CHAPTER 12

12.1 Reorder so that equations are diagonally dominant:

$$7x_1 - x_2 = 5$$
$$3x_1 + 8x_2 = 11$$

First iteration:

$$x_1 = \frac{5 + x_2}{7} = \frac{5 + 0}{7} = 0.714286$$
$$x_{1,r} = \lambda x_1 + (1-\lambda)x_1^{old} = 1.25(0.714286) - 0.25(0) = 0.892857$$
$$x_2 = \frac{11 - 3x_1}{8} = \frac{11 - 3(0.892857)}{8} = 1.040179$$
$$x_{2,r} = \lambda x_2 + (1-\lambda)x_2^{old} = 1.25(1.040179) - 0.25(0) = 1.300223$$

$$\varepsilon_{a,1} = \left|\frac{0.892857 - 0}{0.892857}\right| \times 100\% = 100\% \qquad \varepsilon_{a,2} = \left|\frac{1.300223 - 0}{1.300223}\right| \times 100\% = 100\%$$

Second iteration:

$$x_1 = \frac{5 + x_2}{7} = \frac{5 + 1.300223}{7} = 0.900032$$
$$x_{1,r} = \lambda x_1 + (1-\lambda)x_1^{old} = 1.25(0.900032) - 0.25(0.714286) = 0.901826$$
$$x_2 = \frac{11 - 3x_1}{8} = \frac{11 - 3(0.901826)}{8} = 1.036815$$
$$x_{2,r} = \lambda x_2 + (1-\lambda)x_2^{old} = 1.25(1.036815) - 0.25(1.300223) = 0.970963$$

$$\varepsilon_{a,1} = \left|\frac{0.901826 - 0.892857}{0.901826}\right| \times 100\% = 0.99\% \qquad \varepsilon_{a,2} = \left|\frac{0.970963 - 1.300223}{0.970963}\right| \times 100\% = 33.91\%$$

Third iteration:

$$x_1 = \frac{5 + x_2}{7} = \frac{5 + 0.970963}{7} = 0.852995$$
$$x_{1,r} = \lambda x_1 + (1-\lambda)x_1^{old} = 1.25(0.852995) - 0.25(0.901826) = 0.840787$$
$$x_2 = \frac{11 - 3x_1}{8} = \frac{11 - 3(0.840787)}{8} = 1.059705$$
$$x_{2,r} = \lambda x_2 + (1-\lambda)x_2^{old} = 1.25(1.059705) - 0.25(0.970963) = 1.08189$$

$$\varepsilon_{a,1} = \left|\frac{0.840787 - 0.901826}{0.840787}\right| \times 100\% = 7.26\% \qquad \varepsilon_{a,2} = \left|\frac{1.08189 - 0.970963}{1.08189}\right| \times 100\% = 10.25\%$$

The exact solution is $x_1 = 0.864407$ and $x_2 = 1.050847$ and the true errors after the third iteration are

$$\varepsilon_{t,1} = \left|\frac{0.864407 - 0.840787}{0.864407}\right| \times 100\% = 2.73\% \qquad \varepsilon_{t,2} = \left|\frac{1.050847 - 1.08189}{1.050847}\right| \times 100\% = 2.95\%$$

12.2 The first iteration can be implemented as

$$x_1 = \frac{41 + 0.4x_2}{0.8} = \frac{41 + 0.4(0)}{0.8} = 51.25$$
$$x_2 = \frac{25 + 0.4x_1 + 0.4x_3}{0.8} = \frac{25 + 0.4(51.25) + 0.4(0)}{0.8} = 56.875$$
$$x_3 = \frac{105 + 0.4x_2}{0.8} = \frac{105 + 0.4(56.875)}{0.8} = 159.6875$$

<u>Second iteration:</u>

$$x_1 = \frac{41 + 0.4(56.875)}{0.8} = 79.6875$$
$$x_2 = \frac{25 + 0.4(79.6875) + 0.4(159.6875)}{0.8} = 150.9375$$
$$x_3 = \frac{105 + 0.4(150.9375)}{0.8} = 206.7188$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left|\frac{79.6875 - 51.25}{79.6875}\right| \times 100\% = 35.69\%$$
$$\varepsilon_{a,2} = \left|\frac{150.9375 - 56.875}{150.9375}\right| \times 100\% = 62.32\%$$
$$\varepsilon_{a,3} = \left|\frac{206.7188 - 159.6875}{206.7188}\right| \times 100\% = 22.75\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	unknown	value	ε_a	maximum ε_a
1	x_1	51.25	100.00%	
	x_2	56.875	100.00%	
	x_3	159.6875	100.00%	100.00%
2	x_1	79.6875	35.69%	
	x_2	150.9375	62.32%	
	x_3	206.7188	22.75%	62.32%
3	x_1	126.7188	37.11%	
	x_2	197.9688	23.76%	
	x_3	230.2344	10.21%	37.11%
4	x_1	150.2344	15.65%	
	x_2	221.4844	10.62%	
	x_3	241.9922	4.86%	15.65%
5	x_1	161.9922	7.26%	
	x_2	233.2422	5.04%	
	x_3	247.8711	2.37%	7.26%
6	x_1	167.8711	3.50%	
	x_2	239.1211	2.46%	
	x_3	250.8105	1.17%	3.50%

Thus, after 6 iterations, the maximum error is 3.5% and we arrive at the result: $x_1 = 167.8711$, $x_2 = 239.1211$ and $x_3 = 250.8105$.

(b) The same computation can be developed with relaxation where $\lambda = 1.2$.

First iteration:

$$x_1 = \frac{41 + 0.4x_2}{0.8} = \frac{41 + 0.4(0)}{0.8} = 51.25$$

Relaxation yields: $x_1 = 1.2(51.25) - 0.2(0) = 61.5$

$$x_2 = \frac{25 + 0.4x_1 + 0.4x_3}{0.8} = \frac{25 + 0.4(61.5) + 0.4(0)}{0.8} = 62$$

Relaxation yields: $x_2 = 1.2(62) - 0.2(0) = 74.4$

$$x_3 = \frac{105 + 0.4x_2}{0.8} = \frac{105 + 0.4(74.4)}{0.8} = 168.45$$

Relaxation yields: $x_3 = 1.2(168.45) - 0.2(0) = 202.14$

Second iteration:

$$x_1 = \frac{41 + 0.4(74.4)}{0.8} = 88.45$$

Relaxation yields: $x_1 = 1.2(88.45) - 0.2(61.5) = 93.84$

$$x_2 = \frac{25 + 0.4(93.84) + 0.4(202.14)}{0.8} = 179.24$$

Relaxation yields: $x_2 = 1.2(179.24) - 0.2(74.4) = 200.208$

$$x_3 = \frac{105 + 0.4(200.208)}{0.8} = 231.354$$

Relaxation yields: $x_3 = 1.2(231.354) - 0.2(202.14) = 237.1968$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left| \frac{93.84 - 61.5}{93.84} \right| \times 100\% = 34.46\%$$

$$\varepsilon_{a,2} = \left| \frac{200.208 - 74.4}{200.208} \right| \times 100\% = 62.84\%$$

$$\varepsilon_{a,3} = \left| \frac{237.1968 - 202.14}{237.1968} \right| \times 100\% = 14.78\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	**unknown**	**value**	**relaxation**	ε_a	**maximum** ε_a
1	x_1	51.25	61.5	100.00%	
	x_2	62	74.4	100.00%	
	x_3	168.45	202.14	100.00%	100.000%
2	x_1	88.45	93.84	34.46%	
	x_2	179.24	200.208	62.84%	
	x_3	231.354	237.1968	14.78%	62.839%
3	x_1	151.354	162.8568	42.38%	
	x_2	231.2768	237.49056	15.70%	
	x_3	249.99528	252.55498	6.08%	42.379%
4	x_1	169.99528	171.42298	5.00%	
	x_2	243.23898	244.38866	2.82%	
	x_3	253.44433	253.6222	0.42%	4.997%

Thus, relaxation speeds up convergence. After 6 iterations, the maximum error is 4.997% and we arrive at the result: $x_1 = 171.423$, $x_2 = 244.389$ and $x_3 = 253.622$.

12.3 The first iteration can be implemented as

$$x_1 = \frac{27 - 2x_2 + x_3}{10} = \frac{27 - 2(0) + 0}{10} = 2.7$$
$$x_2 = \frac{-61.5 + 3x_1 - 2x_3}{-6} = \frac{-61.5 + 3(2.7) - 2(0)}{-6} = 8.9$$
$$x_3 = \frac{-21.5 - x_1 - x_2}{5} = \frac{-21.5 - (2.7) - 8.9}{5} = -6.62$$

Second iteration:

$$x_1 = \frac{27 - 2(8.9) - 6.62}{10} = 0.258$$
$$x_2 = \frac{-61.5 + 3(0.258) - 2(-6.62)}{-6} = 7.914333$$
$$x_3 = \frac{-21.5 - (0.258) - 7.914333}{5} = -5.934467$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left|\frac{0.258 - 2.7}{0.258}\right| \times 100\% = 947\%$$
$$\varepsilon_{a,2} = \left|\frac{7.914333 - 8.9}{7.914333}\right| \times 100\% = 12.45\%$$
$$\varepsilon_{a,3} = \left|\frac{-5.934467 - (-6.62)}{-5.934467}\right| \times 100\% = 11.55\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	unknown	value	ε_a	maximum ε_a
1	x_1	2.7	100.00%	
	x_2	8.9	100.00%	
	x_3	−6.62	100.00%	100%
2	x_1	0.258	946.51%	
	x_2	7.914333	12.45%	
	x_3	−5.93447	11.55%	947%
3	x_1	0.523687	50.73%	

	x_2	8.010001	1.19%	
	x_3	–6.00674	1.20%	50.73%
4	x_1	0.497326	5.30%	
	x_2	7.999091	0.14%	
	x_3	–5.99928	0.12%	5.30%
5	x_1	0.500253	0.59%	
	x_2	8.000112	0.01%	
	x_3	–6.00007	0.01%	0.59%

Thus, after 5 iterations, the maximum error is 0.59% and we arrive at the result: $x_1 = 0.500253$, $x_2 = 8.000112$ and $x_3 = -6.00007$.

12.4 The first iteration can be implemented as

$$x_1 = \frac{27 - 2x_2 + x_3}{10} = \frac{27 - 2(0) + 0}{10} = 2.7$$
$$x_2 = \frac{-61.5 + 3x_1 - 2x_3}{-6} = \frac{-61.5 + 3(0) - 2(0)}{-6} = 10.25$$
$$x_3 = \frac{-21.5 - x_1 - x_2}{5} = \frac{-21.5 - 0 - 0}{5} = -4.3$$

Second iteration:

$$x_1 = \frac{27 - 2(10.25) - 4.3}{10} = 0.22$$
$$x_2 = \frac{-61.5 + 3(2.7) - 2(-4.3)}{-6} = 7.466667$$
$$x_3 = \frac{-21.5 - (2.7) - 10.25}{5} = -6.89$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left|\frac{0.22 - 2.7}{0.22}\right| \times 100\% = 1127.3\%$$
$$\varepsilon_{a,2} = \left|\frac{7.466667 - 10.25}{7.466667}\right| \times 100\% = 37.28\%$$
$$\varepsilon_{a,3} = \left|\frac{-6.89 - (-4.3)}{-6.89}\right| \times 100\% = 37.59\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	unknown	value	ε_a	maximum ε_a
1	x_1	2.7	100.00%	
	x_2	10.25	100.00%	
	x_3	–4.3	100.00%	100.00%
2	x_1	0.22	1127.27%	
	x_2	7.466667	37.28%	
	x_3	–6.89	37.59%	1127.27%
3	x_1	0.517667	57.50%	
	x_2	7.843333	4.80%	
	x_3	–5.83733	18.03%	57.50%
4	x_1	0.5476	5.47%	
	x_2	8.045389	2.51%	
	x_3	–5.9722	2.26%	5.47%
5	x_1	0.493702	10.92%	
	x_2	7.985467	0.75%	
	x_3	–6.0186	0.77%	10.92%
6	x_1	0.501047	1.47%	
	x_2	7.99695	0.14%	
	x_3	–5.99583	0.38%	1.47%

Thus, after 6 iterations, the maximum error is 1.47% and we arrive at the result: $x_1 = 0.501047$, $x_2 = 7.99695$ and $x_3 = -5.99583$.

12.5 The first iteration can be implemented as

$$c_1 = \frac{3800 + 3c_2 + c_3}{15} = \frac{3800 + 3(0) + 0}{15} = 253.3333$$

$$c_2 = \frac{1200 + 3c_1 + 6c_3}{18} = \frac{1200 + 3(253.3333) + 6(0)}{18} = 108.8889$$

$$c_3 = \frac{2350 + 4c_1 + c_2}{12} = \frac{2350 + 4(253.3333) + 108.8889}{12} = 289.3519$$

Second iteration:

$$c_1 = \frac{3800 + 3(108.889) + 289.3519}{15} = 294.4012$$

$$c_2 = \frac{1200 + 3(294.4012) + 6(289.3519)}{18} = 212.1842$$

$$c_3 = \frac{2350 + 4(294.4012) + 212.1842}{12} = 311.6491$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left|\frac{294.4012 - 253.3333}{294.4012}\right| \times 100\% = 13.95\%$$
$$\varepsilon_{a,2} = \left|\frac{212.1842 - 108.8889}{212.1842}\right| \times 100\% = 48.68\%$$
$$\varepsilon_{a,3} = \left|\frac{311.6491 - 289.3519}{311.6491}\right| \times 100\% = 7.15\%$$

The remainder of the calculation can be summarized as

iteration	unknown	value	ε_a	maximum ε_a
1	c_1	253.3333	100.00%	
	c_2	108.8889	100.00%	
	c_3	289.3519	100.00%	100.00%
2	c_1	294.4012	13.95%	
	c_2	212.1842	48.68%	
	c_3	311.6491	7.15%	48.68%
3	c_1	316.5468	7.00%	
	c_2	223.3075	4.98%	
	c_3	319.9579	2.60%	7.00%
4	c_1	319.3254	0.87%	
	c_2	226.5402	1.43%	
	c_3	321.1535	0.37%	1.43%

Note that after several more iterations, we arrive at the result: c_1 = 320.2073, c_2 = 227.2021 and c_3 = 321.5026.

12.6 The equations must first be rearranged so that they are diagonally dominant:

$$-8x_1 + x_2 - 2x_3 = -20$$
$$2x_1 - 6x_2 - x_3 = -38$$
$$-3x_1 - x_2 + 7x_3 = -34$$

(a) The first iteration can be implemented as

$$x_1 = \frac{-20 - x_2 + 2x_3}{-8} = \frac{-20 - 0 + 2(0)}{-8} = 2.5$$
$$x_2 = \frac{-38 - 2x_1 + x_3}{-6} = \frac{-38 - 2(2.5) + 0}{-6} = 7.166667$$
$$x_3 = \frac{-34 + 3x_1 + x_2}{7} = \frac{-34 + 3(2.5) + 7.166667}{7} = -2.761905$$

Second iteration:

$$x_1 = \frac{-20 - 7.166667 + 2(-2.761905)}{-8} = 4.08631$$
$$x_2 = \frac{-38 - 2x_1 + x_3}{-6} = \frac{-38 - 2(4.08631) + (-2.761905)}{-6} = 8.155754$$
$$x_3 = \frac{-34 + 3x_1 + x_2}{7} = \frac{-34 + 3(4.08631) + 8.155754}{7} = -1.94076$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left|\frac{4.08631 - 2.5}{4.08631}\right| \times 100\% = 38.82\%$$
$$\varepsilon_{a,2} = \left|\frac{8.155754 - 7.166667}{8.155754}\right| \times 100\% = 12.13\%$$
$$\varepsilon_{a,3} = \left|\frac{-1.94076 - (-2.761905)}{-1.94076}\right| \times 100\% = 42.31\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	**unknown**	**value**	ε_a	**maximum** ε_a
0	x_1	0		
	x_2	0		
	x_3	0		
1	x_1	2.5	100.00%	
	x_2	7.166667	100.00%	
	x_3	–2.7619	100.00%	100.00%
2	x_1	4.08631	38.82%	
	x_2	8.155754	12.13%	
	x_3	–1.94076	42.31%	42.31%
3	x_1	4.004659	2.04%	
	x_2	7.99168	2.05%	
	x_3	–1.99919	2.92%	2.92%

Thus, after 3 iterations, the maximum error is 2.92% and we arrive at the result: $x_1 = 4.004659$, $x_2 = 7.99168$ and $x_3 = -1.99919$.

(b) The same computation can be developed with relaxation where $\lambda = 1.2$.

First iteration:

$$x_1 = \frac{-20 - x_2 + 2x_3}{-8} = \frac{-20 - 0 + 2(0)}{-8} = 2.5$$

Relaxation yields: $x_1 = 1.2(2.5) - 0.2(0) = 3$

$$x_2 = \frac{-38 - 2x_1 + x_3}{-6} = \frac{-38 - 2(3) + 0}{-6} = 7.333333$$

Relaxation yields: $x_2 = 1.2(7.333333) - 0.2(0) = 8.8$

$$x_3 = \frac{-34 + 3x_1 + x_2}{7} = \frac{-34 + 3(3) + 8.8}{7} = -2.3142857$$

Relaxation yields: $x_3 = 1.2(-2.3142857) - 0.2(0) = -2.7771429$

Second iteration:

$$x_1 = \frac{-20 - x_2 + 2x_3}{-8} = \frac{-20 - 8.8 + 2(-2.7771429)}{-8} = 4.2942857$$

Relaxation yields: $x_1 = 1.2(4.2942857) - 0.2(3) = 4.5531429$

$$x_2 = \frac{-38 - 2x_1 + x_3}{-6} = \frac{-38 - 2(4.5531429) - 2.7771429}{-6} = 8.3139048$$

Relaxation yields: $x_2 = 1.2(8.3139048) - 0.2(8.8) = 8.2166857$

$$x_3 = \frac{-34 + 3x_1 + x_2}{7} = \frac{-34 + 3(4.5531429) + 8.2166857}{7} = -1.7319837$$

Relaxation yields: $x_3 = 1.2(-1.7319837) - 0.2(-2.7771429) = -1.5229518$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left| \frac{4.5531429 - 3}{4.5531429} \right| \times 100\% = 34.11\%$$

$$\varepsilon_{a,2} = \left| \frac{8.2166857 - 8.8}{8.2166857} \right| \times 100\% = 7.1\%$$

$$\varepsilon_{a,3} = \left|\frac{-1.5229518-(-2.7771429)}{-1.5229518}\right| \times 100\% = 82.35\%$$

The remainder of the calculation proceeds until all the errors fall below the stopping criterion of 5%. The entire computation can be summarized as

iteration	unknown	value	relaxation	ε_a	maximum ε_a
1	x_1	2.5	3	100.00%	
	x_2	7.3333333	8.8	100.00%	
	x_3	–2.314286	–2.777143	100.00%	100.000%
2	x_1	4.2942857	4.5531429	34.11%	
	x_2	8.3139048	8.2166857	7.10%	
	x_3	–1.731984	–1.522952	82.35%	82.353%
3	x_1	3.9078237	3.7787598	20.49%	
	x_2	7.8467453	7.7727572	5.71%	
	x_3	–2.12728	–2.248146	32.26%	32.257%
4	x_1	4.0336312	4.0846055	7.49%	
	x_2	8.0695595	8.12892	4.38%	
	x_3	–1.945323	–1.884759	19.28%	19.280%
5	x_1	3.9873047	3.9678445	2.94%	
	x_2	7.9700747	7.9383056	2.40%	
	x_3	–2.022594	–2.050162	8.07%	8.068%
6	x_1	4.0048286	4.0122254	1.11%	
	x_2	8.0124354	8.0272613	1.11%	
	x_3	–1.990866	–1.979007	3.60%	3.595%

Thus, relaxation actually seems to retard convergence. After 6 iterations, the maximum error is 3.595% and we arrive at the result: $x_1 = 4.0122254$, $x_2 = 8.0272613$ and $x_3 = -1.979007$.

12.7 As ordered, none of the sets are guaranteed to converge. However, if Set 1 and 2 are reordered so that they are diagonally dominant, they will converge on the solution of (1, 1, 1).

Set 1:

$$\begin{aligned} 8x + 3y + z &= 13 \\ 2x + 5y - z &= 6 \\ -6x \qquad + 8z &= 2 \end{aligned}$$

Set 2:

$$\begin{aligned} 4x + 2y - 2z &= 4 \\ x + 5y - z &= 5 \\ x + y + 6z &= 8 \end{aligned}$$

Because it is not diagonally dominant, Set 3 will not converge on the correct solution of (1, 1, 1). For example, if they are ordered as

$$-2x + 2y - 3z = -3$$
$$2y - z = 1$$
$$-3x + 4y + 5z = 6$$

For this case, Gauss–Seidel iterations yields

iteration	unknown	value	ε_a	maximum ε_a
1	x_1	1.5	100.00%	100.00%
	x_2	0.5	100.00%	
	x_3	1.7	100.00%	
2	x_1	–0.55	372.73%	909.52%
	x_2	1.35	62.96%	
	x_3	–0.21	909.52%	
3	x_1	3.165	117.38%	241.77%
	x_2	0.395	241.77%	
	x_3	2.783	107.55%	
4	x_1	–2.2795	238.85%	265.57%
	x_2	1.8915	79.12%	
	x_3	–1.6809	265.57%	
5	x_1	5.91285	138.55%	655.59%
	x_2	–0.34045	655.59%	
	x_3	5.02007	133.48%	
6	x_1	–6.37056	192.82%	199.80%
	x_2	3.010035	111.31%	
	x_3	–5.03036	199.80%	
7	x_1	12.05558	152.84%	249.37%
	x_2	–2.01518	249.37%	
	x_3	10.04549	150.08%	
8	x_1	–15.5834	177.36%	179.93%
	x_2	5.522745	136.49%	
	x_3	–12.5682	179.93%	

Alternatively, they can be ordered as

$$-3x + 4y + 5z = 6$$
$$2y - z = 1$$
$$-2x + 2y - 3z = -3$$

For this case, Gauss–Seidel iterations yields

iteration	unknown	value	ε_a	maximum ε_a
1	x_1	−2	100.00%	100.00%
	x_2	0.5	100.00%	
	x_3	2.666667	100.00%	
2	x_1	3.111111	164.29%	1700.00%
	x_2	1.833333	72.73%	
	x_3	0.148148	1700.00%	
3	x_1	0.691358	350.00%	350.00%
	x_2	0.574074	219.35%	
	x_3	0.921811	83.93%	
4	x_1	0.301783	129.09%	129.09%
	x_2	0.960905	40.26%	
	x_3	1.439415	35.96%	
5	x_1	1.680232	82.04%	107.71%
	x_2	1.219707	21.22%	
	x_3	0.692984	107.71%	
6	x_1	0.781249	115.07%	115.07%
	x_2	0.846492	44.09%	
	x_3	1.043495	33.59%	
7	x_1	0.867814	9.98%	17.15%
	x_2	1.021747	17.15%	
	x_3	1.102622	5.36%	
8	x_1	1.200034	27.68%	27.68%
	x_2	1.051311	2.81%	
	x_3	0.900852	22.40%	

12.8 The equations to be solved are

$$f_1(x, y) = -x^2 + x + 0.75 - y$$

$$f_2(x, y) = x^2 - y - 5xy$$

The partial derivatives can be computed and evaluated at the initial guesses

$$\frac{\partial f_{1,0}}{\partial x} = -2x + 1 \; = -2(1.2) + 1 = -1.4 \qquad \frac{\partial f_{1,0}}{\partial y} = -1$$

$$\frac{\partial f_{2,0}}{\partial x} = 2x - 5y \; = 2(1.2) - 5(1.2) = -3.6 \qquad \frac{\partial f_{2,0}}{\partial y} = -1 - 5x = -1 - 5(1.2) = -7$$

These can then be used to compute the determinant of the Jacobian for the first iteration:

$$-1.4(-7)-(-1)(-3.6)=6.2$$

The values of the functions can be evaluated at the initial guesses as

$$f_{1,0} = -1.2^2 + 1.2 + 0.75 - 1.2 = -0.69$$

$$f_{2,0} = 1.2^2 - 1.2 - 5(1.2)1.2 = -6.96$$

These values can be substituted into Eq. (12.12) to give

$$x = 1.2 - \frac{-0.69(-7)-(-6.96)(-1)}{6.2} = 1.543548$$

$$y = 1.2 - \frac{-6.96(-1.4)-(-0.69)(-3.6)}{6.2} = 0.029032$$

The computation can be repeated until an acceptable accuracy is obtained. The results are summarized as

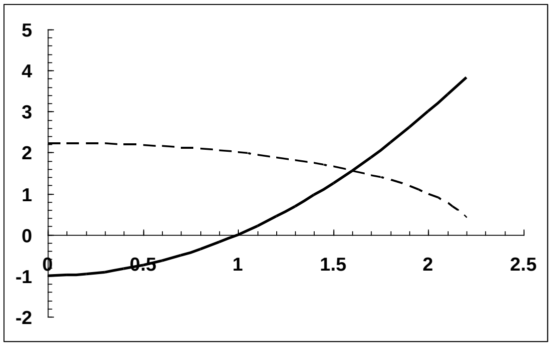
iteration	***x***	***y***	ε_{a1}	ε_{a2}
0	1.2	1.2		
1	1.543548	0.029032	22.257%	4033.333%
2	1.394123	0.222872	10.718%	86.974%
3	1.372455	0.239293	1.579%	6.862%
4	1.372066	0.239502	0.028%	0.087%

12.9 **(a)** The equations can be set up in a form amenable to plotting as

$$y = x^2 - 1$$

$$y = \sqrt{5 - x^2}$$

These can be plotted as



Thus, a solution seems to lie at about $x = y = 1.6$.

(b) The equations can be solved in a number of different ways. For example, the first equation can be solved for x and the second solved for y. For this case, successive substitution does not work

First iteration:

$$x = \sqrt{5 - y^2} = \sqrt{5 - (1.5)^2} = 1.658312$$
$$y = (1.658312)^2 - 1 = 1.75$$

Second iteration:

$$x = \sqrt{5 - (1.75)^2} = 1.391941$$
$$y = (1.391941)^2 - 1 = 0.9375$$

Third iteration:

$$x = \sqrt{5 - (0.9375)^2} = 2.030048$$
$$y = (2.030048)^2 - 1 = 3.12094$$

Thus, the solution is moving away from the solution that lies at approximately $x = y = 1.6$.

An alternative solution involves solving the second equation for x and the first for y. For this case, successive substitution does work

First iteration:

$$x = \sqrt{y + 1} = \sqrt{1.5 + 1} = 1.581139$$
$$y = \sqrt{5 - x^2} = \sqrt{5 - (1.581139)^2} = 1.581139$$

Second iteration:

$$x = \sqrt{1.581139} = 1.606592$$
$$y = \sqrt{5 - (1.606592)^2} = 1.555269$$

Third iteration:

$$x = \sqrt{5 - (1.555269)^2} = 1.598521$$
$$y = (1.598521)^2 - 1 = 1.563564$$

After several more iterations, the calculation converges on the solution of $x = 1.600485$ and $y = 1.561553$.

(c) The equations to be solved are

$$f_1(x, y) = x^2 - y - 1$$
$$f_2(x, y) = 5 - y^2 - x^2$$

The partial derivatives can be computed and evaluated at the initial guesses

$$\frac{\partial f_{1,0}}{\partial x} = 2x \qquad \frac{\partial f_{1,0}}{\partial y} = -1$$
$$\frac{\partial f_{2,0}}{\partial x} = -2x \qquad \frac{\partial f_{2,0}}{\partial y} = -2y$$

They can then be used to compute the determinant of the Jacobian for the first iteration is

$$3(-3) - (-1)(-3) = -12$$

The values of the functions can be evaluated at the initial guesses as

$$f_{1,0} = 1.5(1.5) - 1.5 - 1 = -0.25$$
$$f_{2,0} = 5 - 1.5(1.5) - 1.5(1.5) = 0.5$$

These values can be substituted into Eq. (12.12) to give

$$x = 1.5 - \frac{-0.25(-3) - 0.5(-1)}{-12} = 1.604167$$
$$y = 1.5 - \frac{0.5(3) - (-0.25)(-3)}{-12} = 1.5625$$

The computation can be repeated until an acceptable accuracy is obtained. The results are summarized as

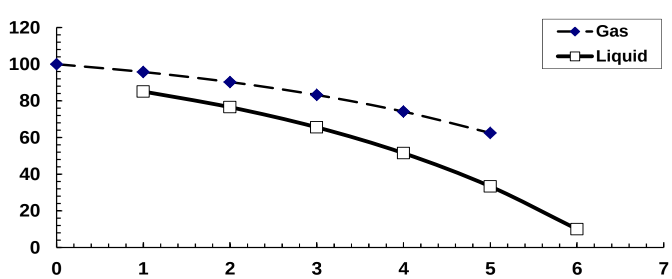
iteration	x	y	ε_{a1}	ε_{a2}
0	1.5	1.5		
1	1.604167	1.5625	6.494%	4.000%
2	1.600489	1.561553	0.230%	0.061%
3	1.600485	1.561553	0.000%	0.000%

12.10 The mass balances can be expressed in matrix form as

$$\begin{bmatrix} 2.8 & 0 & 0 & 0 & 0 & -0.8 & 0 & 0 & 0 & 0 \\ -2 & 2.8 & 0 & 0 & 0 & 0 & -0.8 & 0 & 0 & 0 \\ 0 & -2 & 2.8 & 0 & 0 & 0 & 0 & -0.8 & 0 & 0 \\ 0 & 0 & -2 & 2.8 & 0 & 0 & 0 & 0 & -0.8 & 0 \\ 0 & 0 & 0 & -2 & 2.8 & 0 & 0 & 0 & 0 & -0.8 \\ -0.8 & 0 & 0 & 0 & 0 & 1.8 & -1 & 0 & 0 & 0 \\ 0 & -0.8 & 0 & 0 & 0 & 0 & 1.8 & -1 & 0 & 0 \\ 0 & 0 & -0.8 & 0 & 0 & 0 & 0 & 1.8 & -1 & 0 \\ 0 & 0 & 0 & -0.8 & 0 & 0 & 0 & 0 & 1.8 & -1 \\ 0 & 0 & 0 & 0 & -0.8 & 0 & 0 & 0 & 0 & 1.8 \end{bmatrix} \begin{Bmatrix} c_{G1} \\ c_{G2} \\ c_{G3} \\ c_{G4} \\ c_{G5} \\ c_{L1} \\ c_{L2} \\ c_{L3} \\ c_{L4} \\ c_{L5} \end{Bmatrix} = \begin{Bmatrix} 200 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 10 \end{Bmatrix}$$

These equations can then be solved. The results are tabulated and plotted below:

Reactor	Gas	Liquid
0	100	
1	95.73328	85.06649
2	90.2475	76.53306
3	83.19436	65.5615
4	74.12603	51.45521
5	62.46675	33.31856
6		10



12.11 Substituting centered difference finite differences, the Laplace equation can be written for the node (1, 1) as

$$0=\frac{T_{21}-2T_{11}+T_{01}}{\Delta x^2}+\frac{T_{12}-2T_{11}+T_{10}}{\Delta y^2}$$

Because the grid is square ($\Delta x = \Delta y$), this equation can be expressed as

$$0=T_{21}-4T_{11}+T_{01}+T_{12}+T_{10}$$

The boundary node values ($T_{01} = 100$ and $T_{10} = 75$) can be substituted to give

$$4T_{11}-T_{12}-T_{21}=175$$

The same approach can be written for the other interior nodes. When this is done, the following system of equations results

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}\begin{Bmatrix} T_{11} \\ T_{12} \\ T_{21} \\ T_{22} \end{Bmatrix}=\begin{Bmatrix} 175 \\ 125 \\ 75 \\ 25 \end{Bmatrix}$$

These equations can be solved using the Gauss–Seidel method. For example, the first iteration would be

$$T_{11}=\frac{175+T_{12}+T_{21}}{4}=\frac{175+0+0}{4}=43.75$$
$$T_{12}=\frac{125+T_{11}+T_{22}}{4}=\frac{125+43.75+0}{4}=42.1875$$
$$T_{21}=\frac{75+T_{11}+T_{22}}{4}=\frac{75+43.75+0}{4}=29.6875$$
$$T_{22}=\frac{25+T_{12}+T_{21}}{4}=\frac{25+42.1875+29.6875}{4}=24.21875$$

The computation can be continued as follows:

iteration	**unknown**	**value**	ε_a	**maximum** ε_a
1	x_1	43.75	100.00%	
	x_2	42.1875	100.00%	
	x_3	29.6875	100.00%	
	x_4	24.21875	100.00%	100.00%

2	x_1	61.71875	29.11%	
	x_2	52.73438	20.00%	
	x_3	40.23438	26.21%	
	x_4	29.49219	17.88%	29.11%
3	x_1	66.99219	7.87%	
	x_2	55.37109	4.76%	
	x_3	42.87109	6.15%	
	x_4	30.81055	4.28%	7.87%
4	x_1	68.31055	1.93%	
	x_2	56.03027	1.18%	
	x_3	43.53027	1.51%	
	x_4	31.14014	1.06%	1.93%
5	x_1	68.64014	0.48%	
	x_2	56.19507	0.29%	
	x_3	43.69507	0.38%	
	x_4	31.22253	0.26%	0.48%

Thus, after 5 iterations, the maximum error is 0.48% and we are converging on the final result: $T_{11} = 68.75$, $T_{12} = 56.25$, $T_{21} = 43.75$, and $T_{22} = 31.25$.

12.12

```
function [x,ea,iter] = GaussSeidel(A,b,es,maxit)
% GaussSeidel: Gauss Seidel method
%   x = GaussSeidel(A,b): Gauss Seidel without relaxation
% input:
%   A = coefficient matrix
%   b = right hand side vector
%   es = stop criterion (default = 0.00001%)
%   maxit = max iterations (default = 50)
% output:
%   x = solution vector
%   ea = maximum relative error (%)
%   iter = number of iterations

if nargin<2,error('at least 2 input arguments required'),end
if nargin<4|isempty(maxit),maxit=50;end
if nargin<3|isempty(es),es=0.00001;end
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
C = A;
for i = 1:n
  C(i,i) = 0;
  x(i) = 0;
end
x = x';
for i = 1:n
  C(i,1:n) = C(i,1:n)/A(i,i);
end
for i = 1:n
  d(i) = b(i)/A(i,i);
```

```
end
iter = 0;
while (1)
  xold = x;
  for i = 1:n
    x(i) = d(i)-C(i,:)*x;
    if x(i) ~= 0
      ea(i) = abs((x(i) - xold(i))/x(i)) * 100;
    end
  end
  iter = iter+1;
  if max(ea)<=es | iter >= maxit, break, end
end
ea=max(ea);
```

Example 12.1:

```
>> clear,clc
>> A=[3 -.1 -.2;.1 7 -.3;.3 -.2 10];
>> b=[7.85;-19.3;71.4];
>> [x,ea,iter] = GaussSeidel(A,b)

x =
    3.0000
   -2.5000
    7.0000
ea =
  6.4417e-008
iter =
     6
```

Prob. 12.2*a*:

```
>> clear,clc
>> A=[0.8 -0.4 0;-0.4 0.8 -0.4;0 -0.4 0.8];
>> b=[41;25;105];
>> [x,ea,iter] = GaussSeidel(A,b)

x =
  173.7500
  245.0000
  253.7500
ea =
  6.4536e-006
iter =
    25
```

12.13

```
function [x,ea,iter] = GaussSeidelRelax(A,b,lambda,es,maxit)
% GaussSeidel: Gauss Seidel method with relaxation
%   x = GaussSeidel(A,b): Gauss Seidel with relaxation
% input:
%   A = coefficient matrix
```

```
%   b = right hand side vector
%   lambda = relation factor (default = 1)
%   es = stop criterion (default = 0.00001%)
%   maxit = max iterations (default = 50)
% output:
%   x = solution vector
%   ea = maximum relative error (%)
%   iter = number of iterations

if nargin<2,error('at least 2 input arguments required'),end
if nargin<5|isempty(maxit),maxit=50;end
if nargin<4|isempty(es),es=0.00001;end
if nargin<3|isempty(lambda),lambda=1;end
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
C = A;
for i = 1:n
  C(i,i) = 0;
  x(i) = 0;
end
x = x';
for i = 1:n
  C(i,1:n) = C(i,1:n)/A(i,i);
end
for i = 1:n
  d(i) = b(i)/A(i,i);
end
iter = 0;
while (1)
  xold = x;
  for i = 1:n
    x(i) = d(i)-C(i,:)*x;
    x(i) = lambda*x(i) + (1 - lambda)*xold(i);
    if x(i) ~= 0
      ea(i) = abs((x(i) - xold(i))/x(i)) * 100;
    end
  end
  iter = iter+1;
  if max(ea)<=es | iter >= maxit, break, end
end
ea=max(ea);
```

Example 12.2:

```
>> clear,clc
>> A=[10 -2;-3 12];
>> b=[8;9];
[x,ea,iter] = GaussSeidelRelax(A,b,1.2,[],2)
x =
    1.0531
    0.9783
ea =
   21.4307
iter =
     2
```

Prob. 12.2*b*:

```
>> clear,clc
>> A=[0.8 -0.4 0;-0.4 0.8 -0.4;0 -0.4 0.8];
>> b=[41;25;105];
>> [x,ea,iter] = GaussSeidelRelax(A,b,1.2)

x =
  173.7500
  245.0000
  253.7500
ea =
  6.1014e-006
iter =
    12
```

12.14

```
function [x,f,ea,iter]=newtmult(func,x0,es,maxit,varargin)
% newtmult: Newton-Raphson root zeroes nonlinear systems
% [x,f,ea,iter]=newtmult(func,x0,es,maxit,p1,p2,...):
%   uses the Newton-Raphson method to find the roots of
%   a system of nonlinear equations
% input:
%   func = name of function that returns f and J
%   x0 = initial guess
%   es = desired percent relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by function
% output:
%   x = vector of roots
%   f = vector of functions evaluated at roots
%   ea = approximate percent relative error (%)
%   iter = number of iterations

if nargin<2,error('at least 2 input arguments required'),end
if nargin<3|isempty(es),es=0.0001;end
if nargin<4|isempty(maxit),maxit=50;end
iter = 0;
x=x0;
while (1)
  [J,f]=func(x,varargin{:});
  dx=J\f;
  x=x-dx;
  iter = iter + 1;
  ea=100*max(abs(dx./x));
  if iter>=maxit|ea<=es, break, end
end

function [J,f]=jfreact2(x,varargin)
del=0.000001;
df1dx1=(u(x(1)+del*x(1),x(2))-u(x(1),x(2)))/(del*x(1));
df1dx2=(u(x(1),x(2)+del*x(2))-u(x(1),x(2)))/(del*x(2));
df2dx1=(v(x(1)+del*x(1),x(2))-v(x(1),x(2)))/(del*x(1));
df2dx2=(v(x(1),x(2)+del*x(2))-v(x(1),x(2)))/(del*x(2));
```

```
J=[df1dx1 df1dx2;df2dx1 df2dx2];
f1=u(x(1),x(2));
f2=v(x(1),x(2));
f=[f1;f2];
end
```

Example 12.4: The functions are set up as

```
function f=u(x,y)
f=x^2+x*y-10;
end

function f=v(x,y)
f=y+3*x*y^2-57;
end
```

<u>Script:</u>

```
clear,clc
x0=[1.5;3.5];
[x,f,ea,iter]=newtmult(@jfreact2,x0)
```

<u>Results:</u>

```
x =
    2.0000
    3.0000
f =
  1.0e-004 *

   -0.0129
   -0.2210
ea =

  1.9494e-005
iter =
     4
```

<u>Prob. 12.8:</u> The functions are set up as

```
function f=u(x,y)
f=-x^2+x+0.75-y;
end

function f=v(x,y)
f=x^2-y-5*x*y;
end
```

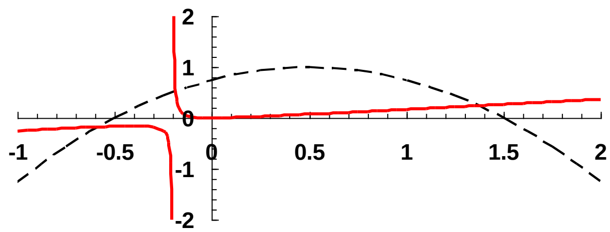
<u>Script:</u>

```
clear,clc
x0=[1.2;1.2];
[x,f,ea,iter]=newtmult(@jfreact2,x0)
```

Results:

```
x =
    1.3721
    0.2395
f =
   1.0e-06 *
   -0.1519
    0.5592
ea =
   2.0263e-05
iter =
     5
```

12.15 The functions can be plotted (y versus x). The plot indicates that there are three real roots at about (–0.6, –0.18), (–0.19, 0.6), and (1.37, 0.24).



(a) There are numerous ways to set this problem up as a fixed–point iteration. One way that converges is to solve the first equation for x and the second for y,

$$x = \sqrt{x + 0.75 - y}$$

$$y = \frac{x^2}{1 + 5x}$$

Using initial values of $x = y = 1.2$, the first iteration can be computed as:

$$x = \sqrt{1.2 + 0.75 - 1.2} = 0.866025$$
$$y = \frac{(0.866025)^2}{1 + 5(0.866025)} = 0.14071$$

<u>Second iteration:</u>

$$x = \sqrt{0.866025 + 0.75 - 0.14071} = 1.214626$$
$$y = \frac{(1.214626)^2}{1 + 5(1.214626)} = 0.20858$$

Third iteration:

$$x = \sqrt{1.214626 + 0.75 - 0.20858} = 1.325159$$

$$y = \frac{(1.325159)^2}{1 + 5(1.325159)} = 0.230277$$

Thus, the computation is converging on the root at $x = 1.372065$ and $y = 0.239502$.

Note that some other configurations are convergent and others are divergent. This exercise is intended to illustrate that although it may sometimes work, fixed–point iteration does not represent a practical general–purpose approach for solving systems of nonlinear equations.

(b) The equations to be solved are

$$f_1(x, y) = -x^2 + x + 0.75 - y$$

$$f_2(x, y) = x^2 - y - 5xy$$

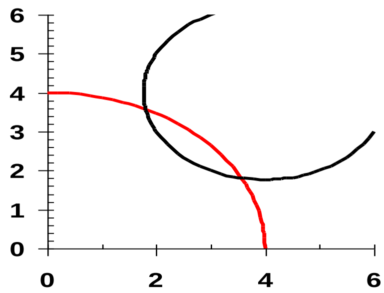
Script & Function:

```
clc, format compact
xguess=[1.2;1.2];
[x,fx] = fsolve(@fProb1215,xguess);
x
fx

function f = fProb1215(x)
f=[-x(1)*x(1)+x(1)+0.75-x(2);x(1)*x(1)-x(2)-5*x(1)*x(2)];
```

Results:

```
Equation solved.

fsolve completed because the vector of function values is near
zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
    1.3721
    0.2395
fx =
   1.0e-08 *
   -0.5484
   -0.2946
```

12.16 A plot of the functions indicates two real roots at about (2, 4) and (4, 2).



(a) The equations to be solved are

$$f_1(x, y) = (x - 4)^2 + (y - 4)^2 - 5$$

$$f_2(x, y) = x^2 + y^2 - 16$$

The partial derivatives can be computed and evaluated at initial guesses of (2, 4).

$$\frac{\partial f_{1,0}}{\partial x} = 2x - 8 \ = 2(2) - 8 = -4 \qquad \frac{\partial f_{1,0}}{\partial y} = 2y - 8 \ = 2(4) - 8 = 0$$

$$\frac{\partial f_{2,0}}{\partial x} = 2x = 2(2) = 4 \qquad \frac{\partial f_{2,0}}{\partial y} = 2y = 2(4) = 8$$

They can then be used to compute the determinant of the Jacobian for the first iteration is

$$-4(8) - 0(4) = -32$$

The values of the functions can be evaluated at the initial guesses as

$$f_{1,0} = (2 - 4)^2 + (4 - 4)^2 - 5 = -1$$

$$f_{2,0} = (2)^2 + (4)^2 - 16 = 4$$

These values can be substituted into Eq. (12.12) to give

$$x = 2 - \frac{-1(8) - 4(0)}{-4(8) - 0(4)} = 1.75$$

$$y = 4 - \frac{4(-4) - (-1)(4)}{-4(8) - 0(4)} = 3.625$$

The computation can be repeated until an acceptable accuracy is obtained. The results are summarized as

iteration	x	y	ε_{a1}	ε_{a2}
0	2	4		
1	1.75	3.625	14.286%	10.345%
2	1.804167	3.570833	3.002%	1.517%
3	1.805827	3.569173	0.092%	0.047%
4	1.805829	3.569171	0.000%	0.000%

Thus, for guesses of (2, 4), the result is (1.8059, 3.5691). Note that for guesses of (4, 2) the result is (3.5691, 1.8059).

(b) <u>Script & function:</u>

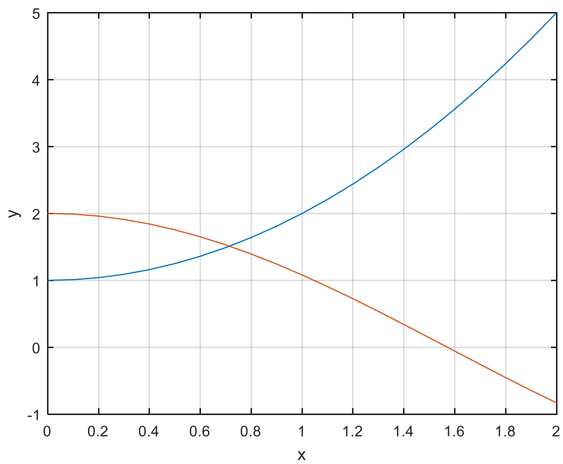
```
clc, format compact
xguess=[2;4];
[x,fx] = fsolve(@fProb1216,xguess);
x
fx

function f = fProb1216(x)
f=[(x(1)-4)*(x(1)-4)+ (x(2)-4)*(x(2)-4)-5;x(1)*x(1)+x(2)*x(2)-
16];
```

<u>Results:</u>

```
Equation solved.

fsolve completed because the vector of function values is near
zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
    1.8058
    3.5692
fx =
   1.0e-11 *
    0.4936
    0.4917
```

12.17 Graph can be generated with

```
clc, format compact
x=[0:.1:2];
y1=x.^2+1;
y2=2*cos(x);
plot(x,y1,x,y2)
xlabel('x');ylabel('y');grid
```



Root indicated near (0.7, 1.5).

(a) The equations to be solved are

$f_1(x, y) = x^2 + 1 - y$

$f_2(x, y) = 2\cos x - y$

The partial derivatives can be computed and evaluated at initial guesses of (0.7, 1.5).

$$\frac{\partial f_{1,0}}{\partial x} = 2x = 2(0.7) = 1.4 \qquad \frac{\partial f_{1,0}}{\partial y} = -1$$

$$\frac{\partial f_{2,0}}{\partial x} = -2\sin x = -2\sin(0.7) = -1.28844 \qquad \frac{\partial f_{2,0}}{\partial y} = -1$$

They can then be used to compute the determinant of the Jacobian for the first iteration is

$1.4(-1) - (-1)(-1.28844) = -2.68844$

The values of the functions can be evaluated at the initial guesses as

$f_{1,0} = 0.7^2 + 1 - 1.5 = -0.01$

$f_{2,0} = 2\cos(0.7) - 1.5 = 0.029684$

These values can be substituted into Eq. (12.12) to give

$$x = 0.7 - \frac{(-0.01)(-1) - 0.029684(-1)}{-2.68844} = 0.714761$$
$$y = 1.5 - \frac{0.029684(1.4) - (-0.01)(1.28844)}{-2.68844} = 1.510666$$

The computation can be repeated until an acceptable accuracy is obtained. The results are summarized as

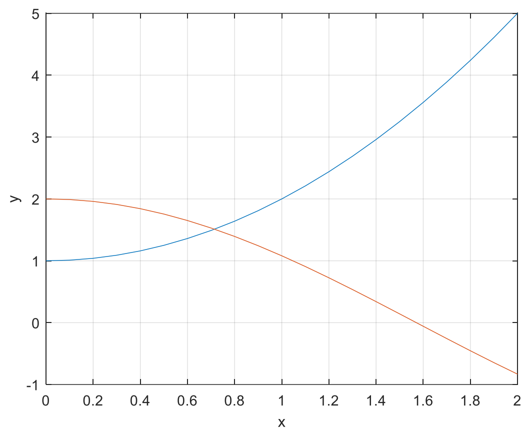
iteration	x	y	ε_{a1}	ε_{a2}
0	0.7	1.5		
1	0.714761	1.510666	2.07%	0.71%
2	0.714621	1.510683	0.02%	0.00%
3	0.714621	1.510683	0.00%	0.00%

(b) Script & function:

```
clc, format compact
x=[0:.1:2];
y1=x.^2+1;
y2=2*cos(x);
plot(x,y1,x,y2)
xlabel('x');ylabel('y');grid
xguess=[0.7;1.5];
[x,fx] = fsolve(@fProb1217,xguess);
x
fx

function f = fProb1217(x)
f=[x(1)^2+1-x(2);2*cos(x(1))-x(2)];
```

Results:

```
Equation solved.

fsolve completed because the vector of function values is near
zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
    0.7146
    1.5107
fx =
   1.0e-07 *
    0.1962
   -0.1482
```



12.18 (a) The functions to be solved are

$$K_1 = \frac{(c_{c,0} + x_1 + x_2)}{(c_{a,0} - 2x_1 - x_2)^2 (c_{b,0} - x_1)}$$

$$K_2 = \frac{(c_{c,0} + x_1 + x_2)}{(c_{a,0} - 2x_1 - x_2)(c_{d,0} - x_2)}$$

or

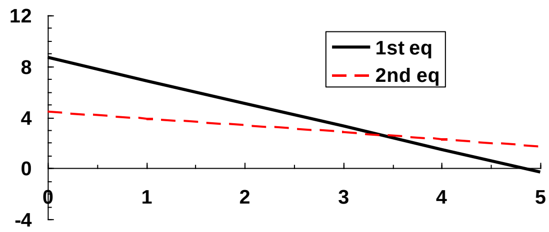
$$f_1(x_1, x_2) = \frac{5 + x_1 + x_2}{(50 - 2x_1 - x_2)^2 (20 - x_1)} - 4 \times 10^{-4}$$

$$f_2(x_1, x_2) = \frac{(5 + x_1 + x_2)}{(50 - 2x_1 - x_2)(10 - x_2)} - 3.7 \times 10^{-2}$$

Graphs can be generated by specifying values of x_1 and solving for x_2 using a numerical method like bisection.

first equation		second equation	
x_1	x_2	x_1	x_2
0	8.6672	0	4.4167
1	6.8618	1	3.9187
2	5.0649	2	3.4010
3	3.2769	3	2.8630
4	1.4984	4	2.3038
5	–0.2700	5	1.7227

These values can then be plotted to yield



Therefore, the root seems to be at about $x_1 = 3.3$ and $x_2 = 2.7$. Employing these values as the initial guesses for the two–variable Newton–Raphson method gives

$$f_1(3.3, 2.7) = -2.36\times10^{-6}$$

$$f_2(3.3, 2.7) = 2.33\times10^{-5}$$

$$\frac{\partial f_1}{\partial x_1} = 9.9\times10^{-5} \qquad \frac{\partial f_2}{\partial x_1} = 5.185\times10^{-3}$$

$$\frac{\partial f_1}{\partial x_2} = 5.57\times10^{-5} \qquad \frac{\partial f_2}{\partial x_2} = 9.35\times10^{-3}$$

$$|J| = 6.37\times10^{-7}$$

$$x_1 = 3.3 - \frac{-2.36\times10^{-6}(9.35\times10^{-3}) - 2.33\times10^{-5}(5.57\times10^{-5})}{6.37\times10^{-7}} = 3.3367$$

$$x_2 = 2.7 - \frac{2.33\times10^{-5}(9.9\times10^{-5}) - (-2.36\times10^{-6})(5.185\times10^{-3})}{6.37\times10^{-7}} = 2.677$$

The second iteration yields $x_1 = 3.3366$ and $x_2 = 2.677$, with a maximum approximate error of 0.003%.

(c) Script & function:

```
clc, format compact
xguess=[3.3;2.7];
[x,fx] = fsolve(@fProb1218,xguess);
x
fx

function f = fProb1218(x)
f=[(5+x(1)+x(2))/(50-2*x(1)-x(2))^2/(20-x(1))-4e-4;...
(5+x(1)+x(2))/(50-2*x(1)-x(2))/(10-x(2))-3.7e-2];
```

Results:

```
Equation solved at initial point.

fsolve completed because the vector of function values at the
initial point
is near zero as measured by the default value of the function
tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
    3.3000
    2.7000
fx =
   1.0e-04 *
   -0.0236
    0.2332
```

12.19 The function along with the suggested script are:

```
clc, format compact
xguess=[7;7;3;7;3];
options=optimset('tolfun',1e-12)
[x1,fx1] = fsolve(@funpH,xguess,options,315);
[x2,fx2] = fsolve(@funpH,xguess,options,400);
x1',fx1',x2',fx2'

function f = funpH(x,pCO2)
% 10^(-x(1))=H, 10^(-x(2))=OH, 10^(-x(3))=HCO3, 10^(-x(4))=CO3,
10^(-x(5))=cT
KH=10^(-1.46); K1=10^(-6.3);K2=10^(-10.3);Kw=10^(-14);
f = [10^6*10^(-x(1))*10^(-x(3))/(KH*pCO2)-K1;...
     10^(-x(1))*10^(-x(4))/10^(-x(3))-K2;...
     10^(-x(1))*10^(-x(2))-Kw;...
     (KH*pCO2)/1e6+10^(-x(3))+10^(-x(4))-10^(-x(5));...
     10^(-x(3))+2*10^(-x(4))+10^(-x(2))-10^(-x(1))];
```

Results:

```
Equation solved.
fsolve completed because the vector of function values is near
zero
as measured by the selected value of the function tolerance, and
the problem appears regular as measured by the gradient.
<stopping criteria details>

Equation solved.
fsolve completed because the vector of function values is near
zero
as measured by the selected value of the function tolerance, and
the problem appears regular as measured by the gradient.
<stopping criteria details>

ans =
```

```
    5.6304    8.3696    5.6312   10.3008    4.8774
ans =
   1.0e-08 *
    0.0000   -0.0000   -0.0000   -0.1069    0.0038
ans =
    5.5787    8.4213    5.5793   10.3006    4.7824
ans =
   1.0e-09 *
    0.0000   -0.0000   -0.0000   -0.7908    0.0157
```