from the previous values b and a as follows:

$$c = b - f(b)\frac{b - a}{f(b) - f(a)}$$

Since the two values used to calculate each approximation, a and b, must bracket a root, similarly to the bisection method, c will select which variable to replace by picking the one that shares the same sign.

Function A: Roots for $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

Steps for approximating the root, $x \cong 0.365105$				
n	x	fx	%rErr	%tErr
0	0.884086	2.88553	-	-
1	0.560575	1.59784	57.7105	53.5411
2	0.424818	0.561106	31.9567	16.3571
3	0.381954	0.165134	11.2221	4.6169
4	0.369743	0.046039	3.30268	1.27221
5	0.366369	0.012641	0.920771	0.34823
6	0.365445	0.003456	0.252814	0.095176
7	0.365193	0.000944	0.069123	0.026035
8	0.365124	0.000258	0.018877	0.007156
9	0.365105	7.03E-05	0.005154	0.002002

(Above) Table for approximating the root, beginning with $a = 0$ and $b = 1.33333$.

Steps for approximating the root, $x \cong 1.92174$				
n	x	fx	%rErr	%tErr
0	1.93668	-0.07635	-	-
1	1.91907	0.013628	0.917193	0.138736
2	1.92174	5.77E-06	0.138726	1.03E-05
3	1.92174	2.36E-09	5.88E-05	4.85E-05

(Above) Table for approximating the root, beginning with $a = 1.33333$ and $b = 2.66667$.

Steps for approximating the root, $x \cong 3.56311$				
n	x	fx	%rErr	%tErr
0	3.09035	-3.01157	-	-
1	3.37537	-1.64348	8.44404	5.27033
2	3.4999	-0.62593	3.55812	1.77538
3	3.54322	-0.20552	1.22261	0.55961
4	3.55701	-0.06416	0.387803	0.172476
5	3.56128	-0.01972	0.119766	0.052774
6	3.56259	-0.00603	0.036678	0.016101
7	3.56299	-0.00184	0.011204	0.004898
8	3.56311	-0.00056	0.00342	0.001479

(Above) Table for approximating the root, beginning with $a = 2.66667$ and $b = 4$.

Function B: Roots for $f(x) = x + 10 - xcosh(\frac{50}{x})$

Steps for approximating the root, $x \cong 126.633$

n	x	fx	%rErr	%tErr
0	126.816	0.014817	-	-
1	126.642	0.000807	0.136765	0.00787
2	126.633	4.39E-05	0.007441	0.000428

(Above) Table for approximating the root, beginning with $a = 120$ and $b = 130$.

V. Modified Secant Method

The modified secant method is another open method and is, as implied by the name, based off of the secant method. However, it attempts to eliminate the need for using a second initial variable by introducing a new constant, δ. The equation to approximate subsequent values for the x-intercept is given below:

$$x_{i+1} = x_i - f(x_i)\frac{\delta x_i}{f(x_i + \delta x_i) - f(x_i)}$$

The constant that affects changing the x's would likely be a small value, so that the method can explore many different possibilities for x. In our case, we set $\delta = 0.01$.

Function A: Roots for $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

Steps for approximating the root, $x \cong 0.365098$				
n	x	fx	%rErr	%tErr
0	0.333333	-0.32593	-	-
1	0.364273	-0.00823	-	-
2	0.3651	2.21E-05	0.226733	0.000676
3	0.365098	-7.75E-08	0.000611	6.45E-05

(Above) Table for approximating the root, beginning with $x_0 = 0.333333$.

Steps for approximating the root, $x \cong 1.92174$				
n	x	fx	%rErr	%tErr
0	1.66667	1.25926	-	-
1	1.93683	-0.07714	-	-
2	1.92175	-3.66E-05	0.784778	0.000421
3	1.92174	-1.80E-08	0.000373	4.87E-05

(Above) Table for approximating the root, beginning with $x_0 = 1.66667$.

Steps for approximating the root, $x \cong 3.56316$				
n	x	fx	%rErr	%tErr
0	3	-3.2	-	-
1	4.8926	35.7632	-	-
2	4.14295	9.73047	18.0945	16.2718
3	3.74232	2.20306	10.7055	5.02801
4	3.59105	0.300425	4.21217	0.782865
5	3.5647	0.016189	0.739309	0.043237
6	3.56321	0.000539	0.041771	0.001465
7	3.56316	1.73E-05	0.001395	6.94E-05

(Above) Table for approximating the root, beginning with $x_0 = 3$.

Function B: Roots for $f(x) = x + 10 - xcosh(\frac{50}{x})$

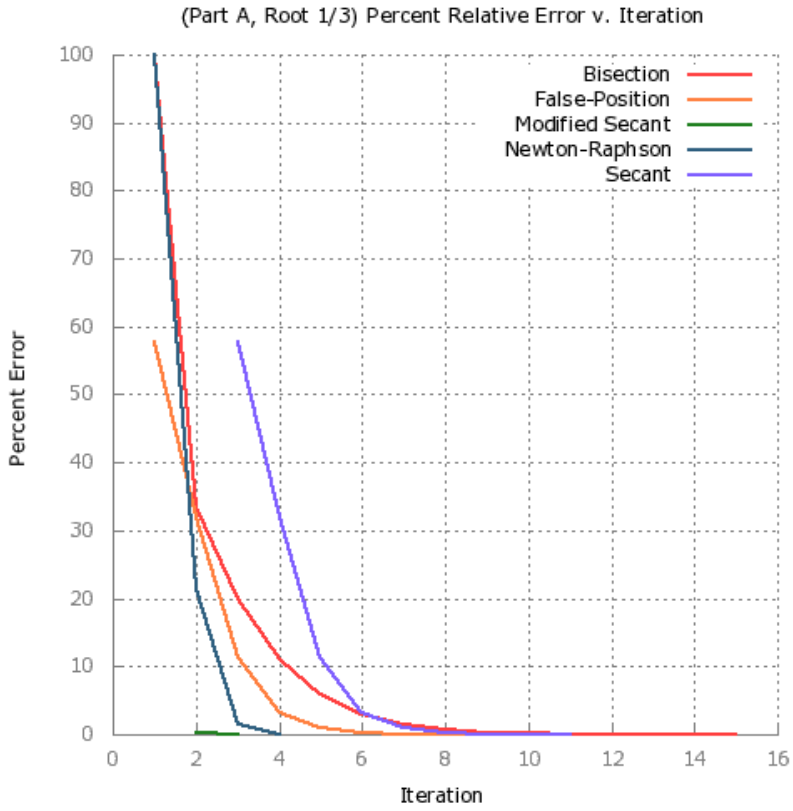
n	x	fx	%rErr	%tErr
0	125	-0.13405	-	-
1	126.627	-0.00041	-	-
2	126.632	4.29E-06	0.004084	4.18E-05

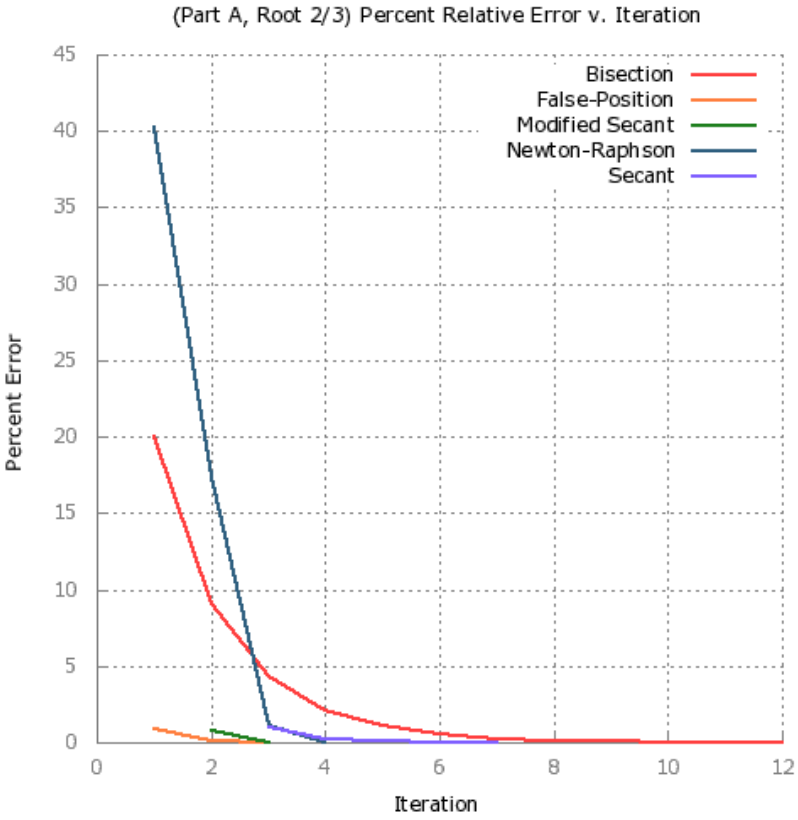
(Above) Table for approximating the root, beginning with $x_0 = 125$.

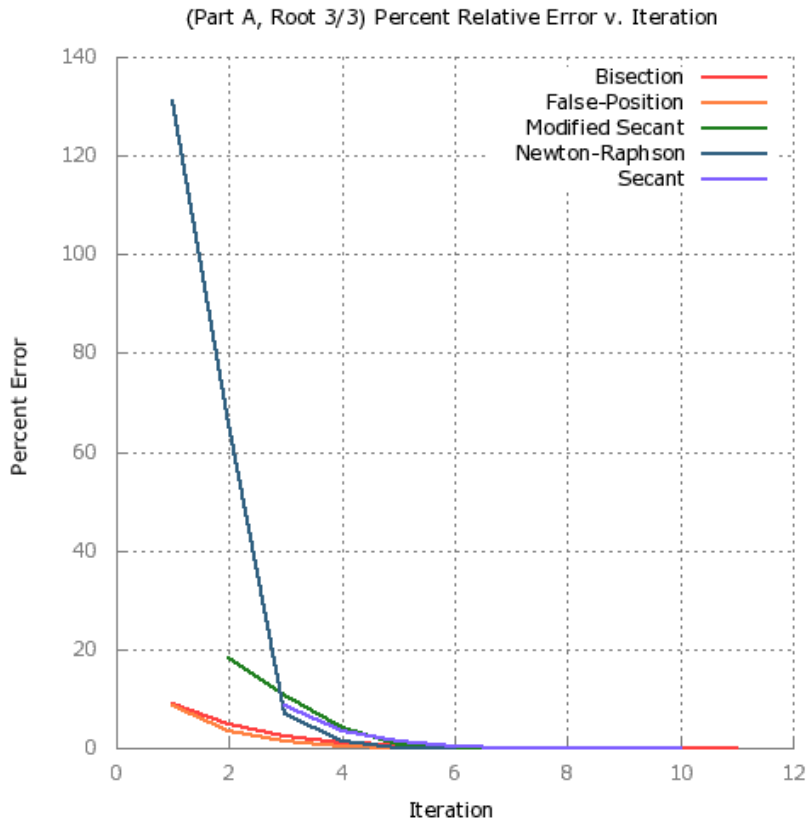
Discussion

Error Comparisons

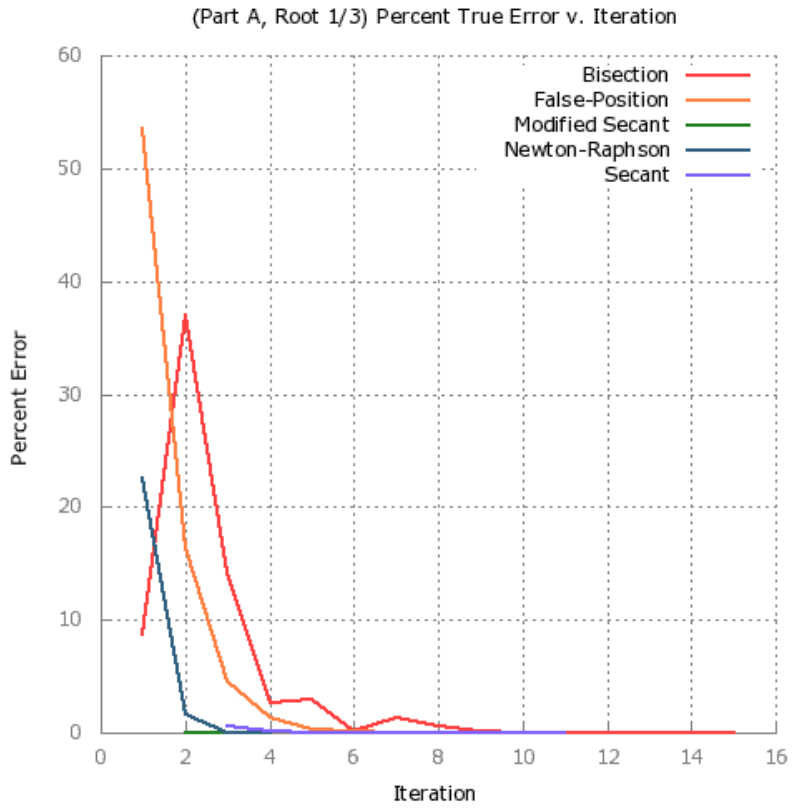
Percent Relative Errors for Part A. For the most part, convergence rates were as expected. Bisection consistently took the most iterations to converge the desired amount, whereas Newton-Raphson, despite sometimes beginning with substantial differences between one approximation to the other, converged incredibly fast. The other three generally had error curves that were similar to one another, which also makes sense, since those three (secant, false position, and modified secant) had nearly identical approximation calculations.

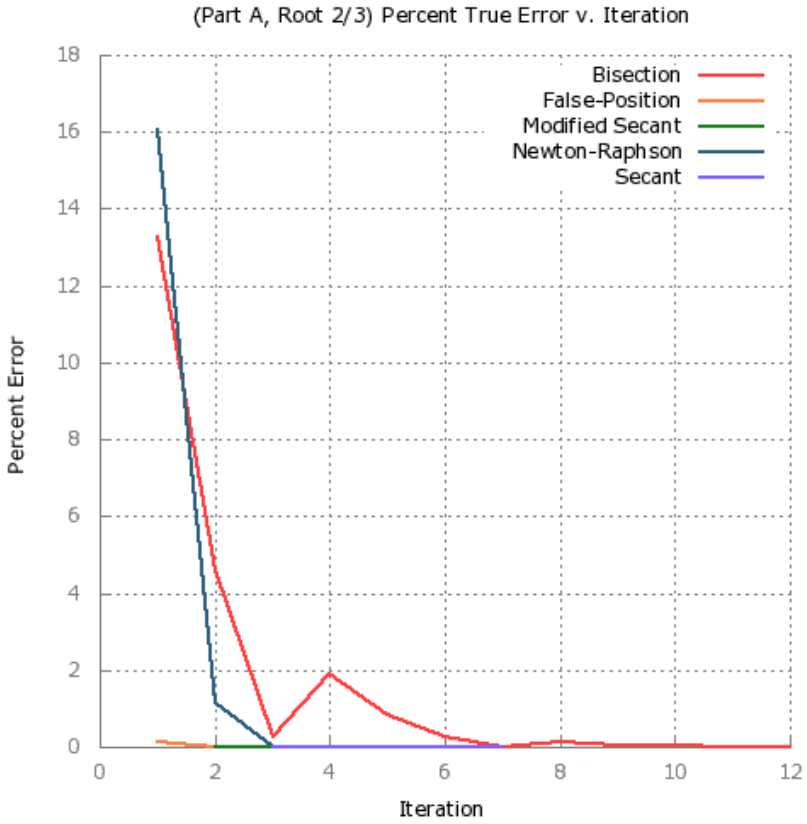

(Part A, Root 1/3) Percent Relative Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
10
20
30
40
50
60
70
80
90
100
0
2
4
6
8
10
12
14
16

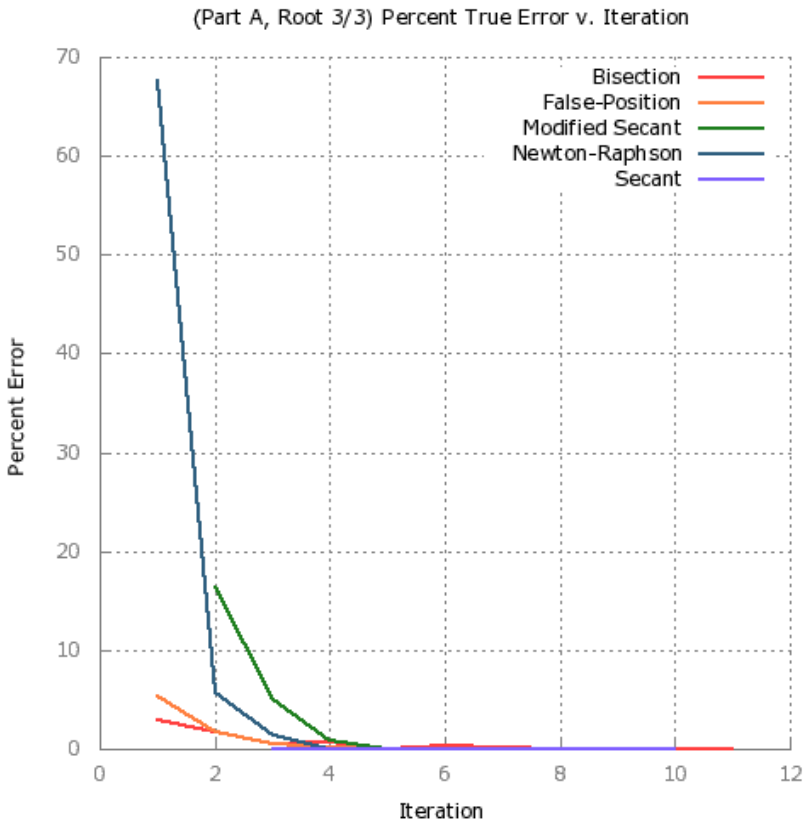

(Part A, Root 2/3) Percent Relative Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
5
10
15
20
25
30
35
40
45
0
2
4
6
8
10
12



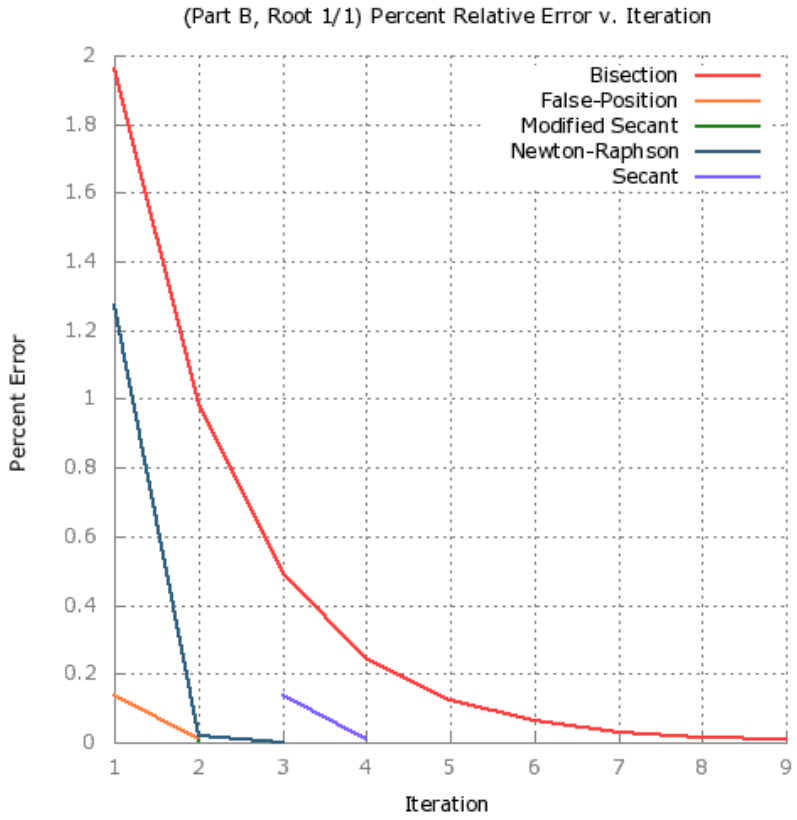
Percent True Errors for Part A. Again, the error curves generally went as expected. As stated previously, one of the downsides of the bisection method is that it is blind to which value between a and b it should replace with c. Evidently seen in the bisection method's error plots below, the bisection method sometimes would snap back and forth between the opposite signs, which would affect its comparison to the true error in a seemingly undesired way, even though in the end, the bisection method still would approximate the roots well enough. Although the false-position method shares with the bisection in its selection of the two values used for subsequent approximations, it is clear that the calculation for the approximation itself has some impact on the convergence patterns and rate; the false-position method's adoption of the slope for approximation gives far more information about where the true x-intercept is compared to simply grabbing midpoints.


(Part A, Root 1/3) Percent True Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
10
20
30
40
50
60
0
2
4
6
8
10
12
14
16

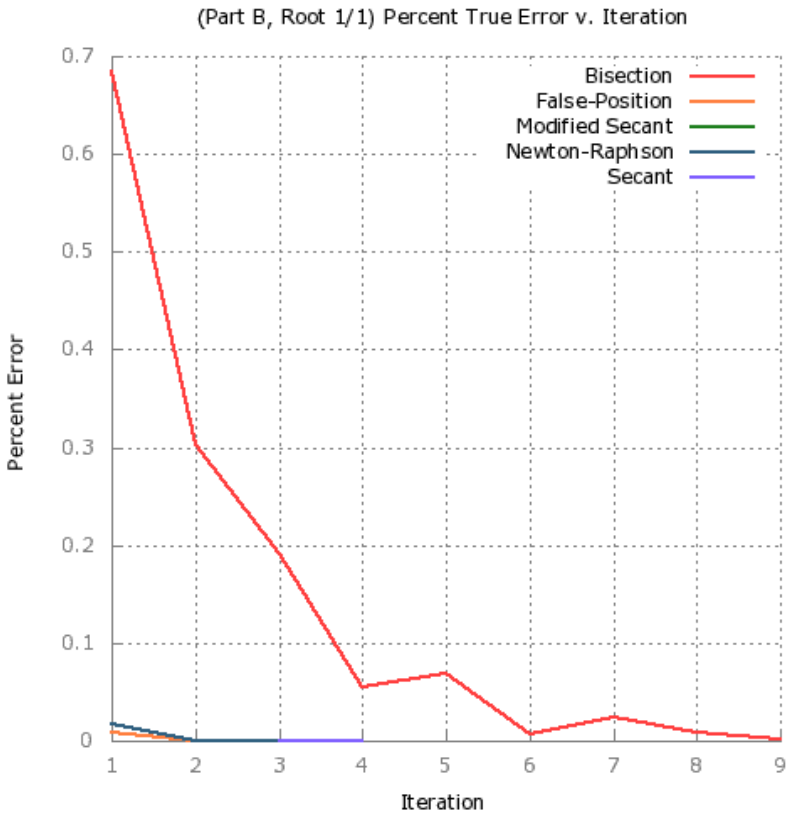

(Part A, Root 2/3) Percent True Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
2
4
6
8
10
12
14
16
18
0
2
4
6
8
10
12



Percent Relative Errors for Part B. The methods tended to converge extremely fast for part B, except for the bisection method. All methods except for the bisection method actually achieved a satisfactory approximation within four iterations.



Percent True Errors for Part B. Again, all of the methods except for the bisection method approximated the root quite well very quickly. And again, the bisection method demonstrates its less-than-optimal next-root approximation.



Implementation Details

Initial Values

Part A. There were three roots for the function $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$ in the interval [0,4]. In fact, there are actually only three roots total, since the function is a third-degree polynomial. Equipped with this knowledge, the initial values were selected by dividing the interval into three parts. In other words, each $piece = \frac{(upper\ bound) - (lower\ bound)}{(number\ of\ roots)}$. In general, the initial values would be set as some result of $(lower\ bound) + (k) * (piece)$, where $k = (iteration)$ or $k = (iteration) + 1$, depending on the method used, since some methods required two initial values or some methods would not work if the initial value was set to 0.

Part B. Since it was known that there was only one root in the function $f(x) = x + 10 - xcosh\left(\frac{50}{x}\right)$, picking initial values for this equation was generally rather easy. For methods that

required two initial values, the known bounds 120 and 130 would be used, and for methods that required only one initial value, I passed in the midpoint. It just so happened that the midpoint also was extremely close to the true root, which made some methods converge fast, which demonstrates that selecting proper initial values is significant.

Practically, however, it may not be known how many roots there are in advance, or perhaps a calculator would want to be as general as possible.

Data Types

Most of the data types I used were simply doubles (for calculations and approximations) or integers (for counting). However, if one wants to calculate with more precision, it would be worth looking into using larger data types. I did not save any prior calculation variables that would not be needed in subsequent iterations.

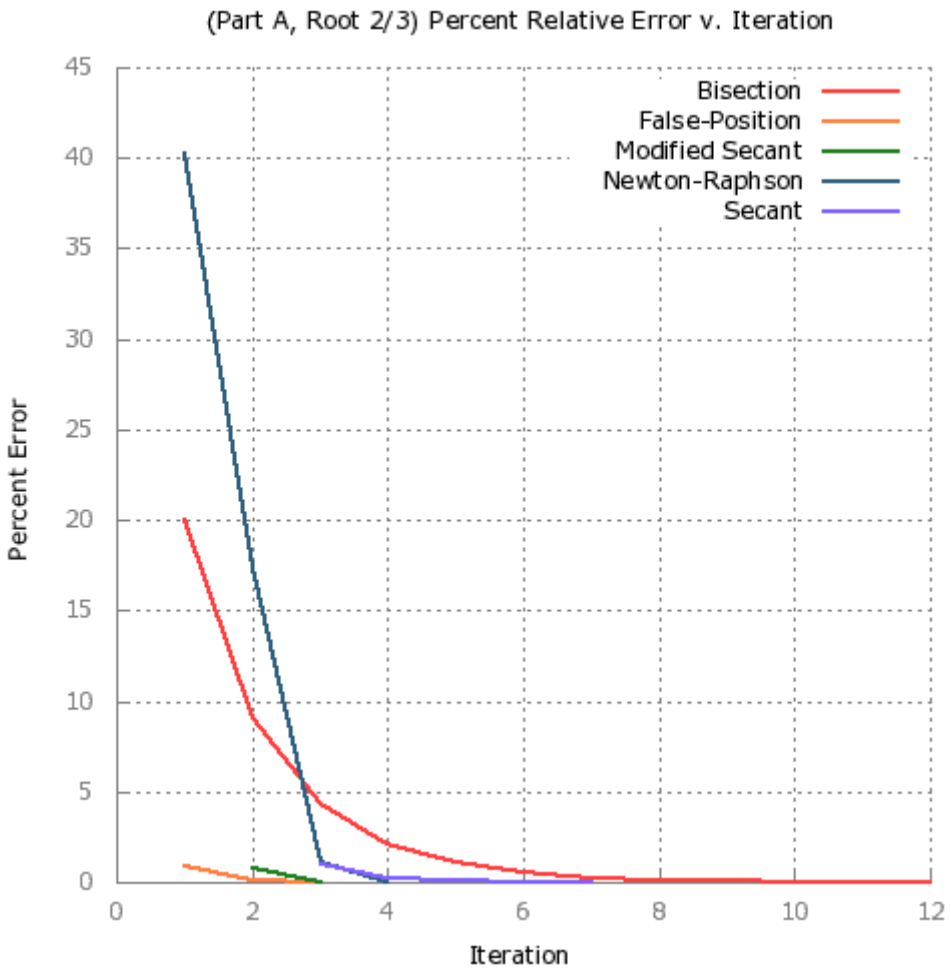
Convergence

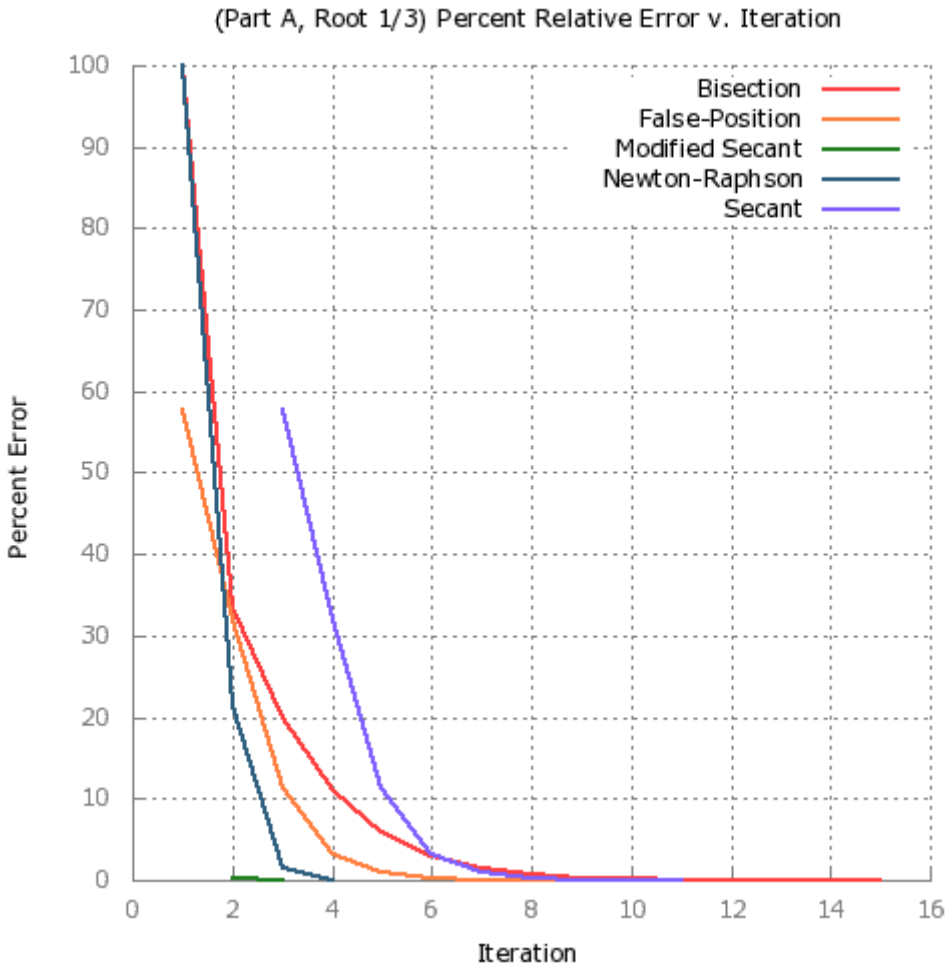
A method was considered to have converged adequately if it reached a relative error percentage less than 0.01%; though the project specifications said less than 1% was sufficient, I wanted to acquire more data points. Interestingly, this did not heavily affect the very fast-converging methods, such as the Newton-Raphson method, but it did affect ones that were slower to converge, such as the bisection method. There is also a maximum of 100 iterations per method.

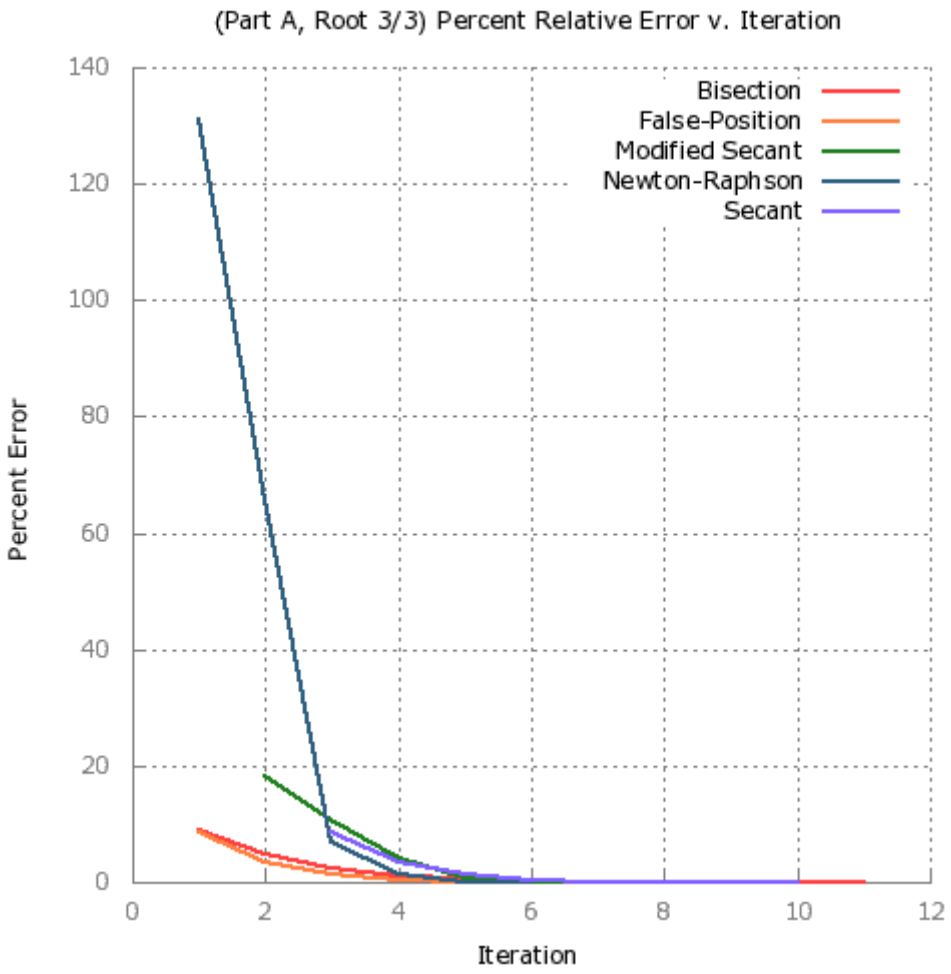
Appendix

This simply contains higher-resolution graphs in case the earlier ones were difficult to see.

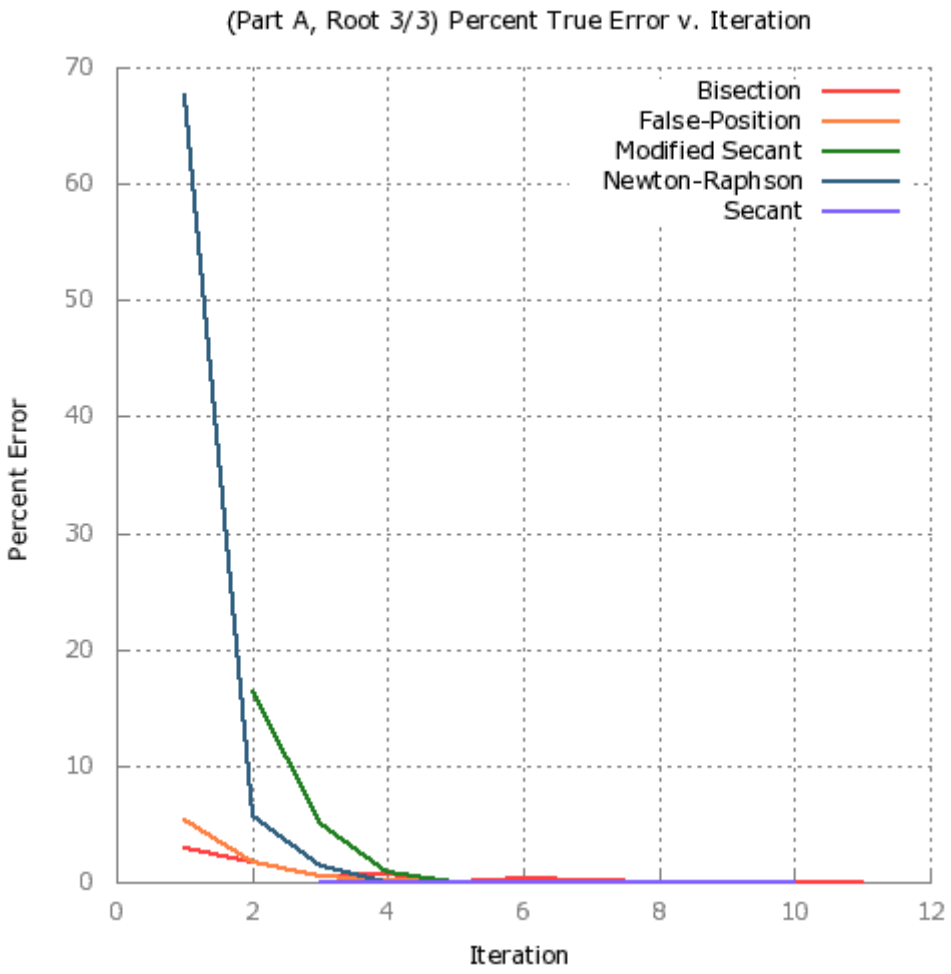
Part A: Relative Error Curves

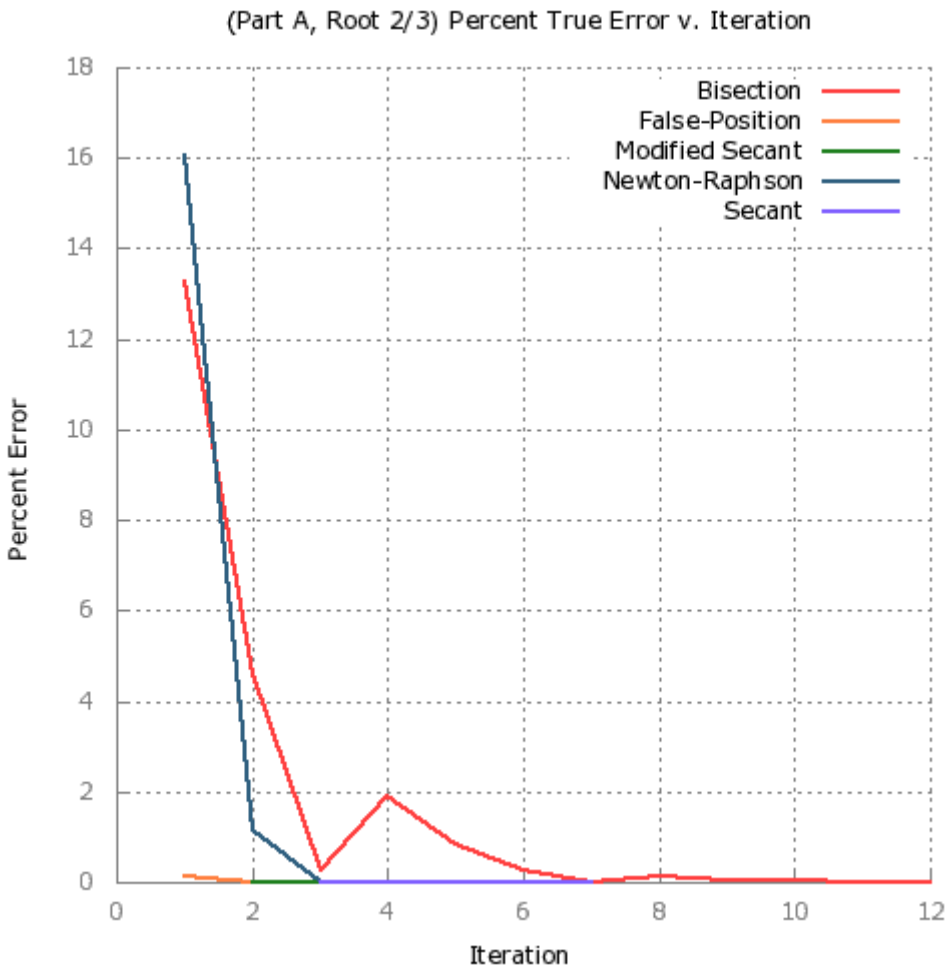

(Part A, Root 2/3) Percent Relative Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
5
10
15
20
25
30
35
40
45
0
2
4
6
8
10
12

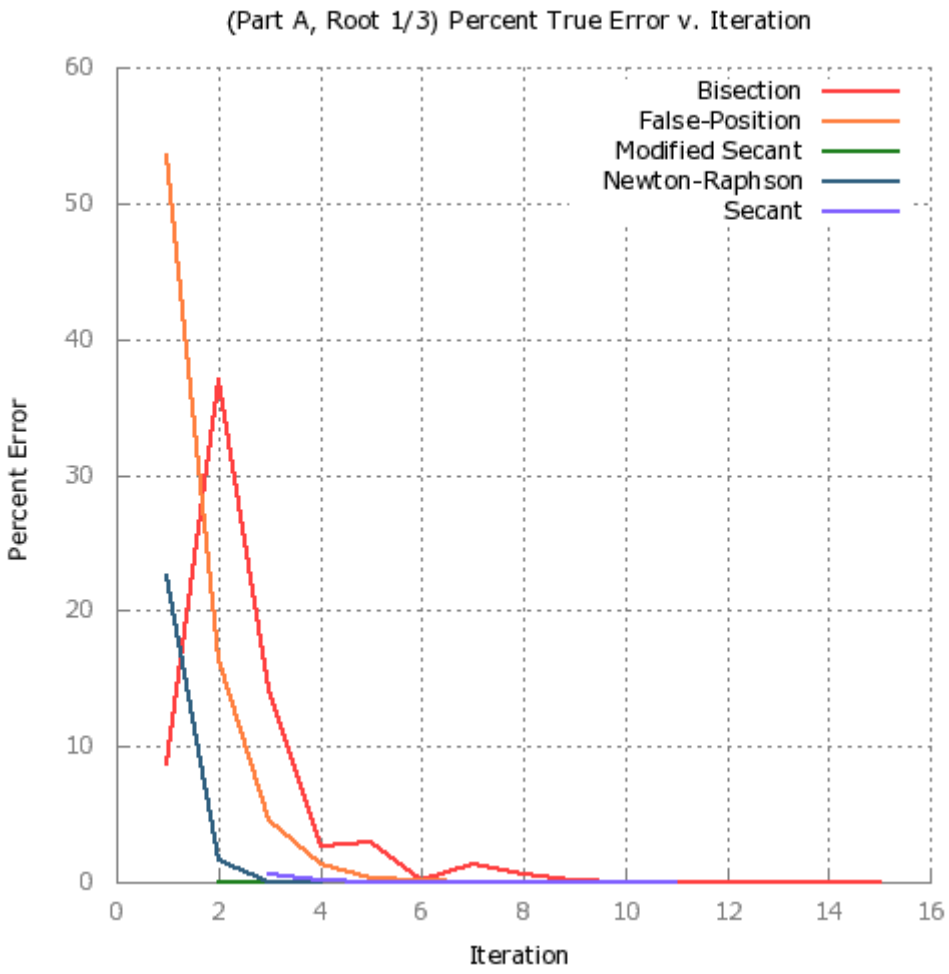

(Part A, Root 1/3) Percent Relative Error v. Iteration
Percent Error
Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
0
10
20
30
40
50
60
70
80
90
100
0
2
4
6
8
10
12
14
16



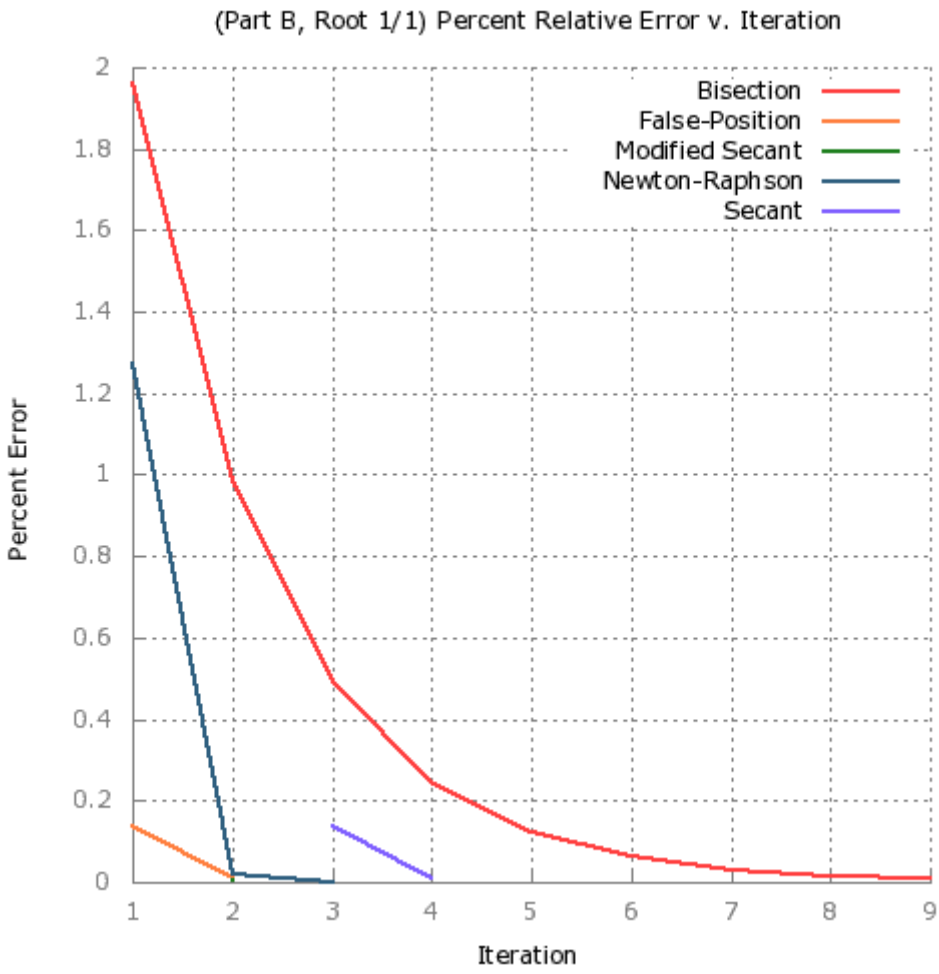
Part A True Error Curves


(Part A, Root 3/3) Percent True Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
10
20
30
40
50
60
70
0
2
4
6
8
10
12

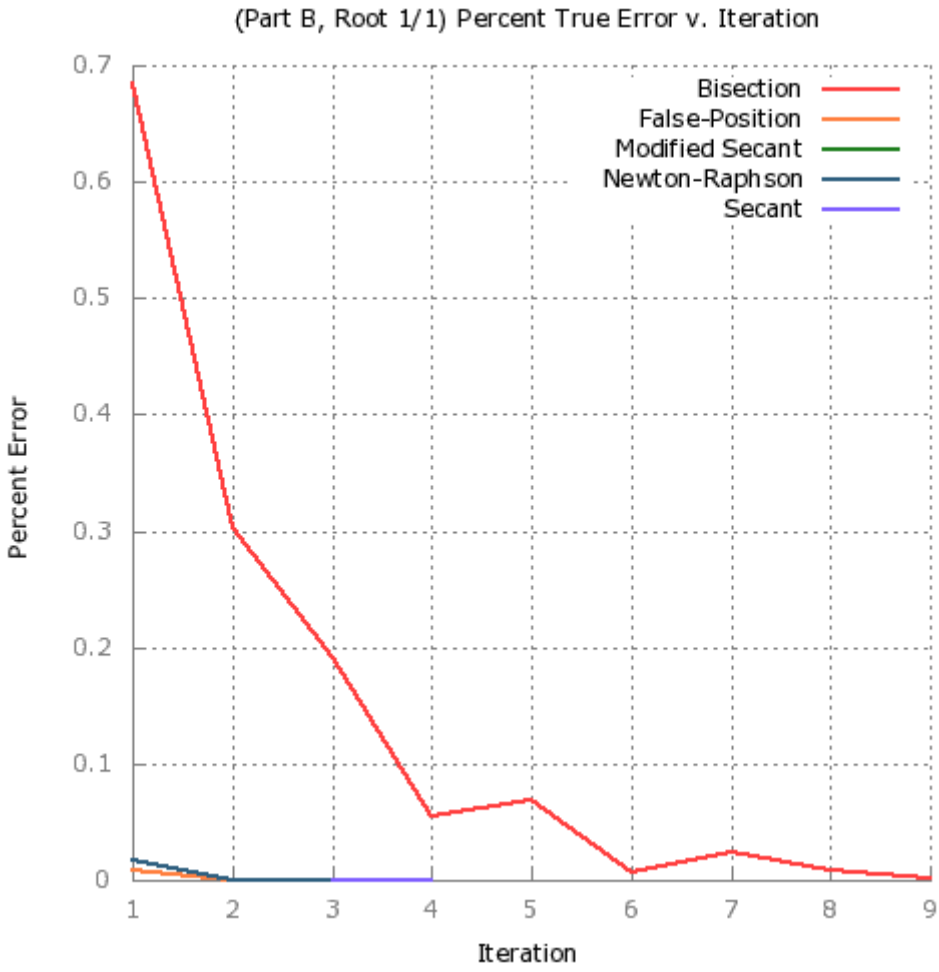

(Part A, Root 2/3) Percent True Error v. Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
Percent Error
Iteration
0
2
4
6
8
10
12
14
16
18



Part B Relative Error Curves



Part B True Error Curves


(Part B, Root 1/1) Percent True Error v. Iteration
Percent Error
Iteration
Bisection
False-Position
Modified Secant
Newton-Raphson
Secant
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9