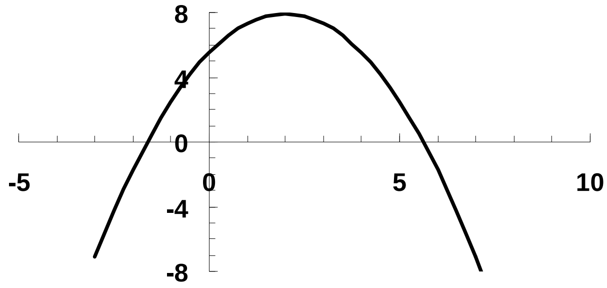
CHAPTER 5

5.1 (a) A plot indicates that roots occur at about $x = -1.6$ and 5.6.



(b)

$$x = \frac{-2.4 \pm \sqrt{(2.4)^2 - 4(-0.6)(5.5)}}{2(-0.6)} = \begin{matrix} -1.62859 \\ \\ 5.62859 \end{matrix}$$

(c) First iteration:

$$x_r = \frac{5+10}{2} = 7.5$$

$$\varepsilon_t = \left|\frac{5.62859 - 7.5}{5.62859}\right| \times 100\% = 33.25\% \qquad \varepsilon_a = \left|\frac{10-5}{10+5}\right| \times 100\% = 33.33\%$$

$$f(5)f(7.5) = 2.5(-10.25) = -25.625$$

Therefore, the bracket is $x_l = 5$ and $x_u = 7.5$.

Second iteration:

$$x_r = \frac{5+7.5}{2} = 6.25$$

$$\varepsilon_t = \left|\frac{5.62859 - 6.25}{5.62859}\right| \times 100\% = 11.04\% \qquad \varepsilon_a = \left|\frac{7.5-5}{7.5+5}\right| \times 100\% = 20.00\%$$

$$f(5)f(6.25) = 2.5(-2.9375) = -7.34375$$

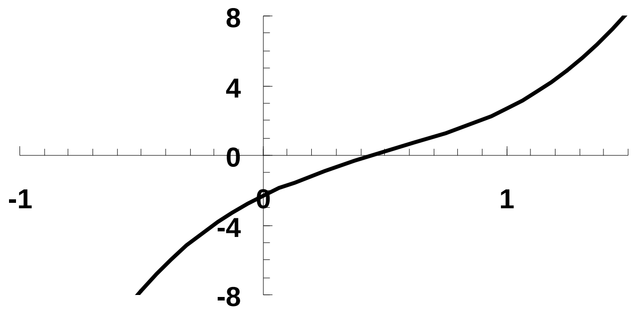
Consequently, the new bracket is $x_l = 5$ and $x_u = 6.25$.

Third iteration:

$$x_r = \frac{5+6.25}{2} = 5.625$$

$$\varepsilon_t = \left|\frac{5.62859 - 5.625}{5.62859}\right| \times 100\% = 0.06\% \qquad \varepsilon_a = \left|\frac{6.25-5}{6.25+5}\right| \times 100\% = 11.11\%$$

5.2 (a) A plot indicates that a single real root occurs at about $x = 0.45$.



(b) First iteration:

$$x_r = \frac{0+1}{2} = 0.5$$

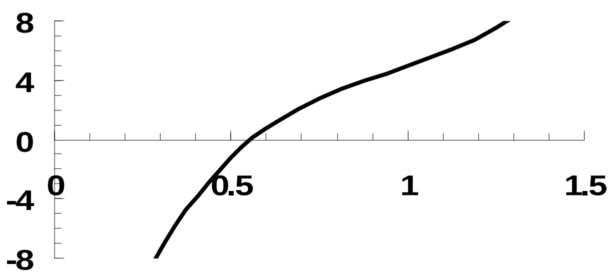
$$\varepsilon_a = \left|\frac{1-0}{1+0}\right| \times 100\% = 100\%$$

$$f(0)\,f(0.5) = -2.3(0.2) = -0.46$$

Therefore, the new bracket is $x_l = 0$ and $x_u = 0.5$. The process can be repeated until the approximate error falls below 10%. As summarized below, this occurs after 5 iterations yielding a root estimate of 0.46875.

iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0	1	0.5	-2.3	0.2	-0.46	100.00%
2	0	0.5	0.25	-2.3	-0.8625	1.98375	100.00%
3	0.25	0.5	0.375	-0.8625	-0.30781	0.265488	33.33%
4	0.375	0.5	0.4375	-0.30781	-0.05098	0.015691	14.29%
5	0.4375	0.5	0.46875	-0.05098	0.074878	-0.00382	6.67%

5.3 (a) A plot indicates that a single real root occurs at about $x = 0.56$.



(b) Bisection:

First iteration:

$$x_r = \frac{0.5+1}{2} = 0.75$$

$$\varepsilon_a = \left|\frac{1-0.5}{1+0.5}\right| \times 100\% = 33.33\%$$

$$f(0.5)\,f(0.75) = -1.21875(2.83105) = -3.45035$$

Therefore, the new bracket is $x_l = 0.5$ and $x_u = 0.75$. The process can be repeated until the approximate error falls below 10%. As summarized below, this occurs after 4 iterations yielding a root estimate of 0.53125.

iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0.50000	1.00000	0.75000	-1.21875	2.83105	-3.45035	33.33%
2	0.50000	0.75000	0.62500	-1.21875	1.19498	-1.45638	20.00%
3	0.50000	0.62500	0.56250	-1.21875	0.10600	-0.12918	11.11%
4	0.50000	0.56250	0.53125	-1.21875	-0.52422	0.63889	5.88%

(c) False position:

First iteration:

$x_l = 0.5$ $\quad f(x_l) = -1.21875$

$x_u = 1$ $\quad f(x_u) = 5$

$$x_r = 1 - \frac{5(0.5-1)}{-1.21875-5} = 0.59799$$

$$f(0.5)\,f(0.59799) = -1.2175(0.75057) = -0.91475$$

Therefore, the bracket is $x_l = 0.5$ and $x_u = 0.59799$.

Second iteration:

$x_l = 0.5$ $\quad f(x_l) = -1.21875$

$x_u = 0.59799$ $\quad f(x_u) = 0.75057$

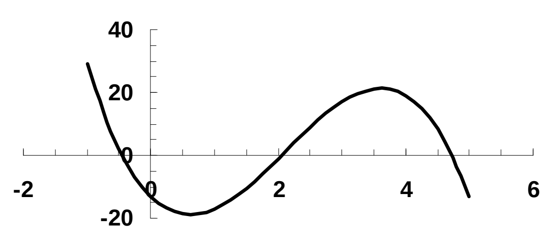
$$x_r = 0.59799 - \frac{0.75057(0.5-0.59799)}{-1.21875-0.75057} = 0.56064$$

$$\varepsilon_a = \left|\frac{0.56064-0.59799}{0.56064}\right| \times 100\% = 6.661\%$$

The process can be repeated until the approximate error falls below 0.2%. As summarized below, this occurs after 4 iterations yielding a root estimate of 0.55705.

iteration	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0.5	1.00000	-1.21875	5.00000	0.59799	0.75057	-0.91475	
2	0.5	0.59799	-1.21875	0.75057	0.56064	0.07024	-0.08561	6.661%
3	0.5	0.56064	-1.21875	0.07024	0.55734	0.00610	-0.00744	0.593%
4	0.5	0.55734	-1.21875	0.00610	0.55705	0.00053	-0.00064	0.051%

5.4 (a) The graph indicates that roots are located at about –0.5, 2 and 4.7.



(b) Using bisection, the first iteration is

$$x_r = \frac{-1+0}{2} = -0.5$$
$$f(-1)\,f(-0.5) = 29(2.125) = 61.625$$

Therefore, the root is in the second interval and the lower guess is redefined as x_l = –0.5. The second iteration is

$$x_r = \frac{-0.5+0}{2} = -0.25$$
$$\varepsilon_a = \left|\frac{-0.25-(-0.5)}{-0.25}\right| 100\% = 100\%$$
$$f(-0.5)\,f(-0.25) = 2.125(-6.76563) = -14.37695$$

Consequently, the root is in the first interval and the upper guess is redefined as x_u = –0.25. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	-1.0	29.00000	0	-13.00000	-0.5	2.12500	100.00%
2	-0.5	2.12500	0	-13.00000	-0.25	-6.76563	100.00%
3	-0.5	2.12500	-0.25	-6.76563	-0.375	-2.66992	33.33%
4	-0.5	2.12500	-0.375	-2.66992	-0.43750	-0.36206	14.29%
5	-0.50000	2.12500	-0.438	-0.36206	-0.46875	0.85880	6.67%
6	-0.46875	0.85880	-0.43750	-0.36206	-0.45313	0.24273	3.45%
7	-0.45313	0.24273	-0.43750	-0.36206	-0.44531	-0.06107	1.75%
8	-0.45313	0.24273	-0.44531	-0.06107	-0.44922	0.09048	0.87%

Thus, after eight iterations, we obtain a root estimate of **–0.44922** with an approximate error of 0.87%, which is below the stopping criterion of 1%.

(c) Using false position, the first iteration is

$$x_r = 0 - \frac{-13(-1-0)}{29-(-13)} = -0.30952$$
$$f(-1)\,f(-0.30952) = 29(-4.90027) = -142.10775$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = –0.30952. The second iteration is

$$x_r = -0.30952 - \frac{-4.90027(-1-(-0.30952))}{29-(-4.90027)} = -0.40933$$
$$\varepsilon_a = \left|\frac{-0.40933-(-0.30952)}{-0.40933}\right| 100\% = 24.383\%$$
$$f(-1)\,f(-0.40933) = 29(-1.42411) = -41.29925$$

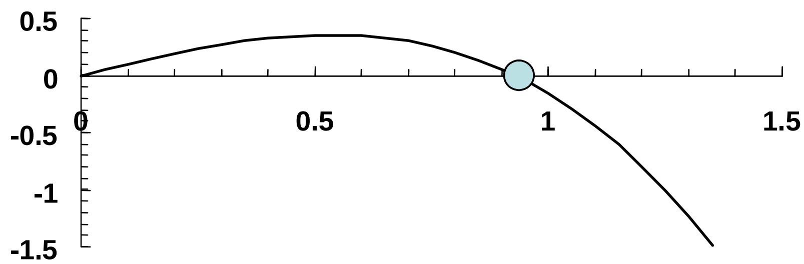
Consequently, the root is in the first interval and the upper guess is redefined as x_u = –0.40933. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	-1	29.00	0	-13	-0.30952	-4.90027	
2	-1	29.00	-0.30952	-4.90027	-0.40933	-1.42411	24.383%

3	-1	29.00	-0.40933	-1.42411	-0.43698	-0.38199	6.327%
4	-1	29.00	-0.43698	-0.38199	-0.44430	-0.10024	1.647%
5	-1	29.00	-0.44430	-0.10024	-0.44621	-0.02615	0.429%

Therefore, after five iterations we obtain a root estimate of **–0.44621** with an approximate error of 0.429%, which is below the stopping criterion of 1%.

5.5 A graph indicates that a nontrivial root (i.e., nonzero) is located at about 0.93.



Using bisection, the first iteration is

$$x_r = \frac{0.5+1}{2} = 0.75$$
$$f(0.5)\,f(0.75) = 0.354426(0.2597638) = 0.092067$$

Therefore, the root is in the second interval and the lower guess is redefined as x_l = 0.75. The second iteration is

$$x_r = \frac{0.75+1}{2} = 0.875$$
$$\varepsilon_a = \left|\frac{0.875-0.75}{0.875}\right| 100\% = 14.29\%$$
$$f(0.75)\,f(0.875) = 0.259764(0.0976216) = 0.025359$$

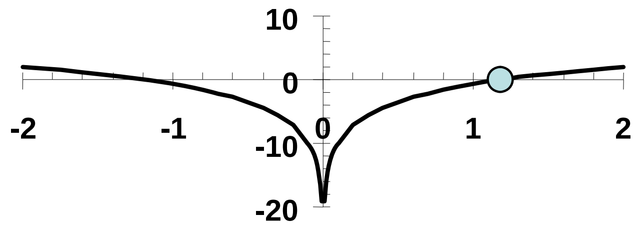
Because the product is positive, the root is in the second interval and the lower guess is redefined as x_l = 0.875. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	0.5	0.354426	1	-0.158529	0.75	0.2597638	
2	0.75	0.259764	1	-0.158529	0.875	0.0976216	14.29%
3	0.875	0.097622	1	-0.158529	0.9375	-0.0178935	6.67%
4	0.875	0.097622	0.9375	-0.0178935	0.90625	0.0429034	3.45%
5	0.90625	0.042903	0.9375	-0.0178935	0.921875	0.0132774	1.69%

Consequently, after five iterations we obtain a root estimate of **0.921875** with an approximate error of 1.69%, which is below the stopping criterion of 2%. The result can be checked by substituting it into the original equation to verify that it is close to zero.

$$f(x) = \sin(x) - x^3 = \sin(0.921875) - 0.921875^3 = 0.0132774$$

5.6 (a) A graph of the function indicates a positive real root at approximately x = 1.2.



(b) Using bisection, the first iteration is

$$x_r = \frac{0.5+2}{2} = 1.25$$
$$\varepsilon_a = \left|\frac{2-0.5}{2+0.5}\right| 100\% = 60\%$$
$$f(0.5)\,f(1.25) = -3.47259(0.19257) = -0.66873$$

Therefore, the root is in the first interval and the upper guess is redefined as $x_u = 1.25$. The second iteration is

$$x_r = \frac{0.5+1.25}{2} = 0.875$$
$$\varepsilon_a = \left|\frac{0.875-1.25}{0.875}\right| 100\% = 42.86\%$$
$$f(0.5)\,f(0.875) = -3.47259(-1.23413) = 4.28561$$

Consequently, the root is in the second interval and the lower guess is redefined as $x_l = 0.875$. All the iterations are displayed in the following table:

i	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0.50000	2.00000	1.25000	-3.47259	0.19257	-0.66873	
2	0.50000	1.25000	0.87500	-3.47259	-1.23413	4.28561	42.86%
3	0.87500	1.25000	1.06250	-1.23413	-0.4575	0.56461	17.65%

Thus, after three iterations, we obtain a root estimate of **1.0625** with an approximate error of 17.65%.

(c) Using false position, the first iteration is

$$x_r = 2 - \frac{2.07259(0.5-2)}{-3.47259-2.07259} = 1.43935$$
$$f(0.5)\,f(1.43935) = -3.47259(0.75678) = -2.62797$$

Therefore, the root is in the first interval and the upper guess is redefined as $x_u = 1.43935$. The second iteration is

$$x_r = 1.43935 - \frac{0.75678(0.5-1.43935)}{-3.47259-0.75678} = 1.27127$$
$$\varepsilon_a = \left|\frac{1.27127-1.43935}{1.27127}\right| 100\% = 13.222\%$$
$$f(0.5)\,f(1.27127) = -3.47259(0.26007) = -0.90312$$

Consequently, the root is in the first interval and the upper guess is redefined as x_u = 1.27127. All the iterations are displayed in the following table:

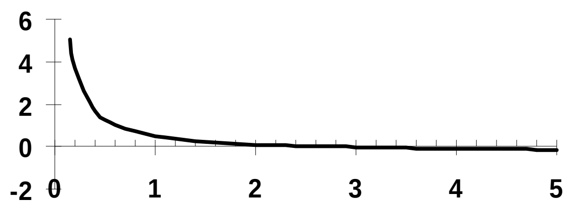
iteration	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0.5	2.00000	-3.47259	2.07259	1.43935	0.75678	-2.62797	
2	0.5	1.43935	-3.47259	0.75678	1.27127	0.26007	-0.90312	13.222%
3	0.5	1.27127	-3.47259	0.26007	1.21753	0.08731	-0.30319	4.414%

After three iterations we obtain a root estimate of **1.21753** with an approximate error of 4.414%.

5.7 (a) $(0.8-0.3x)=0$

$$x=\frac{0.8}{0.3}=2.666667$$

(b) The graph of the function indicates a root between x = 2 and 3. Note that the shape of the curve suggests that it may be ill-suited for solution with the false-position method (refer to Fig. 5.14)



(c) Using false position, the first iteration is

$$x_r=3-\frac{-0.03333(1-3)}{0.5-(-0.03333)}=2.875$$

$$\varepsilon_t=\left|\frac{2.66667-2.875}{2.66667}\right|100\%=7.81\%$$

$$f(1)f(2.875)=0.5(-0.02174)=-0.01087$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 2.875. The second iteration is

$$x_r=2.875-\frac{-0.03333(1-2.875)}{0.5-(-0.03333)}=2.79688$$

$$\varepsilon_a=\left|\frac{2.79688-2.875}{2.79688}\right|100\%=2.793\%$$

$$\varepsilon_t=\left|\frac{2.66667-2.79688}{2.66667}\right|100\%=4.88\%$$

$$f(1)f(2.79688)=0.5(-0.01397)=-0.00698$$

Consequently, the root is in the first interval and the upper guess is redefined as x_u = 2.79688. All the iterations are displayed in the following table:

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a	ε_t

1	1	3.00000	0.5	-0.03333	2.87500	-0.02174	-0.01087		7.81%
2	1	2.87500	0.5	-0.02174	2.79688	-0.01397	-0.00698	2.793%	4.88%
3	1	2.79688	0.5	-0.01397	2.74805	-0.00888	-0.00444	1.777%	3.05%

Therefore, after three iterations we obtain a root estimate of **2.74805** with an approximate error of 1.777%. Note that the true error is greater than the approximate error. This is not good because it means that we could stop the computation based on the erroneous assumption that the true error is at least as good as the approximate error. This is due to the slow convergence that results from the function's shape.

5.8 The square root of 18 can be set up as a roots problem by determining the positive root of the function

$$f(x) = x^2 - 18 = 0$$

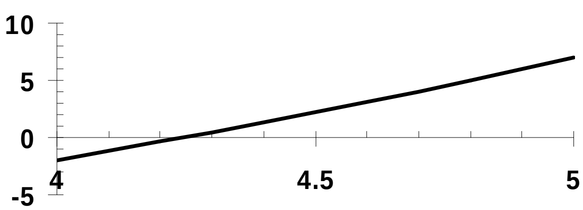
Using false position, the first iteration is

$$x_r = 5 - \frac{7(4-5)}{-2-7} = 4.22222$$
$$f(4)\,f(4.22222) = -2(-0.17284) = 0.34568$$

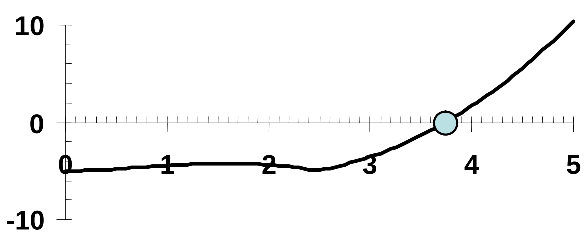
Therefore, the root is in the second interval and the lower guess is redefined as x_l = 4.22222. The second iteration is

$$x_r = 4.22222 - \frac{7(4.22222-5)}{-0.17284-7} = 4.24096$$
$$\varepsilon_a = \left|\frac{4.24096-4.22222}{4.24096}\right| 100\% = 0.442\%$$

Thus, the computation can be stopped after just two iterations because 0.442% < 0.5%. Note that the true value is 4.2426. The technique converges so quickly because the function is very close to being a straight line in the interval between the guesses as in the plot of the function shown below.



5.9 A graph of the function indicates a positive real root at approximately x = 3.7.



Using false position, the first iteration is

$$x_r = 5 - \frac{10.43182(0-5)}{-5-10.43182} = 1.62003$$
$$f(0)\,f(1.62003) = -5(-4.22944) = 21.147$$

Therefore, the root is in the second interval and the lower guess is redefined as x_l = 1.62003. The remaining iterations are summarized below

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0	5.00000	-5	10.43182	1.62003	-4.22944	21.14722	
2	1.62003	5.00000	-4.2294	10.43182	2.59507	-4.72984	20.00459	37.573%
3	2.59507	5.00000	-4.7298	10.43182	3.34532	-2.14219	10.13219	22.427%
4	3.34532	5.00000	-2.1422	10.43182	3.62722	-0.69027	1.47869	7.772%
5	3.62722	5.00000	-0.6903	10.43182	3.71242	-0.19700	0.13598	2.295%
6	3.71242	5.00000	-0.197	10.43182	3.73628	-0.05424	0.01069	0.639%

The final result, x_r = 3.73628, can be checked by substituting it into the original function to yield a near-zero result,

$$f(3.73628) = (3.73628)^2\left|\cos\sqrt{3.73628}\right| - 5 = -0.05424$$

5.10 Using false position, the first iteration is

$$x_r = 6 - \frac{7(4.5-6)}{-3.6875-7} = 5.01754$$
$$f(4.5)\,f(5.01754) = -3.6875(-1.00147) = 3.69294$$

Therefore, the root is in the second interval and the lower guess is redefined as x_u = 5.01754. The true error can be computed as

$$\varepsilon_t = \left|\frac{5.60979 - 5.01754}{5.60979}\right| 100\% = 10.56\%$$

The second iteration is

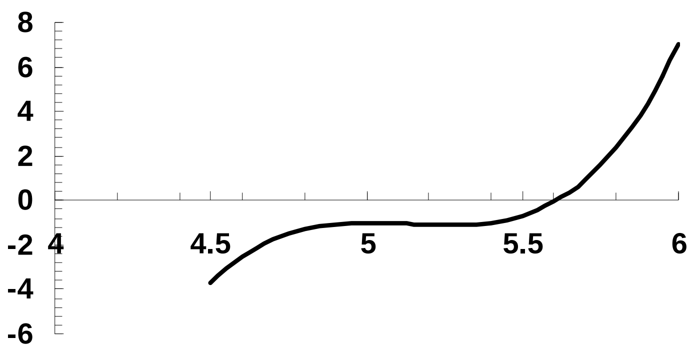
$$x_r = 6 - \frac{7(5.01754-6)}{-1.00147-7} = 5.14051$$
$$\varepsilon_t = \left|\frac{5.60979 - 5.14051}{5.60979}\right| 100\% = 8.37\%$$
$$\varepsilon_a = \left|\frac{5.14051 - 5.01754}{5.14051}\right| 100\% = 2.392\%$$
$$f(5.01754)\,f(5.14051) = -1.00147(-1.06504) = 1.06661$$

Consequently, the root is in the second interval and the lower guess is redefined as x_u = 5.14051. All the iterations are displayed in the following table:

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a, %	ε_t, %
1	4.50000	6	-3.68750	7.00000	5.01754	-1.00147	3.69294		10.56
2	5.01754	6	-1.00147	7.00000	5.14051	-1.06504	1.06661	2.392	8.37
3	5.14051	6	-1.06504	7.00000	5.25401	-1.12177	1.19473	2.160	6.34

4	5.25401	6	-1.12177	7.00000	5.35705	-1.07496	1.20586	1.923	4.51
5	5.35705	6	-1.07496	7.00000	5.44264	-0.90055	0.96805	1.573	2.98
6	5.44264	6	-0.90055	7.00000	5.50617	-0.65919	0.59363	1.154	1.85
7	5.50617	6	-0.65919	7.00000	5.54867	-0.43252	0.28511	0.766	1.09

Notice that the results have the undesirable feature that the true error is greater than the approximate error. This is not good because it means that we could stop the computation based on the erroneous assumption that the true error is at least as good as the approximate error. This is due to the slow convergence that results from the function's shape as shown in the following plot (recall Fig. 5.14).



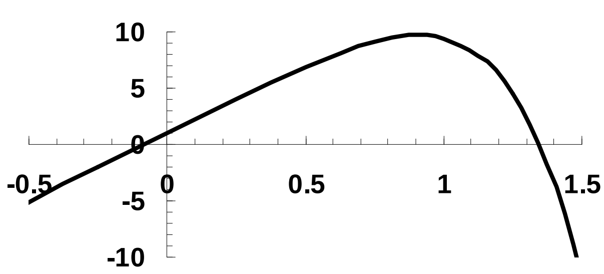
5.11 (a)

$$x_r = (80)^{1/3.5} = 3.497357$$

(b) Here is a summary of the results obtained with false position to the point that the approximate error falls below the stopping criterion of 2.5%:

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	2	5	-68.68629	199.5085	2.76832	-44.70153	3070.382	
2	2.768318	5	-44.70153	199.5085	3.17682	-22.85572	1021.686	12.859%
3	3.176817	5	-22.85572	199.5085	3.36421	-10.16193	232.258	5.570%
4	3.364213	5	-10.16193	199.5085	3.44349	-4.229976	42.985	2.302%

5.12 A plot of the function indicates a maximum at about 0.9.



In order to determine the maximum with a root location technique, we must first differentiate the function to yield

$$f'(x) = -12x^5 - 6.4x^3 + 12$$

The root of this function represents an extremum. Using bisection and the recommended initial guesses gives:

i	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0.00000	1.00000	0.50000	12.00000	10.82500	129.9000	100.00%
2	0.50000	1.00000	0.75000	10.82500	6.45234	69.8466	33.33%
3	0.75000	1.00000	0.87500	6.45234	1.55759	10.0501	14.29%
4	0.87500	1.00000	0.93750	1.55759	-1.96379	-3.0588	6.67%
5	0.87500	0.93750	0.90625	1.55759	-0.09884	-0.1539	3.45%

The maximum can be determined by substituting the root into the original equation to give

$$f(0.90625) = -2(0.90625)^6 - 1.6(0.90625)^4 + 12(0.90625) + 1 = 9.68783$$

5.13 The correct mass can be determined by finding the root of

$$f(m) = \frac{9.8m}{15}\left(1 - e^{-(15/m)9}\right) - 35 = 0$$

Here are the results of using false position with initial guesses of 50 and 70 kg:

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	50	70.00000	-4.52871	4.085733	60.51423	0.288464	-1.30637	
2	50	60.51423	-4.52871	0.288464	59.88461	0.018749	-0.08491	1.051%
3	50	59.88461	-4.52871	0.018749	59.84386	0.001212	-0.00549	0.068%

Thus, after 3 iterations, a value of 59.84386 kg is determined with an approximate error of 0.068%. This result can be verified by substituting it into the equation for velocity to give

$$v = \frac{9.8(59.84386)}{15}\left(1 - e^{-(15/59.84386)9}\right) = 35.00121\,\frac{\text{m}}{\text{s}}$$

5.14 [First printing errata: the initial guesses should be x_l = 3 and x_u = 5].

The function to evaluate is

$$f(c) = \frac{9.81(80)}{c}\left(1 - e^{-(c/80)4}\right) - 36 = 0$$

The first iteration is

$$x_r = \frac{3+5}{2} = 4$$
$$f(3)f(4) = 0.438793(-1.28057) = -0.19086$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 4. The second iteration is

$$x_r = \frac{3+4}{2} = 3.5$$

$$\varepsilon_a = \left|\frac{3.5-4}{3.5}\right| 100\% = 14.29\%$$

$$f(3)\,f(3.5) = 0.438793(-0.0016771) = -0.00074$$

Therefore, the root is in the first interval and the upper bound is redefined as x_u = 3.5. The remaining iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	$\|\varepsilon_a\|$
1	3	0.43879	5	-1.28057	4	-0.43497	
2	3	0.43879	4	-0.43497	3.5	-0.00168	14.29%
3	3	0.43879	3.5	-0.00168	3.25	0.21765	7.69%
4	3.25	0.21765	3.5	-0.00168	3.375	0.10776	3.70%
5	3.375	0.10776	3.5	-0.00168	3.4375	0.05299	1.82%

Thus, after five iterations, we obtain a root estimate of **3.4375** with an approximate error of 1.82%.

5.15 Solve for the reactions:

R_1=265 lbs. R_2= 285 lbs.

Write beam equations:

0<*x*<3

$$M + (16.667x^2)\frac{x}{3} - 265x = 0$$

$$(1) \quad M = 265x - 5.5555556x^3$$

3<*x*<6

$$M + 100(x-3)\left(\frac{x-3}{2}\right) + 150\left(x - \frac{2}{3}(3)\right) - 265x = 0$$

$$(2) \quad M = -50x^2 + 415x - 150$$

6<*x*<10

$$M = 150\left(x - \frac{2}{3}(3)\right) + 300(x-4.5) - 265x$$

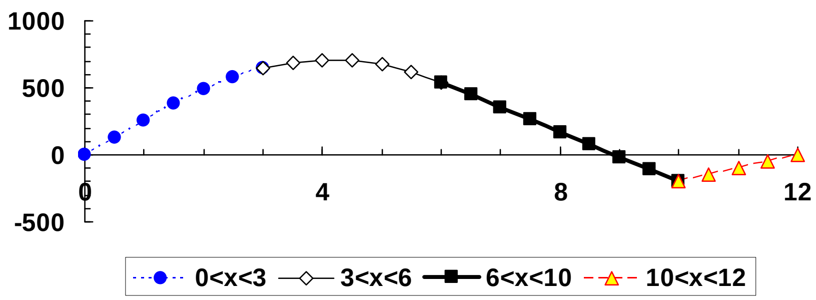
$$(3) \quad M = -185x + 1650$$

10<*x*<12

$$M + 100(12 - x) = 0$$

$$(4) \quad M = 100x - 1200$$

A plot of these equations can be generated:



Combining Equations:

Because the curve crosses the axis between 6 and 10, use (3).

(3) $M = -185x + 1650$

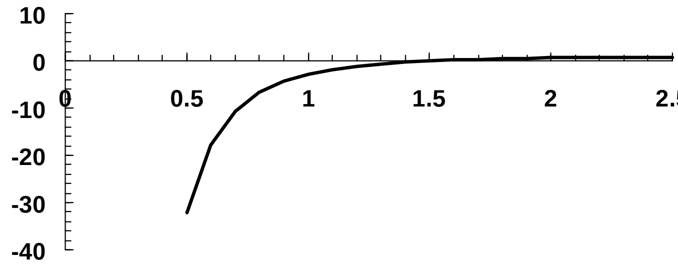
Set $x_L = 6; x_U = 10$

$M(x_L) = 540$
$M(x_U) = -200$ $\qquad x_r = \dfrac{x_L + x_U}{2} = 8$
$M(x_R) = 170 \rightarrow \text{ replaces } x_L$
$M(x_L) = 170$
$M(x_U) = -200$ $\qquad x_r = \dfrac{8+10}{2} = 9$
$M(x_R) = -15 \rightarrow \text{ replaces } x_U$
$M(x_L) = 170$
$M(x_U) = -15$ $\qquad x_r = \dfrac{8+9}{2} = 8.5$
$M(x_R) = 77.5 \rightarrow \text{ replaces } x_L$
$M(x_L) = 77.5$
$M(x_U) = -15$ $\qquad x_r = \dfrac{8.5+9}{2} = 8.75$
$M(x_R) = 31.25 \rightarrow \text{ replaces } x_L$
$M(x_L) = 31.25$
$M(x_U) = -15$ $\qquad x_r = \dfrac{8.75+9}{2} = 8.875$
$M(x_R) = 8.125 \rightarrow \text{ replaces } x_L$
$M(x_L) = 8.125$
$M(x_U) = -15$ $\qquad x_r = \dfrac{8.875+9}{2} = 8.9375$
$M(x_R) = -3.4375 \rightarrow \text{ replaces } x_U$
$M(x_L) = 8.125$
$M(x_U) = -3.4375$ $\qquad x_r = \dfrac{8.875+8.9375}{2} = 8.90625$
$M(x_R) = 2.34375 \rightarrow \text{ replaces } x_L$
$M(x_L) = 2.34375$
$M(x_U) = -3.4375$ $\qquad x_r = \dfrac{8.90625+8.9375}{2} = 8.921875$
$M(x_R) = -0.546875 \rightarrow \text{ replaces } x_U$
$M(x_L) = 2.34375$
$M(x_U) = -0.546875$ $\qquad x_r = \dfrac{8.90625+8.921875}{2} = 8.9140625$
$M(x_R) = 0.8984$ $\qquad$ **Therefore,** $x = 8.91$ *feet*

5.16 (a) The function to be evaluated is

$$f(y) = 1 - \frac{400}{9.81(3y + y^2/2)^3}(3 + y)$$

A graph of the function indicates a positive real root at approximately 1.5.



(b) Using bisection, the first iteration is

$$x_r = \frac{0.5+2.5}{2} = 1.5$$
$$f(0.5)f(1.5) = -32.2582(-0.030946) = 0.998263$$

Therefore, the root is in the second interval and the lower guess is redefined as x_l = 1.5. The second iteration is

$$x_r = \frac{1.5+2.5}{2} = 2$$
$$\varepsilon_a = \left|\frac{2-1.5}{2}\right| 100\% = 25\%$$
$$f(1.5)f(2) = -0.030946(0.601809) = -0.018624$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 2. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	0.5	–32.2582	2.5	0.813032	1.5	–0.030946	
2	1.5	–0.03095	2.5	0.813032	2	0.601809	25.00%
3	1.5	–0.03095	2	0.601809	1.75	0.378909	14.29%
4	1.5	–0.03095	1.75	0.378909	1.625	0.206927	7.69%
5	1.5	–0.03095	1.625	0.206927	1.5625	0.097956	4.00%
6	1.5	–0.03095	1.5625	0.097956	1.53125	0.036261	2.04%
7	1.5	–0.03095	1.53125	0.036261	1.515625	0.003383	1.03%
8	1.5	–0.03095	1.515625	0.003383	1.5078125	–0.013595	0.52%

After eight iterations, we obtain a root estimate of **1.5078125** with an approximate error of 0.52%.

(c) Using false position, the first iteration is

$$x_r = 2.5 - \frac{0.81303(0.5-2.5)}{-32.2582-0.81303} = 2.45083$$
$$f(0.5)f(2.45083) = -32.25821(0.79987) = -25.80248$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 2.45083. The second iteration is

$$x_r = 2.45083 - \frac{0.79987(0.5-2.45083)}{-32.25821-0.79987} = 2.40363$$

$$\varepsilon_a = \left|\frac{2.40363 - 2.45083}{2.40363}\right| 100\% = 1.96\%$$

$$f(0.5)\, f(2.40363) = -32.2582(0.78612) = -25.35893$$

The root is in the first interval and the upper guess is redefined as x_u = 2.40363. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	0.5	–32.2582	2.50000	0.81303	2.45083	0.79987	
2	0.5	–32.2582	2.45083	0.79987	2.40363	0.78612	1.96%
3	0.5	–32.2582	2.40363	0.78612	2.35834	0.77179	1.92%
4	0.5	–32.2582	2.35834	0.77179	2.31492	0.75689	1.88%
5	0.5	–32.2582	2.31492	0.75689	2.27331	0.74145	1.83%
6	0.5	–32.2582	2.27331	0.74145	2.23347	0.72547	1.78%
7	0.5	–32.2582	2.23347	0.72547	2.19534	0.70900	1.74%
8	0.5	–32.2582	2.19534	0.70900	2.15888	0.69206	1.69%
9	0.5	–32.2582	2.15888	0.69206	2.12404	0.67469	1.64%
10	0.5	–32.2582	2.12404	0.67469	2.09077	0.65693	1.59%

After ten iterations we obtain a root estimate of **2.09077** with an approximate error of 1.59%. Thus, after ten iterations, the false position method is converging at a very slow pace and is still far from the root in the vicinity of 1.5 that we detected graphically.

Discussion: This is a classic example of a case where false position performs poorly and is inferior to bisection. Insight into these results can be gained by examining the plot that was developed in part **(a)**. This function violates the premise upon which false position was based–that is, if $f(x_u)$ is much closer to zero than $f(x_l)$, then the root is closer to x_u than to x_l (recall Figs. 5.12 and 5.14). Because of the shape of the present function, the opposite is true.

5.17 The equation to be solved is

$$f(h) = \pi R h^2 - \left(\frac{\pi}{3}\right) h^3 - V$$

Here is a summary of the results obtained with three iterations of false position:

i	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0	3.00000	-30	26.54867	1.59155	-10.3485	310.45424	
2	1.59155	3.00000	-10.348	26.54867	1.98658	-1.01531	10.50688	19.885%
3	1.98658	3.00000	-1.0153	26.54867	2.02390	-0.07591	0.07708	1.844%

The result can be verified by substituting it into the volume equation to give

$$V = \pi(2.0239)^2 \frac{3(3) - 2.0239}{3} = 29.92409$$

5.18 (a) Equation (5.5) can be used to determine the number of iterations

$$n = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right) = \log_2\left(\frac{40}{0.05}\right) = 9.6439$$

which can be rounded up to 10 iterations.

(b) Here is an M-file that evaluates the temperature in ^{o}C using 11 iterations of bisection based on a given value of the oxygen saturation concentration in freshwater:

```
function TC = TempEval(osf)
% function to evaluate the temperature in degrees C based
% on the oxygen saturation concentration in freshwater (osf).
xl = 0 + 273.15;
xu = 40 + 273.15;
if fTa(xl,osf)*fTa(xu,osf)>0  %if guesses do not bracket
  error('no bracket')   %display an error message and terminate
end
xr = xl;
for i = 1:10
  xrold = xr;
  xr = (xl + xu)/2;
  if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
  test = fTa(xl,osf)*fTa(xr,osf);
  if test < 0
    xu = xr;
  elseif test > 0
    xl = xr;
  else
    ea = 0;
  end
end
TC = xr - 273.15;

function f = fTa(Ta, osf)
f = -139.34411 + 1.575701e5/Ta - 6.642308e7/Ta^2;
f = f + 1.2438e10/Ta^3 - 8.621949e11/Ta^4;
f = f - log(osf);
```

The function can be used to evaluate the test cases:

```
>> TempEval(8)

ans =
   26.7578

>> TempEval(10)

ans =
   15.3516

>> TempEval(12)

ans =
    7.4609
```

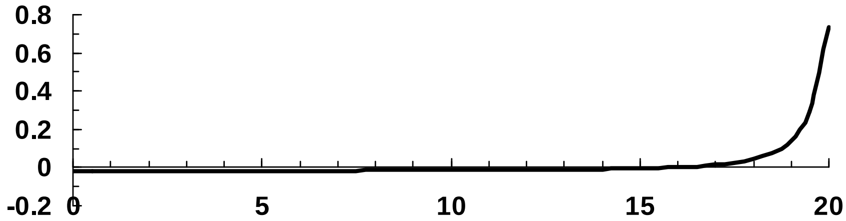
Note that these values can be compared with the true values to verify that the errors are less than 0.05:

o_{sf}	Approximation	Exact	Error
8	26.75781	26.78017	0.0224
10	15.35156	15.38821	0.0366
12	7.46094	7.46519	0.0043

5.19 The function to be solved is

$$f(x) = \frac{(4+x)}{(42-2x)^2(28-x)} - 0.016 = 0$$

(a) A plot of the function indicates a root at about x = 16.



(b) The shape of the function indicates that false position would be a poor choice (recall Fig. 5.14). Bisection with initial guesses of 0 and 20 can be used to determine a root of 15.85938 after 8 iterations with ε_a = 0.493%. Note that false position would have required 68 iterations to attain comparable accuracy.

i	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0	20	10	-0.01592	-0.01439	0.000229	100.000%
2	10	20	15	-0.01439	-0.00585	8.42E-05	33.333%
3	15	20	17.5	-0.00585	0.025788	-0.00015	14.286%
4	15	17.5	16.25	-0.00585	0.003096	-1.8E-05	7.692%
5	15	16.25	15.625	-0.00585	-0.00228	1.33E-05	4.000%
6	15.625	16.25	15.9375	-0.00228	0.000123	-2.8E-07	1.961%
7	15.625	15.9375	15.78125	-0.00228	-0.00114	2.59E-06	0.990%
8	15.78125	15.9375	15.85938	-0.00114	-0.00052	5.98E-07	0.493%

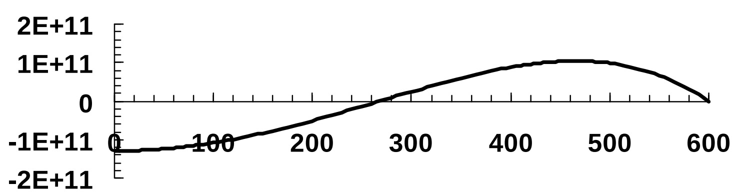
5.20 This problem can be solved by determining the root of the derivative of the elastic curve

$$\frac{dy}{dx} = 0 = \frac{w_0}{120EIL}\left(-5x^4 + 6L^2x^2 - L^4\right)$$

Therefore, after substituting the parameter values, we must determine the root of

$$f(x) = -5x^4 + 2{,}160{,}000x^2 - 1.296\times 10^{11} = 0$$

A plot of the function indicates a root at about x = 270.



Bisection can be used to determine the root. Here are the first few iterations:

i	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l)\times f(x_r)$	ε_a
1	0	500	250	-1.3E+11	-1.4E+10	1.83E+21	
2	250	500	375	-1.4E+10	7.53E+10	-1.1E+21	33.33%
3	250	375	312.5	-1.4E+10	3.37E+10	-4.8E+20	20.00%

4	250	312.5	281.25	-1.4E+10	9.97E+09	-1.4E+20	11.11%
5	250	281.25	265.625	-1.4E+10	-2.1E+09	2.95E+19	5.88%

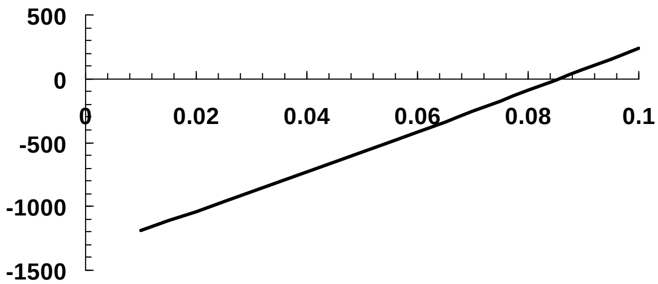
After 20 iterations, the root is determined as x = 268.328. This value can be substituted into Eq. (P8.18) to compute the maximum deflection as

$$y = \frac{2.5}{120(50{,}000)30{,}000(600)}(-(268.328)^5 + 720{,}000(268.328)^3 - 1.296\times 10^{11}(268.328)) = -0.51519$$

5.21 The solution can be formulated as

$$f(i) = 25{,}000\frac{i(1+i)^6}{(1+i)^6 - 1} - 5{,}500$$

A plot of this function suggests a root at about 0.086:

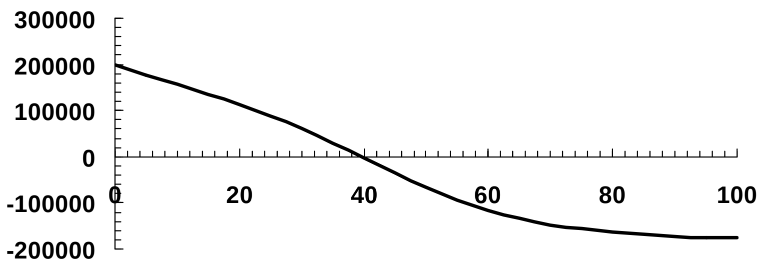


A numerical method can be used to determine that the root is 0.085595.

5.22 (a) The solution can be formulated as

$$f(t) = 1.2\left(75{,}000e^{-0.045t} + 100{,}000\right) - \frac{300{,}000}{1 + 29e^{-0.08t}}$$

A plot of this function suggests a root at about 40:



(b) The false-position method can be implemented with the results summarized as

i	t_l	t_u	$f(t_l)$	$f(t_u)$	t_r	$f(t_r)$	$f(t_l)\times f(t_r)$	ε_a
1	0	100.0000	200000	-176110	53.1760	-84245	-1.685E+10	
2	0	53.1760	200000	-84245	37.4156	14442.8	2.889E+09	42.123%
3	37.4156	53.1760	14443	-84245	39.7221	-763.628	-1.103E+07	5.807%
4	37.4156	39.7221	14443	-763.628	39.6063	3.545288	5.120E+04	0.292%
5	39.6063	39.7221	4	-763.628	39.6068	0.000486	1.724E-03	0.001%

(c) The modified secant method (with $\delta = 0.01$) can be implemented with the results summarized as

i	t_i	$f(t_i)$	δt_i	$t_i+\delta t_i$	$f(t_i+\delta t_i)$	$f'(t_i)$	ε_a
0	50	-66444.8	0.50000	50.5	-69357.6	-5825.72	
1	38.5946	6692.132	0.38595	38.98053	4143.604	-6603.33	29.552%
2	39.6080	-8.14342	0.39608	40.00411	-2632.32	-6625.36	2.559%
3	39.6068	-0.00345	0.39607	40.00287	-2624.09	-6625.35	0.003%

For both parts **(b)** and **(c)**, the root is determined to be $t = 39.6068$. At this time, the ratio of the suburban to the urban population is 135,142.5/112,618.7 = 1.2.

5.23 Here is a VBA program to implement the bisection function (Fig. 5.10) in a user-friendly format:

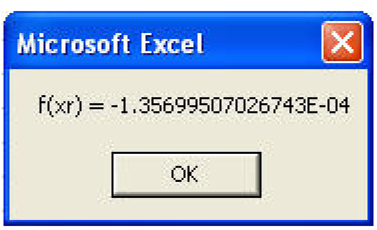
```
Option Explicit

Sub TestBisect()
Dim imax As Integer, iter As Integer
Dim x As Double, xl As Double, xu As Double
Dim es As Double, ea As Double, xr As Double
Dim root As Double
'input information from the user
Sheets("Sheet1").Select
Range("b4").Select
xl = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
xu = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
es = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
imax = ActiveCell.Value
Range("b4").Select
If f(xl) * f(xu) < 0 Then
  'if the initial guesses are valid, implement bisection
  'and display results
  root = Bisect(xl, xu, es, imax, xr, iter, ea)
  MsgBox "The root is: " & root
  MsgBox "Iterations: " & iter
  MsgBox "Estimated error: " & ea
  MsgBox "f(xr) = " & f(xr)
Else
  'if the initial guesses are invalid,
  'display an error message
  MsgBox "No sign change between initial guesses"
End If
End Sub

Function Bisect(xl, xu, es, imax, xr, iter, ea)
Dim xrold As Double, test As Double
iter = 0
Do
  xrold = xr
  'determine new root estimate
  xr = (xl + xu) / 2
  iter = iter + 1
  'determine approximate error
  If xr <> 0 Then
    ea = Abs((xr - xrold) / xr) * 100
  End If
  'determine new bracket
```

```
  test = f(xl) * f(xr)
  If test < 0 Then
    xu = xr
  ElseIf test > 0 Then
    xl = xr
  Else
    ea = 0
  End If
  'terminate computation if stopping criterion is met
  'or maximum iterations are exceeded
  If ea < es Or iter >= imax Then Exit Do
Loop
Bisect = xr
End Function

Function f(c)
f = 9.8 * 68.1 / c * (1 - Exp(-(c / 68.1) * 10)) - 40
End Function
```

For Example 5.3, the Excel worksheet used for input looks like:

	A	B	C	D	E
1	**Bisection Example**				
2					
3					
4	**xl**	**12**		Run	
5	**xu**	**16**			
6	**es**	**0.01**			
7	**imax**	**25**			
8					

The program yields a root of 14.78027 after 12 iterations. The approximate error at this point is 6.6×10^{-3} %. These results are all displayed as message boxes. For example, the solution check is displayed as



5.24 Here is a VBA program to implement the bisection function that minimizes function evaluations (Fig. 5.11) in a user-friendly program:

```
Option Explicit

Sub TestBisectMin()
Dim imax As Integer, iter As Integer
Dim x As Double, xl As Double, xu As Double
Dim es As Double, ea As Double, xr As Double
Dim root As Double
'input information from the user
Sheets("Sheet1").Select
Range("b4").Select
xl = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
```

```
xu = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
es = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
imax = ActiveCell.Value
Range("b4").Select
If f(xl) * f(xu) < 0 Then
  'if the initial guesses are valid, implement bisection
  'and display results
  root = BisectMin(xl, xu, es, imax, xr, iter, ea)
  MsgBox "The root is: " & root
  MsgBox "Iterations: " & iter
  MsgBox "Estimated error: " & ea
  MsgBox "f(xr) = " & f(xr)
Else
  'if the initial guesses are invalid,
  'display an error message
  MsgBox "No sign change between initial guesses"
End If
End Sub

Function BisectMin(xl, xu, es, imax, xr, iter, ea)
Dim xrold As Double, test As Double, fl As Double, fr As Double
iter = 0
fl = f(xl)
Do
  xrold = xr
  'determine new root estimate
  xr = (xl + xu) / 2
  fr = f(xr)
  iter = iter + 1
  'determine approximate error
  If xr <> 0 Then
    ea = Abs((xr - xrold) / xr) * 100
  End If
  'determine new bracket
  test = fl * fr
  If test < 0 Then
    xu = xr
  ElseIf test > 0 Then
    xl = xr
   fl = fr
  Else
    ea = 0
  End If
  'terminate computation if stopping criterion is met
  'or maximum iterations are exceeded
  If ea < es Or iter >= imax Then Exit Do
Loop
BisectMin = xr
End Function

Function f(x)
f = x ^ 10 - 1
End Function
```

For Example 5.6, the Excel worksheet used for input looks like:

	A	B	C	D	E
1	Bisection Example				
2					
3					
4	xl	0		Run	
5	xu	1.3			
6	es	0.01			
7	imax	25			
8					

After 14 iterations, the program yields

The number of function evaluations per iteration can be determined by inspecting the code. After the initial evaluation of the function at the lower bound (`fl = f(xl)`), there is a single additional evaluation per iteration (`fr = f(xr)`). Therefore, the number of function evaluations is equal to the number of iterations plus 1. In contrast, the pseudocode from Fig. 5.10 which does not attempt to minimize function results in function evaluations equaling twice the iterations. Thus, the code in Fig. 5.11 should execute about twice as fast as Fig. 5.10.

5.25 Here is a VBA program to implement false position that is similar in structure to the bisection algorithm outlined in Fig. 5.10:

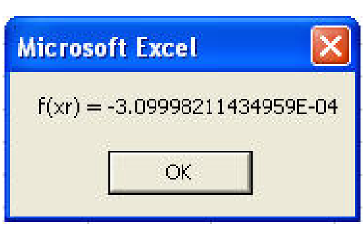
```
Option Explicit

Sub TestFP()
Dim imax As Integer, iter As Integer
Dim x As Double, xl As Double, xu As Double
Dim es As Double, ea As Double, xr As Double
Dim root As Double
'input information from the user
Sheets("Sheet1").Select
Range("b4").Select
xl = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
xu = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
es = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
imax = ActiveCell.Value
Range("b4").Select
If f(xl) * f(xu) < 0 Then
  'if the initial guesses are valid, implement bisection
  'and display results
  root = FalsePos(xl, xu, es, imax, xr, iter, ea)
  MsgBox "The root is: " & root
  MsgBox "Iterations: " & iter
  MsgBox "Estimated error: " & ea
  MsgBox "f(xr) = " & f(xr)
Else
  'if the initial guesses are invalid,
```

```
  'display an error message
  MsgBox "No sign change between initial guesses"
End If
End Sub

Function FalsePos(xl, xu, es, imax, xr, iter, ea)
Dim xrold As Double, test As Double
iter = 0
Do
  xrold = xr
  'determine new root estimate
  xr = xu - f(xu) * (xl - xu) / (f(xl) - f(xu))
  iter = iter + 1
  'determine approximate error
  If xr <> 0 Then
    ea = Abs((xr - xrold) / xr) * 100#
  End If
  'determine new bracket
  test = f(xl) * f(xr)
  If (test < 0) Then
    xu = xr
  ElseIf (test > 0) Then
    xl = xr
  Else
    ea = 0#
  End If
  'terminate computation if stopping criterion is met
  'or maximum iterations are exceeded
  If ea < es Or iter >= imax Then Exit Do
Loop
FalsePos = xr
End Function

Function f(c)
f = 9.8 * 68.1 / c * (1 - Exp(-(c / 68.1) * 10)) - 40
End Function
```

For Example 5.5, the Excel worksheet used for input looks like:

	A	B	C	D	E	F
1	**False Position Example**					
2						
3						
4	**xl**	**12**			RUN	
5	**xu**	**16**				
6	**es**	**0.01**				
7	**imax**	**25**				
8						

The program yields a root of 14.78036 after 4 iterations. The approximate error at this point is 9.015×10^{-3} %. These results are all displayed as message boxes. For example, the solution check is displayed as



5.26 Here is a VBA Sub procedure to implement the false position method which minimizes function evaluations. It is set up to evaluate Example 5.6.

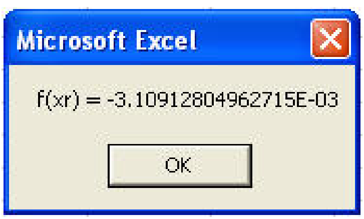
```
Option Explicit

Sub TestFP()
Dim imax As Integer, iter As Integer
Dim x As Double, xl As Double, xu As Double
Dim es As Double, ea As Double, xr As Double
Dim root As Double
'input information from the user
Sheets("Sheet1").Select
Range("b4").Select
xl = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
xu = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
es = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
imax = ActiveCell.Value
Range("b4").Select
If f(xl) * f(xu) < 0 Then
  'if the initial guesses are valid, implement bisection
  'and display results
  root = FalsePosMin(xl, xu, es, imax, xr, iter, ea)
  MsgBox "The root is: " & root
  MsgBox "Iterations: " & iter
  MsgBox "Estimated error: " & ea
  MsgBox "f(xr) = " & f(xr)
Else
  'if the initial guesses are invalid,
  'display an error message
  MsgBox "No sign change between initial guesses"
End If
End Sub

Function FalsePosMin(xl, xu, es, imax, xr, iter, ea)
Dim xrold As Double, test As Double
Dim fl As Double, fu As Double, fr As Double
iter = 0
fl = f(xl)
fu = f(xu)
Do
  xrold = xr
  'determine new root estimate
  xr = xu - fu * (xl - xu) / (fl - fu)
  fr = f(xr)
  iter = iter + 1
  'determine approximate error
  If xr <> 0 Then
    ea = Abs((xr - xrold) / xr) * 100#
  End If
  'determine new bracket
  test = fl * fr
  If (test < 0) Then
    xu = xr
    fu = fr
  ElseIf (test > 0) Then
    xl = xr
    fl = fr
```

```
  Else
    ea = 0#
  End If
  'terminate computation if stopping criterion is met
  'or maximum iterations are exceeded
  If ea < es Or iter >= imax Then Exit Do
Loop
FalsePosMin = xr
End Function

Function f(x)
f = x ^ 10 - 1
End Function
```

For Example 5.6, the Excel worksheet used for input looks like:

	A	B	C	D	E	F
1	**False Position Example**					
2						
3						
4	**xl**	**0**			RUN	
5	**xu**	**1.3**				
6	**es**	**0.01**				
7	**imax**	**100**				
8						

The program yields a root of 0.9996887 after 39 iterations. The approximate error at this point is 9.5×10^{-3} %. These results are all displayed as message boxes. For example, the solution check is displayed as



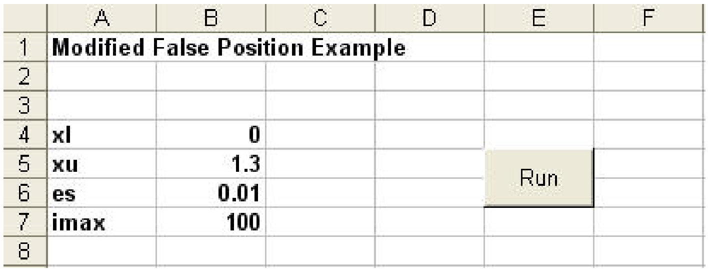
The number of function evaluations for this version is $n + 2$. This is much smaller than the number of function evaluations in the standard false position method (5n).

5.27 Here is a VBA Sub procedure to implement the modified false position method. It is set up to evaluate Example 5.5.

```
Option Explicit

Sub TestModFP()
Dim imax As Integer, iter As Integer
Dim x As Double, xl As Double, xu As Double
Dim es As Double, ea As Double, xr As Double
Dim root As Double
'input information from the user
Sheets("Sheet1").Select
Range("b4").Select
xl = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
xu = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
```

```
es = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
imax = ActiveCell.Value
Range("b4").Select
If f(xl) * f(xu) < 0 Then
  'if the initial guesses are valid, implement bisection
  'and display results
  root = ModFalsePos(xl, xu, es, imax, xr, iter, ea)
  MsgBox "The root is: " & root
  MsgBox "Iterations: " & iter
  MsgBox "Estimated error: " & ea
  MsgBox "f(xr) = " & f(xr)
Else
  'if the initial guesses are invalid,
  'display an error message
  MsgBox "No sign change between initial guesses"
End If
End Sub

Function ModFalsePos(xl, xu, es, imax, xr, iter, ea)
Dim il As Integer, iu As Integer
Dim xrold As Double, fl As Double
Dim fu As Double, fr As Double, test As Double
iter = 0
fl = f(xl)
fu = f(xu)
Do
  xrold = xr
  xr = xu - fu * (xl - xu) / (fl - fu)
  fr = f(xr)
  iter = iter + 1
  If xr <> 0 Then
    ea = Abs((xr - xrold) / xr) * 100
  End If
  test = fl * fr
  If test < 0 Then
    xu = xr
    fu = f(xu)
    iu = 0
    il = il + 1
    If il >= 2 Then fl = fl / 2
  ElseIf test > 0 Then
    xl = xr
    fl = f(xl)
    il = 0
    iu = iu + 1
    If iu >= 2 Then fu = fu / 2
  Else
    ea = 0#
  End If
  If ea < es Or iter >= imax Then Exit Do
Loop
ModFalsePos = xr
End Function

Function f(x)
f = x ^ 10 - 1
End Function
```

For Example 5.6, the Excel worksheet used for input looks like:



	A	B	C	D	E	F
1	Modified False Position Example					
2						
3						
4	xl	0				
5	xu	1.3			Run	
6	es	0.01				
7	imax	100				
8						

The program yields a root of 1.0000057 after 12 iterations. The approximate error at this point is 1.16×10^{-3} %. These results are all displayed as message boxes. For example, the solution check is displayed as

Note that the standard false position method requires 39 iterations to attain comparable accuracy.

5.28 Here is a MATLAB function to implement the algorithm:

```
function [root,Ea,ea,n] = bisect_iter(func,xl,xu,Ead,varargin)
% bisection roots zero
% [root,Ea,ea,n] = bisect_iter(func,xl,xu,Ead,varargin)
%   uses bisection method to find the root of a function
%   with a fixed number of iterations to attain
%   a prespecified tolerance
% input:
%   func = name of function
%   xl, xu = lower and upper guesses
%   Ead = (optional) desired tolerance (default = 0.000001)
%   p1,p2,... = additional parameters used by func
% output:
%   root = real root
%   Ea = absolute error
%   ea = % relative error
%   n = iterations

if func(xl,varargin{:})*func(xu,varargin{:})>0  %if guesses do not bracket a
sign change
  disp('no bracket')    %display an error message
  return                %and terminate
end
% if necessary, assign default values
if nargin<4|isempty(Ead),Ead=0.000001;end  %if Ead blank set to 0.000001
xr = xl;
% compute n and round up to next highest integer
n = round(log2((xu - xl)/Ead) + 0.5);
for i = 1:n
  xrold = xr;
  xr = (xl + xu)/2;
  if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
  Ea = abs(xr - xrold);
  test = func(xl,varargin{:})*func(xr,varargin{:});
  if test < 0
```

```
    xu = xr;
  elseif test > 0
    xl = xr;
  else
    ea = 0;
  end
end
root = xr;
```

The following is a MATLAB session that uses the function to solve Example 5.3 with $E_{a,d}$ = 0.0001.

```
>> format long
>> fcd=@(cd,m,t,v) m*9.81/cd*(1-exp(-cd/m*t))-v;
>> [root,Ea,ea,n] =bisect_iter(fcd,12,16,0.0001,68.1,10,40)

root =
  14.801086425781250
Ea =
    6.103515625000000e-005
ea =
    4.123694335281091e-004
n =
    16
```