

CHAPTER 1

1.1 For body weight:

$$4.5 + 4.5 + 12 + 4.5 + 33 + TW = 60$$

$$TW = 1.5\%$$

For total body water:

$$7.5 + 7.5 + 20 + 7.5 + 2.5 + IW = 100$$

$$IW = 55\%$$

1.2

$$Q_{\text{students}} = 30 \text{ ind} \times 80 \frac{\text{J}}{\text{ind s}} \times 15 \text{ min} \times 60 \frac{\text{s}}{\text{min}} \times \frac{\text{kJ}}{1000 \text{ J}} = 2160 \text{ kJ}$$

$$m = \frac{PVM_{\text{wt}}}{RT} = \frac{(101.325 \text{ kPa})(10\text{m} \times 8\text{m} \times 3\text{m} - 30 \times 0.075 \text{ m}^3)(28.97 \text{ kg/kmol})}{(8.314 \text{ kPa m}^3 /(\text{kmol K}))((20 + 273.15)\text{K})} = 286.3424 \text{ kg}$$

$$\Delta T = \frac{Q_{\text{students}}}{mC_v} = \frac{2160 \text{ kJ}}{(286.3424 \text{ kg})(0.718 \text{ kJ/(kg K)})} = 10.50615 \text{ K}$$

Therefore, the final temperature is $20 + 10.50615 = 30.50615^\circ\text{C}$.

1.3 This is a transient computation. For the period from ending June 1:

Balance = Previous Balance + Deposits – Withdrawals

$$\text{Balance} = 1512.33 + 220.13 - 327.26 = 1405.20$$

The balances for the remainder of the periods can be computed in a similar fashion as tabulated below:

Date	Deposit	Withdrawal	Balance
1-May			\$ 1512.33
	\$ 220.13	\$ 327.26	
1-Jun			\$ 1405.20
	\$ 216.80	\$ 378.61	
1-Jul			\$ 1243.39
	\$ 450.25	\$ 106.80	
1-Aug			\$ 1586.84
	\$ 127.31	\$ 350.61	
1-Sep			\$ 1363.54

$$1.4 \quad Q_{1,\text{in}} = Q_{2,\text{out}} + v_{3,\text{out}} A_3$$

$$A_3 = \frac{Q_{1,\text{in}} - Q_{2,\text{out}}}{v_{3,\text{out}}} = \frac{40 \text{ m}^3/\text{s} - 20 \text{ m}^3/\text{s}}{6 \text{ m/s}} = 3.333 \text{ m}^2$$

$$1.5 \quad \sum M_{\text{in}} - \sum M_{\text{out}} = 0$$

$$\text{Food} + \text{Drink} + \text{Air In} + \text{Metabolism} = \text{Urine} + \text{Skin} + \text{Feces} + \text{Air Out} + \text{Sweat}$$

$$\text{Drink} = \text{Urine} + \text{Skin} + \text{Feces} + \text{Air Out} + \text{Sweat} - \text{Food} - \text{Air In} - \text{Metabolism}$$

$$\text{Drink} = 1.4 + 0.35 + 0.2 + 0.4 + 0.2 - 1 - 0.05 - 0.3 = 1.2 \text{ L}$$

$$1.6 \quad v(t) = \frac{gm}{c}(1 - e^{-(c/m)t})$$

$$\text{jumper #1: } v(t) = \frac{9.8(70)}{12}(1 - e^{-(12/70)t}) = 46.8714$$

$$\text{jumper #2: } 46.8714 = \frac{9.8(75)}{15}(1 - e^{-(15/75)t})$$

$$46.8714 = 49 - 49e^{-0.2t}$$

$$0.04344 = e^{-0.2t}$$

$$\ln 0.04344 = -0.2t$$

$$t = \frac{\ln 0.04344}{-0.2} = 15.6818 \text{ s}$$

1.7 You are given the following differential equation with the initial condition, $v(t=0) = v(0)$,

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

The most efficient way to solve this is with Laplace transforms

$$sV(s) - v(0) = \frac{g}{s} - \frac{c}{m}V(s)$$

Solve algebraically for the transformed velocity

$$V(s) = \frac{v(0)}{s + c/m} + \frac{g}{s(s + c/m)} \quad (1)$$

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

The second term on the right of the equal sign can be expanded with partial fractions

$$\frac{g}{s(s + c/m)} = \frac{A}{s} + \frac{B}{s + c/m}$$

Combining the right-hand side gives

$$\frac{g}{s(s + c/m)} = \frac{A(s + c/m) + Bs}{s(s + c/m)}$$

By equating like terms in the numerator, the following must hold

$$g = A \frac{c}{m}$$

$$0 = As + Bs$$

The first equation can be solved for $A = mg/c$. According to the second equation, $B = -A$. Therefore, the partial fraction expansion is

$$\frac{g}{s(s + c/m)} = \frac{mg/c}{s} - \frac{mg/c}{s + c/m}$$

This can be substituted into Eq. 1 to give

$$V(s) = \frac{v(0)}{s + c/m} + \frac{mg/c}{s} - \frac{mg/c}{s + c/m}$$

Taking inverse Laplace transforms yields

$$v(t) = v(0)e^{-(c/m)t} + \frac{mg}{c} + \frac{mg}{c} e^{-(c/m)t}$$

or collecting terms

$$v(t) = v(0)e^{-(c/m)t} + \frac{mg}{c} \left(1 - e^{-(c/m)t}\right)$$

The first part is the general solution and the second part is the particular solution for the constant forcing function due to gravity.

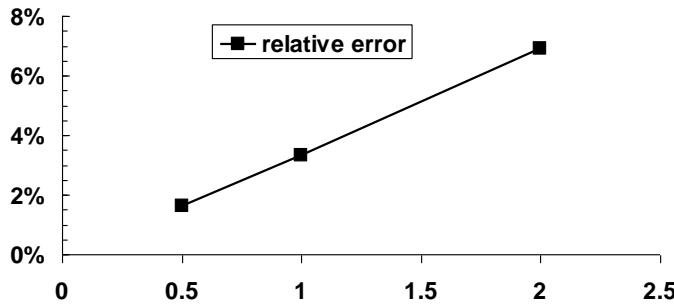
1.8 At $t = 10$ s, the analytical solution is 44.87 (Example 1.1). The relative error can be calculated with

$$\text{absolute relative error} = \left| \frac{\text{analytical} - \text{numerical}}{\text{analytical}} \right| \times 100\%$$

The numerical results are:

step	$v(10)$	absolute relative error
2	47.9690	6.90%
1	46.3639	3.32%
0.5	45.6044	1.63%

The error versus step size can then be plotted as



Thus, halving the step size approximately halves the error.

1.9 (a) You are given the following differential equation with the initial condition, $v(t = 0) = 0$,

$$\frac{dv}{dt} = g - \frac{c'}{m} v^2$$

Multiply both sides by m/c'

$$\frac{m}{c'} \frac{dv}{dt} = \frac{m}{c'} g - v^2$$

Define $a = \sqrt{mg / c'}$

$$\frac{m}{c'} \frac{dv}{dt} = a^2 - v^2$$

Integrate by separation of variables,

$$\int \frac{dv}{a^2 - v^2} = \int \frac{c'}{m} dt$$

A table of integrals can be consulted to find that

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$\int \frac{dx}{a^2 - x^2} = \frac{1}{a} \tanh^{-1} \frac{x}{a}$$

Therefore, the integration yields

$$\frac{1}{a} \tanh^{-1} \frac{v}{a} = \frac{c'}{m} t + C$$

If $v = 0$ at $t = 0$, then because $\tanh^{-1}(0) = 0$, the constant of integration $C = 0$ and the solution is

$$\frac{1}{a} \tanh^{-1} \frac{v}{a} = \frac{c'}{m} t$$

This result can then be rearranged to yield

$$v = \sqrt{\frac{gm}{c'}} \tanh\left(\sqrt{\frac{gc'}{m}} t\right)$$

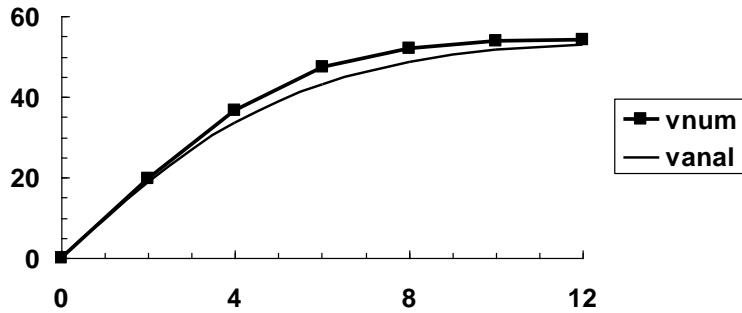
(b) Using Euler's method, the first two steps can be computed as

$$v(2) = 0 + \left[9.8 - \frac{0.225}{68.1} (0)^2 \right] 2 = 19.6$$

$$v(4) = 19.6 + \left[9.8 - \frac{0.225}{68.1} (19.6)^2 \right] 2 = 36.6615$$

The computation can be continued and the results summarized and plotted as:

<i>t</i>	<i>v</i>	<i>dv/dt</i>
0	0	9.8
2	19.6	8.53075
4	36.6615	5.35926
6	47.3800	2.38305
8	52.1461	0.81581
10	53.7777	0.24479
12	54.2673	0.07002
∞	54.4622	



Note that the analytical solution is included on the plot for comparison.

1.10 Before the chute opens ($t < 10$), Euler's method can be implemented as

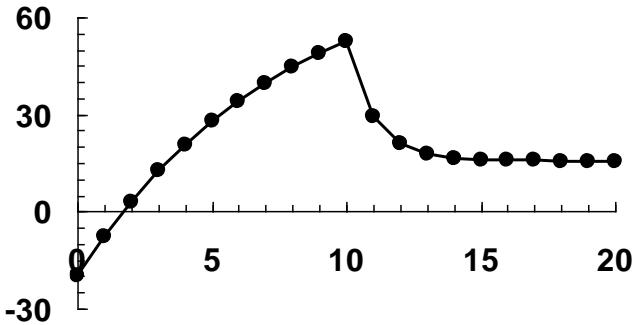
$$v(t + \Delta t) = v(t) + \left[9.8 - \frac{10}{80} v(t) \right] \Delta t$$

After the chute opens ($t \geq 10$), the drag coefficient is changed and the implementation becomes

$$v(t + \Delta t) = v(t) + \left[9.8 - \frac{50}{80} v(t) \right] \Delta t$$

Here is a summary of the results along with a plot:

Chute closed			Chute opened		
<i>t</i>	<i>v</i>	<i>dv/dt</i>	<i>t</i>	<i>v</i>	<i>dv/dt</i>
0	-20.0000	12.3000	10	52.5134	-23.0209
1	-7.7000	10.7625	11	29.4925	-8.6328
2	3.0625	9.4172	12	20.8597	-3.2373
3	12.4797	8.2400	13	17.6224	-1.2140
4	20.7197	7.2100	14	16.4084	-0.4552
5	27.9298	6.3088	15	15.9531	-0.1707
6	34.2385	5.5202	16	15.7824	-0.0640
7	39.7587	4.8302	17	15.7184	-0.0240
8	44.5889	4.2264	18	15.6944	-0.0090
9	48.8153	3.6981	19	15.6854	-0.0034
			20	15.6820	-0.0013



1.11 (a) The force balance can be written as:

$$m \frac{dv}{dt} = -mg(0) \frac{R^2}{(R+x)^2} + cv$$

Dividing by mass gives

$$\frac{dv}{dt} = -g(0) \frac{R^2}{(R+x)^2} + \frac{c}{m} v$$

(b) Recognizing that $dx/dt = v$, the chain rule is

$$\frac{dv}{dt} = v \frac{dv}{dx}$$

Setting drag to zero and substituting this relationship into the force balance gives

$$\frac{dv}{dx} = -\frac{g(0)}{v} \frac{R^2}{(R+x)^2}$$

(c) Using separation of variables

$$v dv = -g(0) \frac{R^2}{(R+x)^2} dx$$

Integrating gives

$$\frac{v^2}{2} = g(0) \frac{R^2}{R+x} + C$$

Applying the initial condition yields

$$\frac{v_0^2}{2} = g(0) \frac{R^2}{R+0} + C$$

which can be solved for $C = v_0^2/2 - g(0)R$, which can be substituted back into the solution to give

$$\frac{v^2}{2} = g(0) \frac{R^2}{R+x} + \frac{v_0^2}{2} - g(0)R$$

or

$$v = \pm \sqrt{v_0^2 + 2g(0) \frac{R^2}{R+x} - 2g(0)R}$$

Note that the plus sign holds when the object is moving upwards and the minus sign holds when it is falling.

(d) Euler's method can be developed as

$$v(x_{i+1}) = v(x_i) + \left[-\frac{g(0)}{v(x_i)} \frac{R^2}{(R+x_i)^2} \right] (x_{i+1} - x_i)$$

The first step can be computed as

$$v(10,000) = 1,400 + \left[-\frac{9.8}{1,400} \frac{(6.37 \times 10^6)^2}{(6.37 \times 10^6 + 0)^2} \right] (10,000 - 0) = 1,400 + (-0.007)10,000 = 1,330$$

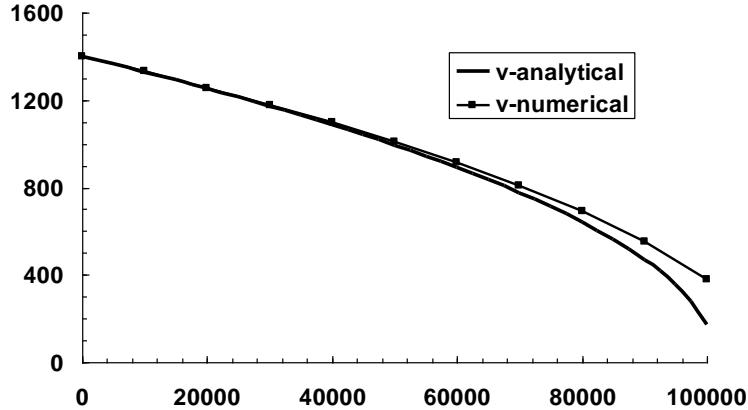
The remainder of the calculations can be implemented in a similar fashion as in the following table

x	v	dv/dx	v-analytical
0	1400.000	-0.00700	1400.000
10000	1330.000	-0.00735	1328.272
20000	1256.547	-0.00775	1252.688
30000	1179.042	-0.00823	1172.500
40000	1096.701	-0.00882	1086.688
50000	1008.454	-0.00957	993.796
60000	912.783	-0.01054	891.612
70000	807.413	-0.01188	776.473
80000	688.661	-0.01388	641.439
90000	549.864	-0.01733	469.650
100000	376.568	-0.02523	174.033

For the analytical solution, the value at 10,000 m can be computed as

$$v = \sqrt{1,400^2 + 2(9.8) \frac{(6.37 \times 10^6)^2}{(6.37 \times 10^6 + 10,000)^2} - 2(9.8)(6.37 \times 10^6)} = 1,328.272$$

The remainder of the analytical values can be implemented in a similar fashion as in the last column of the above table. The numerical and analytical solutions can be displayed graphically.



1.12 (a) The first two steps are

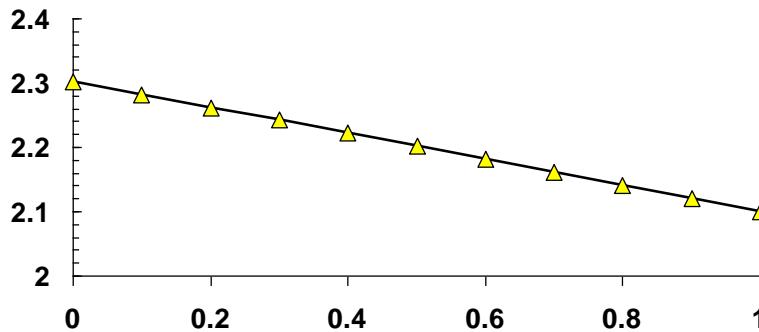
$$c(0.1) = 10 - 0.2(10)0.1 = 9.8 \text{ Bq/L}$$

$$c(0.2) = 9.8 - 0.2(9.8)0.1 = 9.604 \text{ Bq/L}$$

The process can be continued to yield

<i>t</i>	<i>c</i>	<i>dc/dt</i>
0	10.0000	-2.0000
0.1	9.8000	-1.9600
0.2	9.6040	-1.9208
0.3	9.4119	-1.8824
0.4	9.2237	-1.8447
0.5	9.0392	-1.8078
0.6	8.8584	-1.7717
0.7	8.6813	-1.7363
0.8	8.5076	-1.7015
0.9	8.3375	-1.6675
1	8.1707	-1.6341

(b) The results when plotted on a semi-log plot yields a straight line



The slope of this line can be estimated as

$$\frac{\ln(8.1707) - \ln(10)}{1} = -0.20203$$

Thus, the slope is approximately equal to the negative of the decay rate.

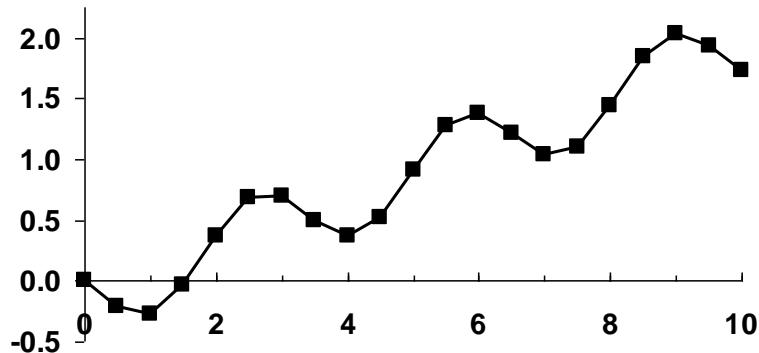
1.13 The first two steps yield

$$y(0.5) = 0 + \left[3 \frac{500}{1200} \sin^2(0) - \frac{500}{1200} \right] 0.5 = 0 + [0 - 0.41667] 0.5 = -0.20833$$

$$y(1) = -0.20833 + [\sin^2(0.5) - 0.41667] 0.5 = -0.27301$$

The process can be continued to give

<i>t</i>	<i>y</i>	<i>dy/dt</i>	<i>t</i>	<i>y</i>	<i>dy/dt</i>
0	0.00000	-0.41667	5.5	1.27629	0.20557
0.5	-0.20833	-0.12936	6	1.37907	-0.31908
1	-0.27301	0.46843	6.5	1.21953	-0.35882
1.5	-0.03880	0.82708	7	1.04012	0.12287
2	0.37474	0.61686	7.5	1.10156	0.68314
2.5	0.68317	0.03104	8	1.44313	0.80687
3	0.69869	-0.39177	8.5	1.84656	0.38031
3.5	0.50281	-0.26286	9	2.03672	-0.20436
4	0.37138	0.29927	9.5	1.93453	-0.40961
4.5	0.52101	0.77779	10	1.72973	-0.04672
5	0.90991	0.73275			



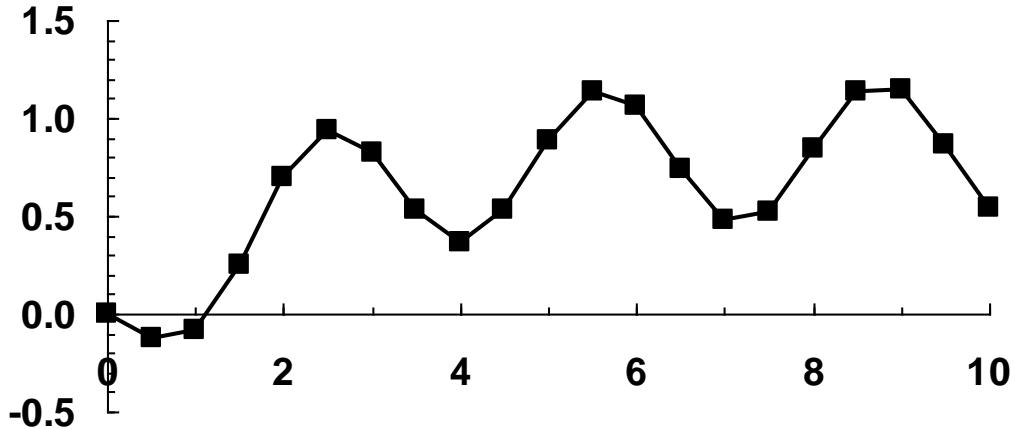
1.14 The first two steps yield

$$y(0.5) = 0 + \left[3 \frac{500}{1200} \sin^2(0) - \frac{300(1+0)^{1.5}}{1200} \right] 0.5 = 0 + [0 - 0.25] 0.5 = -0.125$$

$$y(1) = -0.125 + \left[\sin^2(0.5) - \frac{300(1-0.125)^{1.5}}{1200} \right] 0.5 = -0.08366$$

The process can be continued to give

t	y	<i>dy/dt</i>
0	0.00000	-0.25000
0.5	-0.12500	0.08269
1	-0.08366	0.66580
1.5	0.24924	0.89468
2	0.69658	0.48107
2.5	0.93711	-0.22631
3	0.82396	-0.59094
3.5	0.52849	-0.31862
4	0.36918	0.31541
4.5	0.52689	0.72277
5	0.88827	0.50073
5.5	1.13864	-0.15966
6	1.05881	-0.64093
6.5	0.73834	-0.51514
7	0.48077	0.08906
7.5	0.52530	0.62885
8	0.83973	0.59970
8.5	1.13958	0.01457
9	1.14687	-0.57411
9.5	0.85981	-0.62702
10	0.54630	-0.11076



1.15 The volume of the droplet is related to the radius as

$$V = \frac{4\pi r^3}{3} \quad (1)$$

This equation can be solved for radius as

$$r = \sqrt[3]{\frac{3V}{4\pi}} \quad (2)$$

The surface area is

$$A = 4\pi r^2 \quad (3)$$

Equation (2) can be substituted into Eq. (3) to express area as a function of volume

$$A = 4\pi \left(\frac{3V}{4\pi} \right)^{2/3}$$

This result can then be substituted into the original differential equation,

$$\frac{dV}{dt} = -k 4\pi \left(\frac{3V}{4\pi} \right)^{2/3} \quad (4)$$

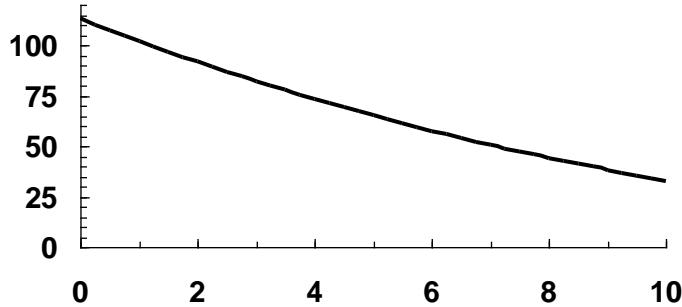
The initial volume can be computed with Eq. (1),

$$V = \frac{4\pi r^3}{3} = \frac{4\pi (3)^3}{3} = 113.0973 \text{ mm}^3$$

Euler's method can be used to integrate Eq. (4). Here are the beginning and last steps

<i>t</i>	<i>V</i>	<i>dV/dt</i>
0	113.0973	-11.3097
0.25	110.2699	-11.1204
0.5	107.4898	-10.9327
0.75	104.7566	-10.7466
1	102.07	-10.5621
•		
•		
•		
9	38.29357	-5.49416
9.25	36.92003	-5.36198
9.5	35.57954	-5.2314
9.75	34.27169	-5.1024
10	32.99609	-4.97499

A plot of the results is shown below:



Eq. (2) can be used to compute the final radius as

$$r = \sqrt[3]{\frac{3(32.99609)}{4\pi}} = 1.9897$$

Therefore, the average evaporation rate can be computed as

$$k = \frac{(3 - 1.9897) \text{ mm}}{10 \text{ min}} \frac{60 \text{ min}}{\text{hr}} = 0.10103 \frac{\text{mm}}{\text{min}}$$

which is approximately equal to the given evaporation rate.

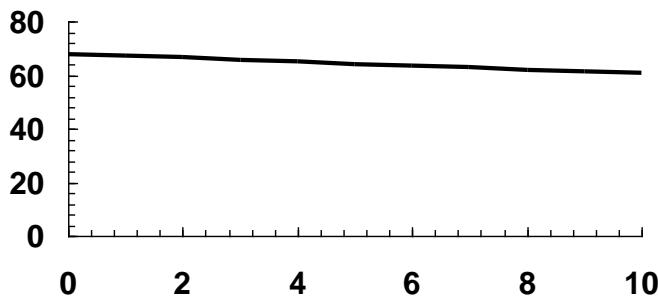
1.16 The first two steps can be computed as

$$T(1) = 68 + [-0.017(68 - 21)]1 = 68 + (-0.799)1 = 67.201$$

$$T(2) = 67.201 + [-0.017(67.201 - 21)]1 = 68 + (-0.78542)1 = 66.41558$$

The remaining results are displayed below along with a plot

<i>t</i>	<i>T</i>	<i>dT/dt</i>
0	68.00000	-0.79900
1	67.20100	-0.78542
2	66.41558	-0.77206
3	65.64352	-0.75894
4	64.88458	-0.74604
5	64.13854	-0.73336
6	63.40519	-0.72089
7	62.68430	-0.70863
8	61.97566	-0.69659
9	61.27908	-0.68474
10	60.59433	-0.67310



1.17 (a) The solution of the differential equation is

$$N = N_0 e^{\mu t}$$

The doubling time can be computed as the time when $N = 2N_0$,

$$2N_0 = N_0 e^{\mu(20)}$$

$$\mu = \frac{\ln 2}{20 \text{ hrs}} = \frac{0.693}{20 \text{ hrs}} = 0.034657/\text{hr}$$

(b) The volume of an individual spherical cell is

$$\text{cell volume} = \frac{\pi d^3}{6} \quad (1)$$

The total volume is

$$\text{volume} = \frac{\pi d^3}{6} N \quad (2)$$

The rate of change of N is defined as

$$\frac{dN}{dt} = \mu N \quad (3)$$

If $N = N_0$ at $t = 0$, Eq. 3 can be integrated to give

$$N = N_0 e^{\mu t} \quad (4)$$

Therefore, substituting (4) into (2) gives an equation for volume

$$\text{volume} = \frac{\pi d^3}{6} N_0 e^{\mu t} \quad (5)$$

(c) This equation can be solved for time

$$t = \frac{\ln \frac{6 \times \text{volume}}{\pi d^3 N_0}}{\mu} \quad (6)$$

The volume of a 500 μm diameter tumor can be computed with Eq. 2 as 65,449,847. Substituting this value along with $d = 20 \mu\text{m}$, $N_0 = 1$ and $\mu = 0.034657/\text{hr}$ gives

$$t = \frac{\ln \left(\frac{6 \times 65,449,847}{\pi 20^3 (1)} \right)}{0.034657} = 278.63 \text{ hr} = 11.6 \text{ d} \quad (6)$$

1.18 Continuity at the nodes can be used to determine the flows as follows:

$$Q_1 = Q_2 + Q_3 = 0.6 + 0.4 = 1 \frac{\text{m}^3}{\text{s}}$$

$$Q_{10} = Q_1 = 1 \frac{\text{m}^3}{\text{s}}$$

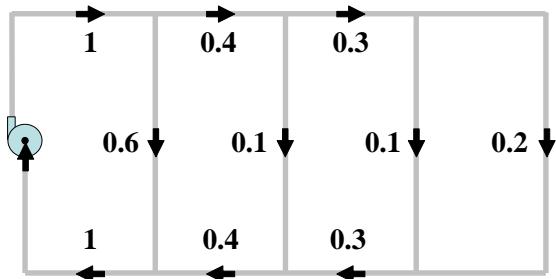
$$Q_9 = Q_{10} - Q_2 = 1 - 0.6 = 0.4 \frac{\text{m}^3}{\text{s}}$$

$$Q_4 = Q_9 - Q_8 = 0.4 - 0.3 = 0.1 \frac{\text{m}^3}{\text{s}}$$

$$Q_5 = Q_3 - Q_4 = 0.4 - 0.1 = 0.3 \frac{\text{m}^3}{\text{s}}$$

$$Q_6 = Q_5 - Q_7 = 0.3 - 0.2 = 0.1 \frac{\text{m}^3}{\text{s}}$$

Therefore, the final results are



CHAPTER 2

2.1 Two possible versions can be developed:

```

IF x ≥ 10 THEN
DO
    x = x - 5
    IF x < 50 EXIT
END DO
ELSE
    IF x < 5 THEN
        x = 5
    ELSE
        x = 7.5
    END IF
ENDIF

```

```

IF x ≥ 10 THEN
DO
    x = x - 5
    IF x < 50 EXIT
END DO
ELSEIF x < 5
    x = 5
ELSE
    x = 7.5
ENDIF

```

2.2

```

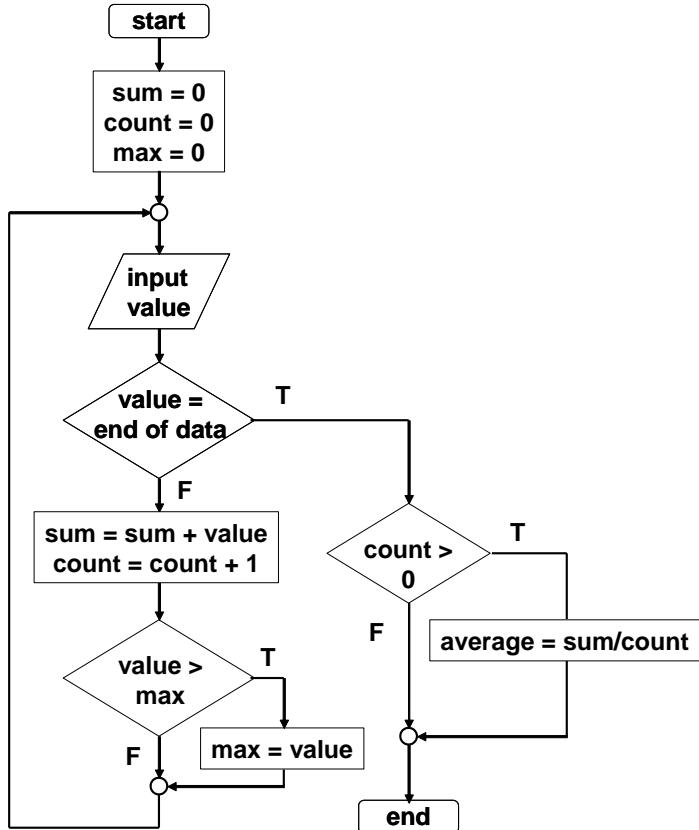
DO
    i = i + 1
    IF z > 50 EXIT
    x = x + 5
    IF x > 5 THEN
        y = x
    ELSE
        y = 0
    ENDIF
    z = x + y
ENDDO

```

2.3 Note that this algorithm is made simpler by recognizing that concentration cannot by definition be negative. Therefore, the maximum can be initialized as zero at the start of the algorithm.

- Step 1: Start
- Step 2: Initialize **sum**, **count** and **maximum** to zero
- Step 3: Examine top card.
- Step 4: If it says “end of data” proceed to step 9; otherwise, proceed to next step.
- Step 5: Add **value** from top card to **sum**.
- Step 6: Increase **count** by 1.
- Step 7: If **value** is greater than **maximum**, set **maximum** to **value**.
- Step 8: Discard top card
- Step 9: Is the **count** greater than zero?
 - If yes, proceed to step 10.
 - If no, proceed to step 11.
- Step 10: Calculate **average** = **sum/count**
- Step 11: End

2.4 Flowchart:



2.5 Students could implement the subprogram in any number of languages. The following Fortran 90 program is one example. It should be noted that the availability of complex variables in Fortran 90 would allow this subroutine to be made even more concise. However, we did not exploit this feature, in order to make the code more compatible with languages such as Visual BASIC or C.

```

PROGRAM Rootfind
IMPLICIT NONE
INTEGER::ier
REAL::a, b, c, r1, i1, r2, i2
DATA a,b,c/1.,6.,2./
CALL Roots(a, b, c, ier, r1, i1, r2, i2)
IF (ier == 0) THEN
  PRINT *, r1,i1," i"
  PRINT *, r2,i2," i"
ELSE
  PRINT *, "No roots"
END IF
END

SUBROUTINE Roots(a, b, c, ier, r1, i1, r2, i2)
IMPLICIT NONE
INTEGER::ier
REAL::a, b, c, d, r1, i1, r2, i2
r1=0.
r2=0.
  
```

```

i1=0.
i2=0.
IF (a == 0.) THEN
  IF (b /= 0) THEN
    r1 = -c/b
  ELSE
    ier = 1
  END IF
ELSE
  d = b**2 - 4.*a*c
  IF (d >= 0) THEN
    r1 = (-b + SQRT(d)) / (2*a)
    r2 = (-b - SQRT(d)) / (2*a)
  ELSE
    r1 = -b / (2*a)
    r2 = r1
    i1 = SQRT(ABS(d)) / (2*a)
    i2 = -i1
  END IF
END IF
END

```

The answers for the 3 test cases are: (a) $-0.3542, -5.646$; (b) 0.4 ; (c) $-0.4167 + 1.4696i$; $-0.4167 - 1.4696i$.

Several features of this subroutine bear mention:

- The subroutine does not involve input or output. Rather, information is passed in and out via the arguments. This is often the preferred style, because the I/O is left to the discretion of the programmer within the calling program.
- Note that an error code is passed (`IER = 1`) for the case where no roots are possible.

2.6 The development of the algorithm hinges on recognizing that the series approximation of the cosine can be represented concisely by the summation,

$$\sum_{i=1}^n (-1)^{i-1} \frac{x^{2i-2}}{(2i-2)!}$$

where i = the order of the approximation. The following algorithm implements this summation:

- Step 1: Start
- Step 2: Input value to be evaluated x and maximum order n
- Step 3: Set order (i) equal to one
- Step 4: Set accumulator for approximation (approx) to zero
- Step 5: Set accumulator for factorial product (factor) equal to one
- Step 6: Calculate true value of cos(x)
- Step 7: If order is greater than n then proceed to step 13
Otherwise, proceed to next step
- Step 8: Calculate the approximation with the formula

$$\text{approx} = \text{approx} + (-1)^{i-1} \frac{x^{2i-2}}{\text{factor}}$$

Step 9: Determine the error

$$\% \text{ error} = \left| \frac{\text{true} - \text{approx}}{\text{true}} \right| 100\%$$

Step 10: Increment the order by one

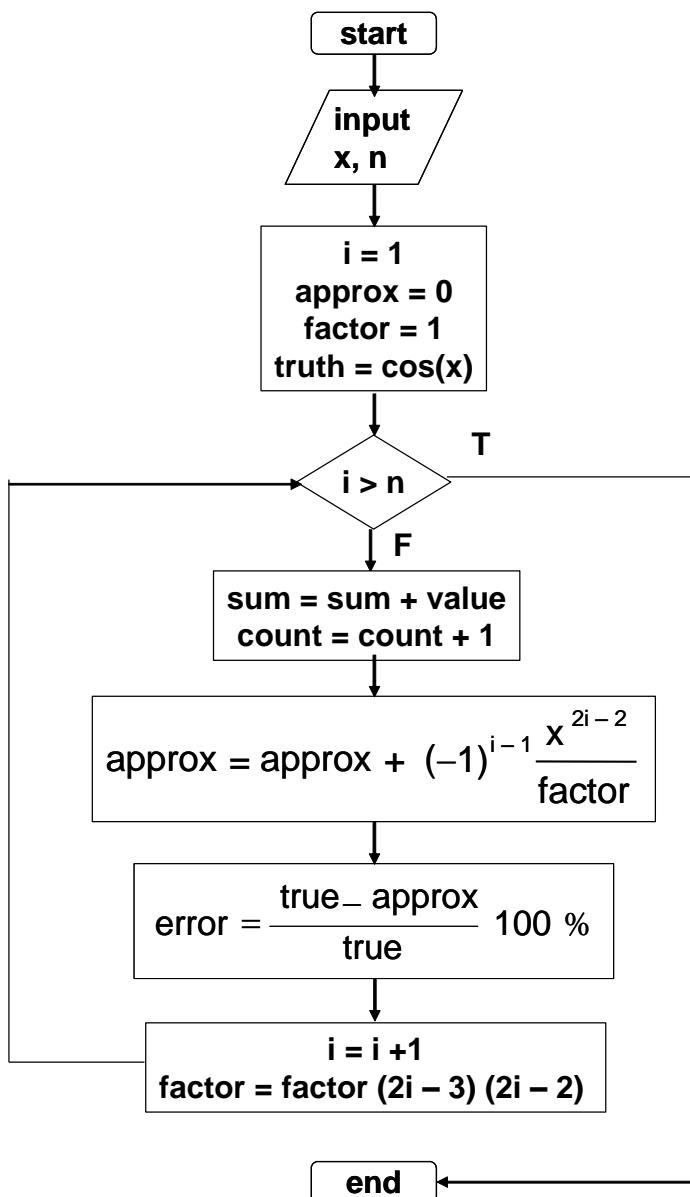
Step 11: Determine the factorial for the next iteration

$$\text{factor} = \text{factor} \cdot (2 \cdot i - 3) \cdot (2 \cdot i - 2)$$

Step 12: Return to step 7

Step 13: End

2.7 (a) Structured flowchart



(b) Pseudocode:

```

SUBROUTINE Coscomp (n,x)
i = 1
approx = 0
factor = 1
truth = cos (x)
DO
    IF i > n EXIT
    approx = approx + (-1)i-1•x2•i-2 / factor
    error = (true - approx) / true * 100
    DISPLAY i, true, approx, error
    i = i + 1
    factor = factor•(2•i-3) •(2•i-2)
END DO
END

```

2.8 Students could implement the subprogram in any number of languages. The following MATLAB M-file is one example. It should be noted that MATLAB allows direct calculation of the factorial through its intrinsic function `factorial`. However, we did not exploit this feature, in order to make the code more compatible with languages such as Visual BASIC and Fortran.

```

function coscomp(x,n)
i = 1;
tru = cos(x);
approx = 0;
f = 1;
fprintf('\n');
fprintf('order   true value      approximation      error\n');
while (1)
    if i > n, break, end
    approx = approx + (-1)^(i - 1) * x^(2*i-2) / f;
    er = (tru - approx) / tru * 100;
    fprintf('%3d  %14.10f  %14.10f  %12.8f\n',i,tru,approx,er);
    i = i + 1;
    f = f*(2*i-3)*(2*i-2);
end

```

Here is a run of the program showing the output that is generated:

```

>> coscomp(1.25, 6)

order   true value      approximation      error
1      0.3153223624    1.000000000000 -217.13576938
2      0.3153223624    0.218750000000  30.62655045
3      0.3153223624    0.3204752604    -1.63416828
4      0.3153223624    0.3151770698    0.04607749
5      0.3153223624    0.3153248988    -0.00080437
6      0.3153223624    0.3153223323    0.00000955

```

2.9 (a) The following pseudocode provides an algorithm for this problem. Notice that the input of the quizzes and homeworks is done with logical loops that terminate when the user enters a negative grade:

```

INPUT WQ, WH, WF
nq = 0
sumq = 0
DO
    INPUT quiz (enter negative to signal end of quizzes)
    IF quiz < 0 EXIT
    nq = nq + 1
    sumq = sumq + quiz
END DO
AQ = sumq / nq
nh = 0
sumh = 0
DO
    INPUT homework (enter negative to signal end of homeworks)
    IF homework < 0 EXIT
    nh = nh + 1
    sumh = sumh + homework
END DO
AH = sumh / nh
DISPLAY "Is there a final grade (y or n) "
INPUT answer
IF answer = "y" THEN
    INPUT FE
    AG = (WQ * AQ + WH * AH + WF * FE) / (WQ + WH + WF)
ELSE
    AG = (WQ * AQ + WH * AH) / (WQ + WH)
END IF
DISPLAY AG
END

```

(b) Students could implement the program in any number of languages. The following VBA code is one example.

```

Sub Grader()
Dim WQ As Double, WH As Double, WF As Double
Dim nq As Integer, sumq As Double, AQ As Double
Dim nh As Integer, sumh As Double, AH As Double
Dim answer As String, FE As Double
Dim AG As Double

'enter weights
WQ = InputBox("enter quiz weight")
WH = InputBox("enter homework weight")
WF = InputBox("enter final exam weight")
'enter quiz grades
nq = 0
sumq = 0
Do
    quiz = InputBox("enter negative to signal end of quizzes")
    If quiz < 0 Then Exit Do

```

```

nq = nq + 1
sumq = sumq + quiz
Loop
AQ = sumq / nq
'enter homework grades
nh = 0
sumh = 0
Do
    homework = InputBox("enter negative to signal end of homeworks")
    If homework < 0 Then Exit Do
    nh = nh + 1
    sumh = sumh + homework
Loop
AH = sumh / nh
'determine and display the average grade
answer = InputBox("Is there a final grade (y or n) ")
If answer = "y" Then
    FE = InputBox("final grade:")
    AG = (WQ * AQ + WH * AH + WF * FE) / (WQ + WH + WF)
Else
    AG = (WQ * AQ + WH * AH) / (WQ + WH)
End If
MsgBox "Average grade = " & AG
End Sub

```

The results should conform to:

$$AQ = 437/5 = 87.4$$

$$AH = 541/6 = 90.1667$$

without final

$$AG = \frac{35(87.4) + 30(90.1667)}{35 + 30} = 88.677$$

with final

$$AG = \frac{35(87.4) + 30(90.1667) + 35(92)}{35 + 30 + 35} = 89.84$$

2.10 (a) Pseudocode:

```

IF a > 0 THEN
    tol = 10^-5
    x = a/2
    DO
        y = (x + a/x)/2
        e = |(y - x)/y|
        x = y
        IF e < tol EXIT
    END DO
    SquareRoot = x
ELSE
    SquareRoot = 0
END IF

```

(b) Students could implement the function in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
<pre> Option Explicit Function SquareRoot(a) Dim x As Double, y As Double Dim e As Double, tol As Double If a > 0 Then tol = 0.00001 x = a / 2 Do y = (x + a / x) / 2 e = Abs((y - x) / y) x = y If e < tol Then Exit Do Loop SquareRoot = x Else SquareRoot = 0 End If End Function </pre>	<pre> function s = SquareRoot(a) if a > 0 tol = 0.00001; x = a / 2; while(1) y = (x + a / x) / 2; e = abs((y - x) / y); x = y; if e < tol, break, end end s = x; else s = 0; end </pre>

2.11 A MATLAB M-file can be written to solve this problem as

```

function futureworth(P, i, n)
nn = 0:n;
F = P*(1+i).^nn;
y = [nn;F];
fprintf('\n year future worth\n');
fprintf('%5d %14.2f\n',y);

```

This function can be used to evaluate the test case,

```

>> futureworth(100000,0.06,5)

year    future worth
0      100000.00
1      106000.00
2      112360.00
3      119101.60
4      126247.70
5      133822.56

```

2.12 A MATLAB M-file can be written to solve this problem as

```

function annualpayment(P, i, n)
nn = 1:n;
A = P*i*(1+i).^nn./((1+i).^nn-1);
y = [nn;A];
fprintf('\n year annual payment\n');
fprintf('%5d %14.2f\n',y);

```

This function can be used to evaluate the test case,

```
>> annualpayment(55000,0.066,5)

    year    annual payment
      1        58630.00
      2        30251.49
      3        20804.86
      4        16091.17
      5        13270.64
```

2.13 Students could implement the function in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
<pre>Option Explicit Function avgtemp(Tm, Tp, ts, te) Dim pi As Double, w As Double Dim Temp As Double, t As Double Dim sum As Double, i As Integer Dim n As Integer pi = 4 * Atan(1) w = 2 * pi / 365 sum = 0 n = 0 t = ts For i = ts To te Temp = Tm + (Tp - Tm) * Cos(w * (t - 205)) sum = sum + Temp n = n + 1 t = t + 1 Next i avgtemp = sum / n End Function</pre>	<pre>function Ta = avgtemp(Tm, Tp, ts, te) w = 2*pi/365; t = ts:te; T = Tm + (Tp-Tm)*cos(w*(t-205)); Ta = mean(T);</pre>

The function can be used to evaluate the test cases. The following show the results for MATLAB,

```
>> avgtemp(22.1,28.3,0,59)

ans =
16.2148

>> avgtemp(10.7,22.9,180,242)

ans =
22.2491
```

2.14 The programs are student specific and will be similar to the codes developed for VBA, MATLAB and Fortran as outlined in sections 2.4, 2.5 and 2.6. The numerical results for the different time steps are tabulated below along with an estimate of the absolute value of the true relative error at $t = 12$ s:

Step	$v(12)$	$ \varepsilon_t (\%)$
2	49.96	5.2
1	48.70	2.6
0.5	48.09	1.3

The general conclusion is that the error is halved when the step size is halved.

2.15 Students could implement the subprogram in any number of languages. The following Fortran 90 and VBA/Excel programs are two examples based on the algorithm outlined in Fig. P2.15.

Fortran 90	VBA/Excel
<pre> Subroutine BubbleFor(n, b) Implicit None !sorts an array in ascending !order using the bubble sort Integer(4)::m, i, n Logical::switch Real::a(n),b(n),dum m = n - 1 Do switch = .False. Do i = 1, m If (b(i) > b(i + 1)) Then dum = b(i) b(i) = b(i + 1) b(i + 1) = dum switch = .True. End If End Do If (switch == .False.) Exit m = m - 1 End Do End </pre>	<pre> Option Explicit Sub Bubble(n, b) 'sorts an array in ascending 'order using the bubble sort Dim m As Integer Dim i As Integer Dim switch As Boolean Dim dum As Double m = n - 1 Do switch = False For i = 1 To m If b(i) > b(i + 1) Then dum = b(i) b(i) = b(i + 1) b(i + 1) = dum switch = True End If Next i If switch = False Then Exit Do m = m - 1 Loop End Sub </pre>

For MATLAB, the following M-file implements the bubble sort following the algorithm outlined in Fig. P2.15:

```

function y = Bubble(x)
n = length(x);
m = n - 1;
b = x;
while(1)
    s = 0;
    for i = 1:m
        if b(i) > b(i + 1)
            dum = b(i);

```

```

b(i) = b(i + 1);
b(i + 1) = dum;
s = 1;
end
end
if s == 0, break, end
m = m - 1;
end
y = b;

```

Notice how the `length` function allows us to omit the length of the vector in the function argument. Here is an example MATLAB session that invokes the function to sort a vector:

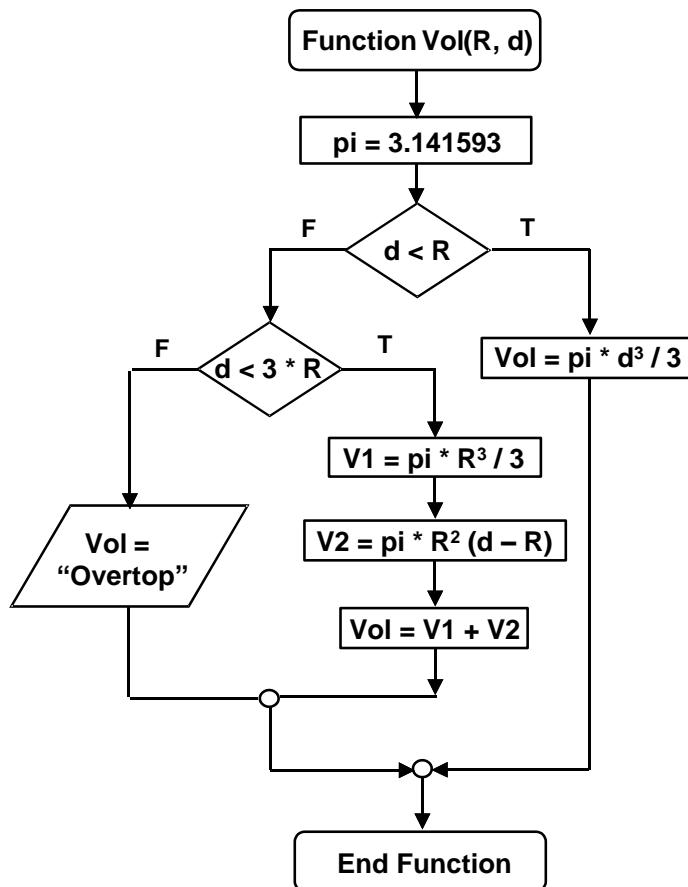
```

>> a=[3 4 2 8 5 7];
>> Bubble(a)

ans =
    2      3      4      5      7      8

```

2.16 Here is a flowchart for the algorithm:



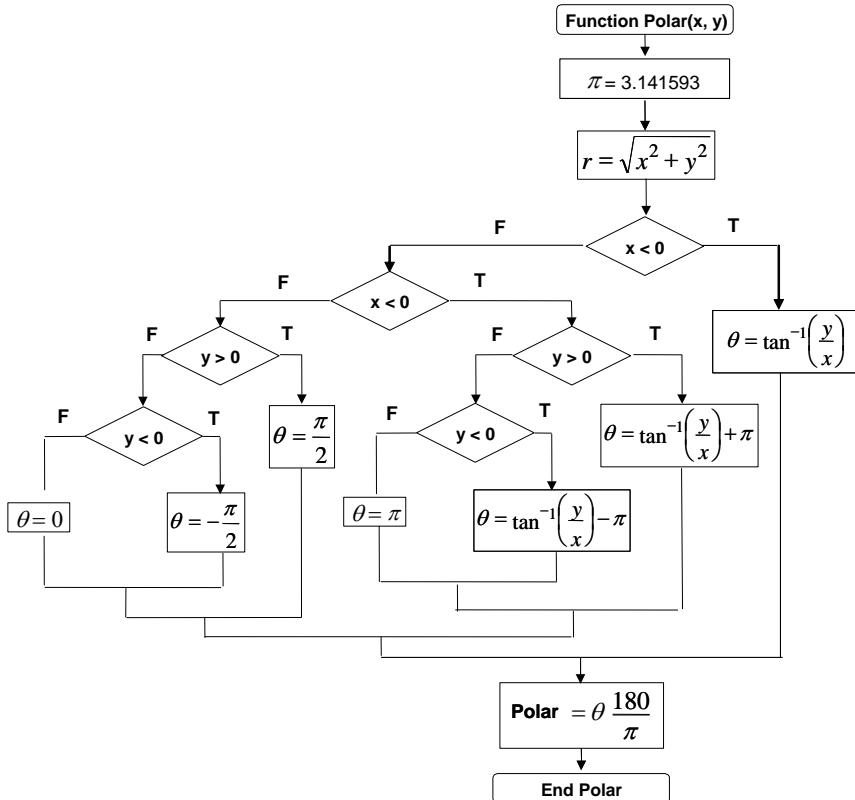
Students could implement the function in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
<pre> Option Explicit Function Vol(R, d) Dim V1 As Double, V2 As Double Dim pi As Double pi = 4 * Atn(1) If d < R Then Vol = pi * d ^ 3 / 3 ElseIf d <= 3 * R Then V1 = pi * R ^ 3 / 3 V2 = pi * R ^ 2 * (d - R) Vol = V1 + V2 Else Vol = "overtop" End If End Function </pre>	<pre> function Vol = tankvolume(R, d) if d < R Vol = pi * d ^ 3 / 3; elseif d <= 3 * R V1 = pi * R ^ 3 / 3; V2 = pi * R ^ 2 * (d - R); Vol = V1 + V2; else Vol = 'overtop'; end </pre>

The results are:

R	d	Volume
1	0.5	0.1309
1	1.2	1.675516
1	3	7.330383
1	3.1	overtop

2.17 Here is a flowchart for the algorithm:



Students could implement the function in any number of languages. The following MATLAB M-file is one option. Versions in other languages such as Fortran 90, Visual Basic, or C would have a similar structure.

```

function polar(x, y)
r = sqrt(x .^ 2 + y .^ 2);
n = length(x);
for i = 1:n
    if x(i) > 0
        th(i) = atan(y(i) / x(i));
    elseif x(i) < 0
        if y(i) > 0
            th(i) = atan(y(i) / x(i)) + pi;
        elseif y(i) < 0
            th(i) = atan(y(i) / x(i)) - pi;
        else
            th(i) = pi;
        end
    else
        if y(i) > 0
            th(i) = pi / 2;
        elseif y(i) < 0
            th(i) = -pi / 2;
        else
            th(i) = 0;
        end
    end
    th(i) = th(i) * 180 / pi;
end
ou=[x;y;r;th];
fprintf('\n      x          y          radius      angle\n');
fprintf('%8.2f %8.2f %10.4f %10.4f\n',ou);

```

This function can be used to evaluate the test cases.

```

>> x=[1 1 1 -1 -1 -1 0 0 0];
>> y=[1 -1 0 1 -1 0 1 -1 0];
>> polar(x,y)

      x          y          radius      angle
    1.00      1.00      1.4142      45.0000
    1.00     -1.00      1.4142     -45.0000
    1.00      0.00      1.0000      0.0000
   -1.00      1.00      1.4142     135.0000
   -1.00     -1.00      1.4142    -135.0000
   -1.00      0.00      1.0000    180.0000
    0.00      1.00      1.0000      90.0000
    0.00     -1.00      1.0000     -90.0000
    0.00      0.00      0.0000      0.0000

```

2.18 Students could implement the function in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
<pre>Function grade(s) If s >= 90 Then grade = "A" ElseIf s >= 80 Then grade = "B" ElseIf s >= 70 Then grade = "C" ElseIf s >= 60 Then grade = "D" Else grade = "F" End If End Function</pre>	<pre>function grade = lettergrade(score) if score >= 90 grade = 'A'; elseif score >= 80 grade = 'B'; elseif score >= 70 grade = 'C'; elseif score >= 60 grade = 'D'; else grade = 'F'; end</pre>

2.19 Students could implement the functions in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
(a) Factorial <pre>Function factor(n) Dim x As Long, i As Integer x = 1 For i = 1 To n x = x * i Next i factor = x End Function</pre>	<pre>function fout = factor(n) x = 1; for i = 1:n x = x * i; end fout = x;</pre>
(b) Minimum <pre>Function min(x, n) Dim i As Integer min = x(1) For i = 2 To n If x(i) < min Then min = x(i) Next i End Function</pre>	<pre>function xm = xmin(x) n = length(x); xm = x(1); for i = 2:n if x(i) < xm, xm = x(i); end end</pre>
(c) Average <pre>Function mean(x, n) Dim sum As Double Dim i As Integer sum = x(1) For i = 2 To n sum = sum + x(i) Next i mean = sum / n End Function</pre>	<pre>function xm = xmean(x) n = length(x); s = x(1); for i = 2:n s = s + x(i); end xm = s / n;</pre>

2.20 Students could implement the functions in any number of languages. The following VBA and MATLAB codes are two possible options.

VBA Function Procedure	MATLAB M-File
(a) Square root sum of squares	

```

Function SSS(x, n, m)
Dim i As Integer, j As Integer
SSS = 0
For i = 1 To n
    For j = 1 To m
        SSS = SSS + x(i, j) ^ 2
    Next j
Next i
SSS = Sqr(SSS)
End Function

```

(b) Normalization

```

Sub normal(x, n, m, y)
Dim i As Integer, j As Integer
Dim max As Double
For i = 1 To n
    max = Abs(x(i, 1))
    For j = 2 To m
        If Abs(x(i, j)) > max Then
            max = x(i, j)
        End If
    Next j
    For j = 1 To m
        y(i, j) = x(i, j) / max
    Next j
Next i
End Sub

```

```

function s = SSS(x)
[n,m] = size(x);
s = 0;
for i = 1:n
    for j = 1:m
        s = s + x(i, j) ^ 2;
    end
end
s = sqrt(s);

```

```

function y = normal(x)
[n,m] = size(x);
for i = 1:n
    mx = abs(x(i, 1));
    for j = 2:m
        if abs(x(i, j)) > mx
            mx = x(i, j);
        end
    end
    for j = 1:m
        y(i, j) = x(i, j) / mx;
    end
end

```

Alternate version:

```

function y = normal(x)
n = size(x);
for i = 1:n
    y(i,:) = x(i,:)/max(x(i,:));
end

```

CHAPTER 3

3.1 (a)

$$(101101)_2 = (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

$$= 32 + 8 + 4 + 1 = 45$$

(b)

$$(101.101)_2 = (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$

$$= 4 + 1 + 0.5 + 0.125 = 5.625$$

(c)

$$(0.01101)_2 = (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) + (0 \times 2^{-4}) + (1 \times 2^{-5})$$

$$= 0.25 + 0.125 + 0.03125 = 0.40625$$

3.2 Here are VBA and MATLAB implementations of the algorithm:

VBA Function Procedure	MATLAB M-File
<pre>Option Explicit Sub GetEps() Dim epsilon As Double epsilon = 1 Do If epsilon + 1 <= 1 Then Exit Do epsilon = epsilon / 2 Loop epsilon = 2 * epsilon MsgBox epsilon End Sub</pre>	<pre>function e = geteps e = 1; while(1) if e + 1 <= 1, break, end e = e / 2; end e = 2 * e;</pre>

Both routines yield a result of 2.22044604925031E–16 on my desktop PC. For single precision, the result is 1.192093E–07. Note that MATLAB has a built-in function `eps` that yields the same result.

3.3 Here are VBA and MATLAB implementations of the algorithm:

VBA Function Procedure	MATLAB M-File
<pre>Option Explicit Sub GetMin() Dim x As Double, xmin As Double x = 1 Do If x <= 0 Then Exit Do xmin = x x = x / 2 Loop MsgBox xmin End Sub</pre>	<pre>function xmin = getmin x = 1; while(1) if x <= 0, break, end xmin = x; x = x / 2; end</pre>

Both yield a result of 4.94065645841247E–324 on my desktop PC. For single precision, the result is 1.401298E–45.

3.4 Here is a VBA Program to compute the series in ascending order

```

Option Explicit

Sub SeriesForward()
Dim i As Integer, n As Integer
Dim sum As Single, pi As Double, truth As Double
pi = 4 * Atan(1)
truth = pi ^ 4 / 90
sum = 0
n = 10000
For i = 1 To n
    sum = sum + 1 / i ^ 4
Next i
MsgBox sum
'Display true percent relative error
MsgBox 100 * Abs((truth - sum) / truth)
End Sub

```

This yields a result of 1.0823221 with a true relative error of $1.02838 \times 10^{-4}\%$.

VBA Program to compute in descending order:

```

Option Explicit

Sub SeriesBackward()
Dim i As Integer, n As Integer
Dim sum As Single, pi As Double, truth As Double
pi = 4 * Atan(1)
truth = pi ^ 4 / 90
sum = 0
n = 10000
For i = n To 1 Step -1
    sum = sum + 1 / i ^ 4
Next i
MsgBox sum
'Display true percent relative error
MsgBox 100 * Abs((truth - sum) / truth)
End Sub

```

This yields a result of 1.0823232 with a true relative error of $3.71 \times 10^{-6}\%$.

The latter version yields a superior result because summing in descending order mitigates the roundoff error that occurs when adding a large and small number.

3.5 For the first series, after 20 terms are summed, the result is

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	$x^n/n!$	Sign	Series	True Value	et (%)	ea(%)
3	0	1	1	1	1.000000E+00	6.737947E-03	14741.32	
4	1	1	5	-1	-4.000000E+00	6.737947E-03	59465.26	125.0000000
5	2	2	12.5	1	8.500000E+00	6.737947E-03	126051.2	147.0588235
6	3	6	20.83333	-1	-1.233333E+01	6.737947E-03	183142.9	168.9189189
7	4	24	26.04167	1	1.370833E+01	6.737947E-03	203349.7	189.9696049
8	5	120	26.04167	-1	-1.233333E+01	6.737947E-03	183142.9	211.1486486
9	6	720	21.70139	1	9.368056E+00	6.737947E-03	138934.3	231.6530764
10	7	5040	15.50099	-1	-6.132937E+00	6.737947E-03	91120.85	252.7499191
11	8	40320	9.68812	1	3.555184E+00	6.737947E-03	52663.6	272.5068890
12	9	362880	5.382289	-1	-1.827105E+00	6.737947E-03	27216.65	294.5801032
13	10	3628800	2.691144	1	8.640391E-01	6.737947E-03	12723.48	311.4609662
14	11	39916800	1.223247	-1	-3.592084E-01	6.737947E-03	5431.125	340.5397724
15	12	4.79E+08	0.509686	1	1.504780E-01	6.737947E-03	2133.292	338.7115011
16	13	6.23E+09	0.196033	-1	-4.555520E-02	6.737947E-03	776.0992	430.3202153
17	14	8.72E+10	0.070012	1	2.445667E-02	6.737947E-03	262.9692	286.2690254
18	15	1.31E+12	0.023337	-1	-1.119380E-03	6.737947E-03	83.38693	2084.8412684
19	16	2.09E+13	0.007293	1	8.412283E-03	6.737947E-03	24.84936	86.6935085
20	17	3.56E+14	0.002145	-1	-6.267312E-03	6.737947E-03	6.984846	34.2247480
21	18	6.4E+15	0.000596	1	6.863137E-03	6.737947E-03	1.857988	8.6815321
22	19	1.22E+17	0.000157	-1	-6.706341E-03	6.737947E-03	0.469074	2.3380286
23	20	2.43E+18	3.92E-05	1	6.745540E-03	6.737947E-03	0.112692	0.5811105

The result oscillates at first. By $n = 20$ (21 terms), it is starting to converge on the true value. However, the relative error is still a substantial 0.11%. If carried out further to $n = 27$, the series eventually converges to within 7 significant digits.

In contrast the second series converges much faster. It attains 6 significant digits by $n = 20$ with a percent relative error of $8.1 \times 10^{-6}\%$.

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	$x^n/n!$	Series	1/Series	True Value	et (%)	ea(%)
3	0	1	1	1.000000E+00	1.000000E+00	6.737947E-03	1.47E+04	
4	1	1	5	6.000000E+00	1.666667E-01	6.737947E-03	2.37E+03	5.00E+02
5	2	2	12.5	1.850000E+01	5.405405E-02	6.737947E-03	7.02E+02	2.08E+02
6	3	6	20.83333	3.933333E+01	2.542373E-02	6.737947E-03	2.77E+02	1.13E+02
7	4	24	26.04167	6.537500E+01	1.529637E-02	6.737947E-03	1.27E+02	6.62E+01
8	5	120	26.04167	9.141667E+01	1.093892E-02	6.737947E-03	6.23E+01	3.98E+01
9	6	720	21.70139	1.131181E+02	8.840322E-03	6.737947E-03	3.12E+01	2.37E+01
10	7	5040	15.50099	1.286190E+02	7.774898E-03	6.737947E-03	1.54E+01	1.37E+01
11	8	40320	9.68812	1.383072E+02	7.230283E-03	6.737947E-03	7.31E+00	7.53E+00
12	9	362880	5.382289	1.436895E+02	6.959453E-03	6.737947E-03	3.29E+00	3.89E+00
13	10	3628800	2.691144	1.463806E+02	6.831506E-03	6.737947E-03	1.39E+00	1.87E+00
14	11	39916800	1.223247	1.476038E+02	6.774891E-03	6.737947E-03	5.48E-01	8.36E-01
15	12	4.79E+08	0.509686	1.481135E+02	6.751577E-03	6.737947E-03	2.02E-01	3.45E-01
16	13	6.23E+09	0.196033	1.483096E+02	6.742653E-03	6.737947E-03	6.98E-02	1.32E-01
17	14	8.72E+10	0.070012	1.483796E+02	6.739472E-03	6.737947E-03	2.26E-02	4.72E-02
18	15	1.31E+12	0.023337	1.484029E+02	6.738412E-03	6.737947E-03	6.90E-03	1.57E-02
19	16	2.09E+13	0.007293	1.484102E+02	6.738081E-03	6.737947E-03	1.99E-03	4.91E-03
20	17	3.56E+14	0.002145	1.484124E+02	6.737983E-03	6.737947E-03	5.42E-04	1.45E-03
21	18	6.4E+15	0.000596	1.484130E+02	6.737956E-03	6.737947E-03	1.40E-04	4.01E-04
22	19	1.22E+17	0.000157	1.484131E+02	6.737949E-03	6.737947E-03	3.45E-05	1.06E-04
23	20	2.43E+18	3.92E-05	1.484131E+02	6.737948E-03	6.737947E-03	8.11E-06	2.64E-05

3.6 The true value can be computed as

$$f'(0.577) = \frac{6(0.577)}{(1 - 3 \times 0.577^2)^2} = 2,352,911$$

Using 3-digits with chopping

$$\begin{aligned} 6x &= 6(0.577) = 3.462 \xrightarrow{\text{chopping}} 3.46 \\ x &= 0.577 \\ x^2 &= 0.332929 \xrightarrow{\text{chopping}} 0.332 \\ 3x^2 &= 0.996 \\ 1 - 3x^2 &= 0.004 \end{aligned}$$

$$f'(0.577) = \frac{3.46}{(1 - 0.996)^2} = \frac{3.46}{0.004^2} = 216,250$$

This represents a percent relative error of

$$\varepsilon_t = \left| \frac{2,352,911 - 216,250}{2,352,911} \right| = 90.8\%$$

Using 4-digits with chopping

$$\begin{aligned} 6x &= 6(0.577) = 3.462 \xrightarrow{\text{chopping}} 3.462 \\ x &= 0.577 \\ x^2 &= 0.332929 \xrightarrow{\text{chopping}} 0.3329 \\ 3x^2 &= 0.9987 \\ 1 - 3x^2 &= 0.0013 \end{aligned}$$

$$f'(0.577) = \frac{3.462}{(1 - 0.9987)^2} = \frac{3.462}{0.0013^2} = 2,048,521$$

This represents a percent relative error of

$$\varepsilon_t = \left| \frac{2,352,911 - 2,048,521}{2,352,911} \right| = 12.9\%$$

Although using more significant digits improves the estimate, the error is still considerable. The problem stems primarily from the fact that we are subtracting two nearly equal numbers in the denominator. Such subtractive cancellation is worsened by the fact that the denominator is squared.

3.7 First, the correct result can be calculated as

$$y = 1.37^3 - 7(1.37)^2 + 8(1.37) - 0.35 = 0.043053$$

(a) Using 3-digits with chopping

$$\begin{array}{rccccc}
 1.37^3 & \rightarrow & 2.571353 & \rightarrow & 2.57 \\
 -7(1.37)^2 & \rightarrow & -7(1.87) & \rightarrow & -13.0 \\
 8(1.37) & \rightarrow & 10.96 & \rightarrow & 10.9 \\
 & & & & -\frac{0.35}{0.12}
 \end{array}$$

This represents an error of

$$\varepsilon_t = \left| \frac{0.043053 - 0.12}{0.043053} \right| = 178.7\%$$

(b) Using 3-digits with chopping

$$y = ((1.37 - 7)1.37 + 8)1.37 - 0.35$$

$$y = (-5.63 \times 1.37 + 8)1.37 - 0.35$$

$$y = (-7.71 + 8)1.37 - 0.35$$

$$y = 0.29 \times 1.37 - 0.35$$

$$y = 0.397 - 0.35$$

$$y = 0.047$$

This represents an error of

$$\varepsilon_t = \left| \frac{0.043053 - 0.047}{0.043053} \right| = 9.2\%$$

Hence, the second form is superior because it tends to minimize round-off error.

3.8

$$20 \times 40 \times 120 = 96,000 \text{ words} @ 64 \text{ bits/word} = 8 \text{ bytes/word}$$

$$96,000 \text{ words} @ 8 \text{ bytes/word} = 768,000 \text{ bytes}$$

$$768,000 \text{ bytes} / 1024 \text{ bytes/kilobyte} = 750 \text{ kilobytes} = 0.75 \text{ Mbytes}$$

3.9 Here is a MATLAB M-file to solve the problem. Programs in other languages would have a similar structure and outcome.

```
% Given: Taylor Series Approximation for
% cos(x) = 1 - x^2/2! + x^4/4! - ...
% Find: number of terms needed to represent cos(x) to
% 8 significant figures at the point where: x = 0.3 pi
```

```
x = 0.3 * pi;
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

es = 0.5e-08;
%approximation
cosi = 1;
j = 1;
% j=terms counter
fprintf('j= %2.0f cos(x)= %0.10f\n', j,cosi)
fact = 1;
while(1)
    j = j + 1;
    i = 2 * j - 2;
    fact = fact * i * (i - 1);
    cosn = cosi + ((-1) ^ (j + 1)) * ((x) ^ i) / fact;
    ea = abs((cosn - cosi) / cosn);
    fprintf('j= %2.0f cos(x)= %0.10f ea = %0.1e\n',j,cosn,ea)
    if ea < es, break, end
    cosi = cosn;
end

j= 1 cos(x)= 1.0000000000
j= 2 cos(x)= 0.5558678020 ea = 8.0e-001
j= 3 cos(x)= 0.5887433702 ea = 5.6e-002
j= 4 cos(x)= 0.5877699636 ea = 1.7e-003
j= 5 cos(x)= 0.5877854037 ea = 2.6e-005
j= 6 cos(x)= 0.5877852513 ea = 2.6e-007
j= 7 cos(x)= 0.5877852523 ea = 1.7e-009

```

The true value of $\cos(0.3\pi)$ is 0.5877852525. Therefore, 6 terms of the Maclaurin series are necessary to approximate the true value to 8 significant figures.

3.10 First we can evaluate the exact values using the standard formula with double-precision arithmetic as

$$\frac{x_1 = \frac{5,000.002 \pm \sqrt{(5,000.002)^2 - 4(1)10}}{2(1)}}{0.002} = 5,000$$

We can then determine the square root term with 5-digit arithmetic and chopping

$$\begin{aligned} \sqrt{(5,000.0)^2 - 4(1)10} &= \sqrt{25,000,000 - 4(1)10} = \sqrt{24,999,960} \xrightarrow{\text{chopping}} \sqrt{24,999,000} \\ &= 4,999.996 \xrightarrow{\text{chopping}} 4,999.9 \end{aligned}$$

Equation (3.12):

$$x_1 = \frac{5,000.2 + 4,999.9}{2} = \frac{9,999.95}{2} \xrightarrow{\text{chopping}} \frac{9,999.9}{2} = 4,999.95 \xrightarrow{\text{chopping}} 4,999.9$$

$$x_2 = \frac{5,000.2 - 4,999.9}{2} = \frac{0.1}{2} = 0.05$$

Thus, although the first root is reasonably close to the true value ($\varepsilon_t = 0.002\%$), the second is considerably off ($\varepsilon_t = 2400\%$) due primarily to subtractive cancellation.

Equation (3.13):

$$x_1 = \frac{-2(10)}{-5,000.0 + 4,999.9} = \frac{-20}{-0.1} = 200$$

$$x_2 = \frac{-2(10)}{-5,000.0 - 4,999.9} = \frac{-20}{-9,999.9} = 0.002$$

For this case, the second root is well approximated, whereas the first is considerably off ($\varepsilon_t = 96\%$). Again, the culprit is the subtraction of two nearly equal numbers.

3.11 Remember that the machine epsilon is related to the number of significant digits by Eq. 3.11

$$\xi = b^{1-t}$$

which can be solved in base 10 for a machine epsilon of 1.19209×10^{-7} for

$$t = 1 - \log_{10}(\xi) = 1 - \log_{10}(1.19209 \times 10^{-7}) = 7.92$$

To be conservative, assume that 7 significant figures are good enough. Recall that Eq. 3.7 can then be used to estimate a stopping criterion,

$$\varepsilon_s = (0.5 \times 10^{2-n})\%$$

Thus, for 7 significant digits, the result would be

$$\varepsilon_s = (0.5 \times 10^{2-7})\% = 5 \times 10^{-6}\%$$

The total calculation can be expressed in one formula as

$$\varepsilon_s = (0.5 \times 10^{2-\text{Int}(1-\log_{10}(\xi))})\%$$

It should be noted that iterating to the machine precision is often overkill. Consequently, many applications use the old engineering rule of thumb that you should iterate to 3 significant digits or better.

As an application, I used Excel to evaluate the second series from Prob. 3.5. The results are:

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	$x^n/n!$	Series	1/Series	True Value	et (%)	ea(%)
3	0	1	1	1.000000E+00	1.000000E+00	6.737947E-03	1.47E+04	
4	1	1	5	6.000000E+00	1.666667E-01	6.737947E-03	2.37E+03	5.00E+02
5	2	2	12.5	1.850000E+01	5.405405E-02	6.737947E-03	7.02E+02	2.08E+02
6	3	6	20.83333	3.933333E+01	2.542373E-02	6.737947E-03	2.77E+02	1.13E+02
7	4	24	26.04167	6.537500E+01	1.529637E-02	6.737947E-03	1.27E+02	6.62E+01
8	5	120	26.04167	9.141667E+01	1.093892E-02	6.737947E-03	6.23E+01	3.98E+01
9	6	720	21.70139	1.131181E+02	8.840322E-03	6.737947E-03	3.12E+01	2.37E+01
10	7	5040	15.50099	1.286190E+02	7.774898E-03	6.737947E-03	1.54E+01	1.37E+01
11	8	40320	9.68812	1.383072E+02	7.230283E-03	6.737947E-03	7.31E+00	7.53E+00
12	9	362880	5.382289	1.436895E+02	6.959453E-03	6.737947E-03	3.29E+00	3.89E+00
13	10	3628800	2.691144	1.463806E+02	6.831506E-03	6.737947E-03	1.39E+00	1.87E+00
14	11	39916800	1.223247	1.476038E+02	6.774891E-03	6.737947E-03	5.48E-01	8.36E-01
15	12	4.79E+08	0.509686	1.481135E+02	6.751577E-03	6.737947E-03	2.02E-01	3.45E-01
16	13	6.23E+09	0.196033	1.483096E+02	6.742653E-03	6.737947E-03	6.98E-02	1.32E-01
17	14	8.72E+10	0.070012	1.483796E+02	6.739472E-03	6.737947E-03	2.26E-02	4.72E-02
18	15	1.31E+12	0.023337	1.484029E+02	6.738412E-03	6.737947E-03	6.90E-03	1.57E-02
19	16	2.09E+13	0.007293	1.484102E+02	6.738081E-03	6.737947E-03	1.99E-03	4.91E-03
20	17	3.56E+14	0.002145	1.484124E+02	6.737983E-03	6.737947E-03	5.42E-04	1.45E-03
21	18	6.4E+15	0.000596	1.484130E+02	6.737956E-03	6.737947E-03	1.40E-04	4.01E-04
22	19	1.22E+17	0.000157	1.484131E+02	6.737949E-03	6.737947E-03	3.45E-05	1.06E-04
23	20	2.43E+18	3.92E-05	1.484131E+02	6.737948E-03	6.737947E-03	8.11E-06	2.64E-05
24	21	5.11E+19	9.33E-06	1.484132E+02	6.737947E-03	6.737947E-03	1.82E-06	6.29E-06
25	22	1.12E+21	2.12E-06	1.484132E+02	6.737947E-03	6.737947E-03	3.91E-07	1.43E-06

Notice how after summing 21 terms, the result is correct to 7 significant figures. At this point, the true and the approximate percent relative errors are at $1.82 \times 10^{-6}\%$ and $6.29 \times 10^{-6}\%$, respectively. The process would repeat one more time so that the error estimates would fall below the precalculated stopping criterion of $5 \times 10^{-6}\%$.

CHAPTER 4

4.1 (a) For this case $x_i = 0$ and $h = x$. Thus,

$$f(x) = f(0) + f'(0)x + f''(0)\frac{x^2}{2} + \dots$$

$$f(0) = f'(0) = f''(0) = e^0 = 1$$

$$f(x) = 1 + x + \frac{x^2}{2} + \dots$$

(b)

$$f(x_{i+1}) = e^{-x_i} - e^{-x_i}h + e^{-x_i}\frac{h^2}{2} - e^{-x_i}\frac{h^3}{6} + \dots$$

for $x_i = 0.2$, $x_{i+1} = 1$ and $h = 0.8$. True value $= e^{-1} = 0.367879$.

zero order:

$$f(1) = e^{-0.2} = 0.818731$$

$$\varepsilon_t = \left| \frac{0.367879 - 0.818731}{0.367879} \right| \times 100\% = 122.55\%$$

first order:

$$f(1) = 0.818731 - 0.818731(0.8) = 0.163746$$

$$\varepsilon_t = \left| \frac{0.367879 - 0.163746}{0.367879} \right| \times 100\% = 55.49\%$$

second order:

$$f(1) = 0.818731 - 0.818731(0.8) + 0.818731\frac{0.8^2}{2} = 0.42574$$

$$\varepsilon_t = \left| \frac{0.367879 - 0.42574}{0.367879} \right| \times 100\% = 15.73\%$$

third order:

$$f(1) = 0.818731 - 0.818731(0.8) + 0.818731\frac{0.8^2}{2} - 0.818731\frac{0.8^3}{6} = 0.355875$$

$$\varepsilon_t = \left| \frac{0.367879 - 0.355875}{0.367879} \right| \times 100\% = 3.26\%$$

4.2 Use the stopping criterion

$$\varepsilon_s = 0.5 \times 10^{-2} \% = 0.5\%$$

True value: $\cos(\pi/3) = 0.5$

zero order:

$$\cos\left(\frac{\pi}{3}\right) = 1$$

$$\varepsilon_t = \left| \frac{0.5 - 1}{0.5} \right| \times 100\% = 100\%$$

first order:

$$\cos\left(\frac{\pi}{3}\right) = 1 - \frac{(\pi/3)^2}{2} = 0.451689$$

$$\varepsilon_t = 9.66\% \quad \varepsilon_a = \left| \frac{0.451689 - 1}{0.451689} \right| \times 100\% = 121.4\%$$

second order:

$$\cos\left(\frac{\pi}{3}\right) = 0.451689 + \frac{(\pi/3)^4}{24} = 0.501796$$

$$\varepsilon_t = 0.359\% \quad \varepsilon_a = \left| \frac{0.501796 - 0.451689}{0.501796} \right| \times 100\% = 9.986\%$$

third order:

$$\cos\left(\frac{\pi}{3}\right) = 0.501796 - \frac{(\pi/3)^6}{720} = 0.499965$$

$$\varepsilon_t = 0.00709\% \quad \varepsilon_a = \left| \frac{0.499965 - 0.501796}{0.499965} \right| \times 100\% = 0.366\%$$

Since the approximate error is below 0.5%, the computation can be terminated.

4.3 Use the stopping criterion: $\varepsilon_s = 0.5 \times 10^{-2} \% = 0.5\%$

True value: $\sin(\pi/3) = 0.866025\dots$

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

zero order:

$$\sin\left(\frac{\pi}{3}\right) = \frac{\pi}{3} = 1.047198$$

$$\varepsilon_t = \left| \frac{0.866025 - 1.047198}{0.866025} \right| \times 100\% = 20.92\%$$

first order:

$$\sin\left(\frac{\pi}{3}\right) = 1.047198 - \frac{(\pi/3)^3}{6} = 0.855801$$

$$\varepsilon_t = 1.18\% \quad \varepsilon_a = \left| \frac{0.855801 - 1.047198}{0.855801} \right| \times 100\% = 22.36\%$$

second order:

$$\sin\left(\frac{\pi}{3}\right) = 0.855801 + \frac{(\pi/3)^5}{120} = 0.866295$$

$$\varepsilon_t = 0.031\% \quad \varepsilon_a = \left| \frac{0.866295 - 0.855801}{0.866295} \right| \times 100\% = 1.211\%$$

third order:

$$\sin\left(\frac{\pi}{3}\right) = 0.866295 - \frac{(\pi/3)^7}{5040} = 0.866021$$

$$\varepsilon_t = 0.000477\% \quad \varepsilon_a = \left| \frac{0.866021 - 0.866295}{0.866021} \right| \times 100\% = 0.0316\%$$

Since the approximate error is below 0.5%, the computation can be terminated.

4.4 True value: $f(3) = 554$.

zero order:

$$f(3) = f(1) = -62$$

$$\varepsilon_t = \left| \frac{554 - (-62)}{554} \right| \times 100\% = 111.191\%$$

first order:

$$f(3) = -62 + f'(1)(3-1) = -62 + 70(2) = 78 \quad \varepsilon_t = 85.921\%$$

second order:

$$f(3) = 78 + \frac{f''(1)}{2}(3-1)^2 = 78 + \frac{138}{2}4 = 354 \quad \varepsilon_t = 36.101\%$$

third order:

$$f(3) = 354 + \frac{f^{(3)}(1)}{6}(3-1)^3 = 354 + \frac{150}{6}8 = 554 \quad \varepsilon_t = 0\%$$

Thus, the third-order result is perfect because the original function is a third-order polynomial.

4.5 True value: $f(2.5) = \ln(2.5) = 0.916291\dots$

zero order:

$$f(2.5) = f(1) = 0$$

$$\varepsilon_t = \left| \frac{0.916291 - 0}{0.916291} \right| \times 100\% = 100\%$$

first order:

$$f(2.5) = f(1) + f'(1)(2.5-1) = 0 + 1(1.5) = 1.5$$

$$\varepsilon_t = \left| \frac{0.916291 - 1.5}{0.916291} \right| \times 100\% = 63.704\%$$

second order:

$$f(2.5) = 1.5 + \frac{f''(1)}{2}(2.5-1)^2 = 1.5 + \frac{-1}{2}1.5^2 = 0.375$$

$$\varepsilon_t = \left| \frac{0.916291 - 0.375}{0.916291} \right| \times 100\% = 59.074\%$$

third order:

$$f(2.5) = 0.375 + \frac{f^{(3)}(1)}{6}(2.5-1)^3 = 0.375 + \frac{2}{6}1.5^3 = 1.5$$

$$\varepsilon_t = \left| \frac{0.916291 - 1.5}{0.916291} \right| \times 100\% = 63.704\%$$

fourth order:

$$f(2.5) = 1.5 + \frac{f^{(4)}(1)}{24} (2.5 - 1)^4 = 1.5 + \frac{-6}{24} 1.5^4 = 0.234375$$

$$\varepsilon_t = \left| \frac{0.916291 - 0.234375}{0.916291} \right| \times 100\% = 74.421\%$$

Thus, the process seems to be diverging suggesting that a smaller step would be required for convergence.

4.6 True value:

$$f'(x) = 75x^2 - 12x + 7$$

$$f'(2) = 75(2)^2 - 12(2) + 7 = 283$$

function values:

$x_{i-1} = 1.8$	$f(x_{i-1}) = 50.96$
$x_i = 2$	$f(x_i) = 102$
$x_{i+1} = 2.2$	$f(x_{i+1}) = 164.56$

forward:

$$f'(2) = \frac{164.56 - 102}{0.2} = 312.8$$

$$\varepsilon_t = \left| \frac{283 - 312.8}{283} \right| \times 100\% = 10.53\%$$

backward:

$$f'(2) = \frac{102 - 50.96}{0.2} = 255.2$$

$$\varepsilon_t = \left| \frac{283 - 255.2}{283} \right| \times 100\% = 9.823\%$$

centered:

$$f'(2) = \frac{164.56 - 50.96}{2(0.2)} = 284$$

$$\varepsilon_t = \left| \frac{283 - 284}{283} \right| \times 100\% = 0.353\%$$

Both the forward and backward have errors that can be approximated by (recall Eq. 4.15),

$$|E_t| \approx \frac{f''(x_i)}{2} h$$

$$f''(2) = 150x - 12 = 150(2) - 12 = 288$$

$$|E_t| \approx \frac{288}{2} 0.2 = 28.8$$

This is very close to the actual error that occurred in the approximations

$$\text{forward: } |E_t| \approx |283 - 312.8| = 29.8$$

$$\text{backward: } |E_t| \approx |283 - 255.2| = 27.8$$

The centered approximation has an error that can be approximated by,

$$E_t \approx -\frac{f^{(3)}(x_i)}{6} h^2 = -\frac{150}{6} 0.2^2 = -1$$

which is exact: $E_t = 283 - 284 = -1$. This result occurs because the original function is a cubic equation which has zero fourth and higher derivatives.

4.7 True value:

$$f''(x) = 150x - 12$$

$$f''(2) = 150(2) - 12 = 288$$

$h = 0.25$:

$$f''(2) = \frac{f(2.25) - 2f(2) + f(1.75)}{0.25^2} = \frac{182.1406 - 2(102) + 39.85938}{0.25^2} = 288$$

$h = 0.125$:

$$f''(2) = \frac{f(2.125) - 2f(2) + f(1.875)}{0.125^2} = \frac{139.6738 - 2(102) + 68.82617}{0.125^2} = 288$$

Both results are exact because the errors are a function of 4th and higher derivatives which are zero for a 3rd-order polynomial.

4.8

$$\frac{\partial v}{\partial c} = \frac{c g t e^{-(c/m)t} - g m (1 - e^{-(c/m)t})}{c^2} = -1.38666$$

$$\Delta v(\tilde{c}) = \left| \frac{\partial v}{\partial c} \right| \Delta \tilde{c} = 1.38666(1.5) = 2.079989$$

$$v(12.5) = \frac{9.8(50)}{12.5} \left(1 - e^{-12.5(6)/50} \right) = 30.4533$$

$$v = 30.4533 \pm 2.079989$$

Thus, the bounds computed with the first-order analysis range from 28.3733 to 32.5333. This result can be verified by computing the exact values as

$$v(c - \Delta c) = \frac{9.8(50)}{11} \left(1 - e^{-(11/50)^6} \right) = 32.6458$$

$$v(c + \Delta c) = \frac{9.8(50)}{14} \left(1 - e^{-(14/50)^6} \right) = 28.4769$$

Thus, the range of ± 2.0844 is close to the first-order estimate.

4.9

$$v(12.5) = \frac{9.8(50)}{12.5} \left(1 - e^{-12.5(6)/50} \right) = 30.4533$$

$$\Delta v(\tilde{c}, \tilde{m}) = \left| \frac{\partial v}{\partial c} \right| \Delta \tilde{c} + \left| \frac{\partial v}{\partial m} \right| \Delta \tilde{m}$$

$$\frac{\partial v}{\partial c} = \frac{c g t e^{-(c/m)t} - g m (1 - e^{-(c/m)t})}{c^2} = -1.38666$$

$$\frac{\partial v}{\partial m} = \frac{g t}{m} e^{-(c/m)t} + \frac{g}{c} \left(1 - e^{-(c/m)t} \right) = 0.871467$$

$$\Delta v(\tilde{c}, \tilde{m}) = |-1.38666|(1.5) + |0.871467|(2) = 2.079989 + 1.742934 = 3.822923$$

$$v = 30.4533 \pm 3.822923$$

4.10 For $\Delta\tilde{T} = 20$,

$$\Delta H(\tilde{T}) = \left| \frac{\partial H}{\partial T} \right| \Delta \tilde{T}$$

$$\frac{\partial H}{\partial T} = 4Ae\sigma T^3 = 4(0.15)0.9(5.67 \times 10^{-8})650^3 = 8.408$$

$$\Delta H(\tilde{T}) = 8.408(20) = 168.169$$

Exact error:

$$\Delta H_{\text{true}} = \frac{H(670) - H(630)}{2} = \frac{1542.468 - 1205.81}{2} = 168.3286$$

Thus, the first-order approximation is close to the exact result.

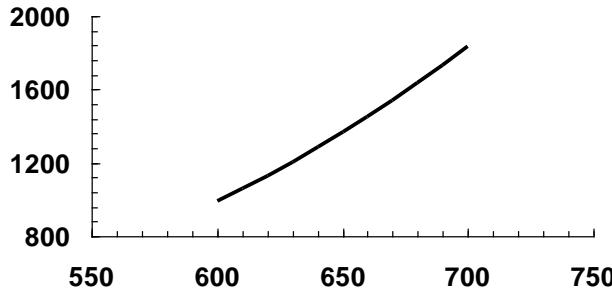
For $\Delta\tilde{T} = 40$,

$$\Delta H(\tilde{T}) = 8.408(40) = 336.3387$$

Exact error:

$$\Delta H_{\text{true}} = \frac{H(690) - H(610)}{2} = \frac{1735.055 - 1059.83}{2} = 337.6124$$

Again, the first-order approximation is close to the exact result. The results are good because $H(T)$ is nearly linear over the ranges we are examining. This is illustrated by the following plot.



4.11 For a sphere, $A = 4\pi r^2$. Therefore,

$$H = 4\pi r^2 e\sigma T^4$$

At the mean values of the parameters,

$$H(0.15, 0.9, 550) = 4\pi(0.15)^2 0.90(5.67 \times 10^{-8})(550)^4 = 1320.288$$

$$\Delta H = \left| \frac{\partial H}{\partial r} \right| \Delta \tilde{r} + \left| \frac{\partial H}{\partial e} \right| \Delta \tilde{e} + \left| \frac{\partial H}{\partial T} \right| \Delta \tilde{T}$$

$$\frac{\partial H}{\partial r} = 8\pi r e \sigma T^4 = 17,603.84$$

$$\frac{\partial H}{\partial e} = 4\pi r^2 \sigma T^4 = 1466.987$$

$$\frac{\partial H}{\partial T} = 16\pi r^2 e \sigma T^3 = 9.6021$$

$$\Delta H = 17603.84(0.01) + 1466.987(0.05) + 9.6021(20) = 441.4297$$

To check this result, we can compute

$$H(0.14, 0.85, 530) = 4\pi(0.14)^2 0.85(5.67 \times 10^{-8})(530)^4 = 936.6372$$

$$H(0.16, 0.95, 570) = 4\pi(0.16)^2 0.95(5.67 \times 10^{-8})(570)^4 = 1829.178$$

$$\Delta H_{\text{true}} = \frac{1829.178 - 936.6372}{2} = 446.2703$$

4.12 The condition number is computed as

$$CN = \frac{\tilde{x}f'(\tilde{x})}{f(\tilde{x})}$$

$$(a) CN = \frac{1.00001 \left[\frac{1}{2\sqrt{1.00001 - 1}} \right]}{\sqrt{1.00001 - 1} + 1} = \frac{1.00001(158.1139)}{1.003162} = 157.617$$

The result is ill-conditioned because the derivative is large near $x = 1$.

$$(b) CN = \frac{10(-e^{-10})}{e^{-10}} = \frac{10(-4.54 \times 10^{-5})}{4.54 \times 10^{-5}} = -10$$

The result is ill-conditioned because x is large.

$$(c) CN = \frac{300 \left[\frac{300}{\sqrt{300^2 + 1}} - 1 \right]}{\sqrt{300^2 + 1} - 300} = \frac{300(-5.555556 \times 10^{-6})}{0.0016667} = -0.99999444$$

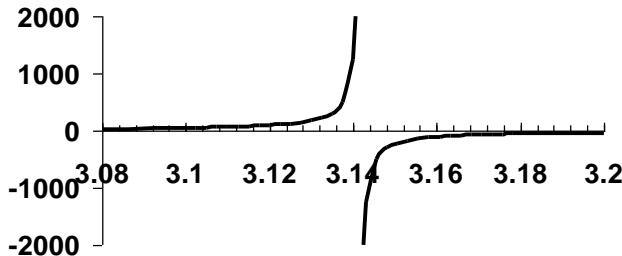
The result is well-conditioned.

$$(d) CN = \frac{x \frac{-xe^{-x} - e^{-x} + 1}{x^2}}{\left(\frac{e^{-x} - 1}{x} \right)} = \frac{0.001(0.499667)}{-0.9995} = -0.0005$$

The result is well-conditioned.

$$(e) CN = \frac{x \frac{(1 + \cos x) \cos x + \sin x (\sin x)}{(1 + \cos x)^2}}{\frac{\sin x}{1 + \cos x}} = \frac{3.141907(20,264,237)}{-6366.2} = -10,001$$

The result is ill-conditioned because, as in the following plot, the function has a singularity at $x = \pi$.



4.13 Addition and subtraction:

$$f(u, v) = u + v$$

$$\Delta f = \left| \frac{\partial f}{\partial u} \right| \Delta \tilde{u} + \left| \frac{\partial f}{\partial v} \right| \Delta \tilde{v}$$

$$\left| \frac{\partial f}{\partial u} \right| = 1 \quad \left| \frac{\partial f}{\partial v} \right| = 1$$

$$f(\tilde{u}, \tilde{v}) = \Delta \tilde{u} + \Delta \tilde{v}$$

Multiplication:

$$f(u, v) = u \cdot v$$

$$\left| \frac{\partial f}{\partial u} \right| = v \quad \left| \frac{\partial f}{\partial v} \right| = u$$

$$f(\tilde{u}, \tilde{v}) = |\tilde{v}| \Delta \tilde{u} + |\tilde{u}| \Delta \tilde{v}$$

Division:

$$f(u, v) = u / v$$

$$\left| \frac{\partial f}{\partial u} \right| = \frac{1}{v} \quad \left| \frac{\partial f}{\partial v} \right| = \frac{u}{v^2}$$

$$f(\tilde{u}, \tilde{v}) = \left| \frac{1}{v} \right| \Delta \tilde{u} + \left| \frac{u}{v^2} \right| \Delta \tilde{v}$$

$$f(\tilde{u}, \tilde{v}) = \frac{|v| \Delta \tilde{u} + |u| \Delta \tilde{v}}{|v^2|}$$

4.14

$$f(x) = ax^2 + bx + c$$

$$f'(x) = 2ax + b$$

$$f''(x) = 2a$$

Substitute these relationships into Eq. (4.4),

$$ax_{i+1}^2 + bx_{i+1} + c = ax_i^2 + bx_i + c + (2ax_i + b)(x_{i+1} - x_i) + \frac{2a}{2!}(x_{i+1}^2 - 2x_{i+1}x_i + x_i^2)$$

Collect terms

$$ax_{i+1}^2 + bx_{i+1} + c = ax_i^2 + 2ax_i(x_{i+1} - x_i) + a(x_{i+1}^2 - 2x_{i+1}x_i + x_i^2) + bx_i + b(x_{i+1} - x_i) + c$$

$$ax_{i+1}^2 + bx_{i+1} + c = ax_i^2 + 2ax_i x_{i+1} - 2ax_i^2 + ax_{i+1}^2 - 2ax_{i+1}x_i + ax_i^2 + bx_i + bx_{i+1} - bx_i + c$$

$$ax_{i+1}^2 + bx_{i+1} + c = (ax_i^2 - 2ax_i^2 + ax_i^2) + ax_{i+1}^2 + (2ax_i x_{i+1} - 2ax_{i+1}x_i) + (bx_i - bx_i) + bx_{i+1} + c$$

$$ax_{i+1}^2 + bx_{i+1} + c = ax_{i+1}^2 + bx_{i+1} + c$$

4.15 The first-order error analysis can be written as

$$\Delta Q = \left| \frac{\partial Q}{\partial n} \right| \Delta n + \left| \frac{\partial Q}{\partial S} \right| \Delta S$$

$$\frac{\partial Q}{\partial n} = -\frac{1}{n^2} \frac{(BH)^{5/3}}{(B+2H)^{2/3}} S^{0.5} = -50.74 \quad \frac{\partial Q}{\partial S} = \frac{1}{n} \frac{(BH)^{5/3}}{(B+2H)^{2/3}} \frac{1}{2S^{0.5}} = 2536.9$$

$$\Delta Q = |-50.74|0.003 + |2536.9|0.00003 = 0.152 + 0.076 = 0.228$$

The error from the roughness is about 2 times the error caused by the uncertainty in the slope. Thus, improving the precision of the roughness measurement would be the best strategy.

4.16 Use the stopping criterion

$$\varepsilon_s = 0.5 \times 10^{-2} \% = 0.5\%$$

True value: $1/(1 - 0.1) = 1.11111\dots$

zero order:

$$\frac{1}{1-x} = 1$$

$$\varepsilon_t = \left| \frac{1.11111 - 1}{1.11111} \right| \times 100\% = 10\%$$

first order:

$$\frac{1}{1-x} = 1 + 0.1 = 1.1$$

$$\varepsilon_t = 1\% \quad \varepsilon_a = \left| \frac{1.1 - 1}{1.1} \right| \times 100\% = 9.0909\%$$

second order:

$$\frac{1}{1-x} = 1 + 0.1 + 0.01 = 1.11$$

$$\varepsilon_t = 0.1\% \quad \varepsilon_a = \left| \frac{1.11 - 1.1}{1.11} \right| \times 100\% = 0.9009009\%$$

third order:

$$\frac{1}{1-x} = 1 + 0.1 + 0.01 + 0.001 = 1.111$$

$$\varepsilon_t = 0.01\% \quad \varepsilon_a = \left| \frac{1.111 - 1.11}{1.111} \right| \times 100\% = 0.090009\%$$

Since the approximate error is below 0.5%, the computation can be terminated.

4.17

$$\Delta(\sin \phi_0) = \left| \frac{d \sin \phi_0}{d\alpha} \right| \Delta\alpha$$

$$\frac{d \sin \phi_0}{d\alpha} = \frac{-\beta}{2\sqrt{(1+\alpha)(1+\alpha-\alpha\beta)}} + \sqrt{1 - \frac{\alpha\beta}{1+\alpha}}$$

where $\beta = (v_e/v_0)^2 = 4$ and $\alpha = 0.25$ to give,

$$\frac{d \sin \phi_0}{d\alpha} = \frac{-4}{2\sqrt{(1+0.25)(1+0.25-0.25(4))}} + \sqrt{1 - \frac{0.25(4)}{1+0.25}} = -3.1305$$

$$\Delta(\sin \phi_0) = 3.1305 \Delta\alpha$$

For $\Delta\alpha = 0.25(0.02) = 0.005$,

$$\Delta(\sin \phi_0) = 3.1305(0.005) = 0.015652$$

$$\sin \phi_0 = (1+0.25) \sqrt{1 - \frac{0.25}{1+0.25}} 4 = 0.559017$$

Therefore,

$$\max \sin \phi_0 = 0.559017 + 0.015652 = 0.574669$$

$$\min \sin \phi_0 = 0.559017 - 0.015652 = 0.543365$$

$$\max \phi_0 = \arcsin(0.574669) \times \frac{180}{\pi} = 35.076^\circ$$

$$\min \phi_0 = \arcsin(0.543365) \times \frac{180}{\pi} = 32.913^\circ$$

4.18

The derivatives can be computed as

$$f(x) = x - 1 - 0.5 \sin x$$

$$f'(x) = 1 - 0.5 \cos x$$

$$f''(x) = 0.5 \sin x$$

$$f^{(3)}(x) = 0.5 \cos x$$

$$f^{(4)}(x) = -0.5 \sin x$$

The first through fourth-order Taylor series expansions can be computed based on Eq. 4.5 as

First-order:

$$f_1(x) = f(a) + f'(a)(x - a)$$

$$f_1(x) = \frac{\pi}{2} - 1 - 0.5 \sin \frac{\pi}{2} + \left[1 - 0.5 \cos \frac{\pi}{2} \right] \left(x - \frac{\pi}{2} \right) = x - 1.5$$

Second-order:

$$f_2(x) = f_1(x) + \frac{f''(a)}{2}(x - a)^2$$

$$f_2(x) = x - 1.5 + 0.25 \sin(\pi/2)(x - \pi/2)^2$$

Third-order:

$$f_3(x) = f_2(x) + \frac{f^{(3)}(a)}{6}(x - a)^3$$

$$f_3(x) = x - 1.5 + 0.25 \sin(\pi/2)(x - \pi/2)^2 + \frac{0.5 \cos(\pi/2)}{6}(x - a)^3$$

$$f_3(x) = x - 1.5 + 0.25 \sin(\pi/2)(x - \pi/2)^2$$

Fourth-order:

$$f_4(x) = f_3(x) + \frac{f^{(4)}(a)}{24}(x - a)^4$$

$$f_4(x) = x - 1.5 + 0.25 \sin(\pi/2)(x - \pi/2)^2 - \frac{0.5 \sin(\pi/2)}{24}(x - \pi/2)^4$$

$$f_4(x) = x - 1.5 + 0.25 \sin(\pi/2)(x - \pi/2)^2 - \frac{1}{48}(x - \pi/2)^4$$

Note the 2nd and 3rd Order Taylor Series functions are the same. The following MATLAB script file which implements and plots each of the functions indicates that the 4th-order expansion satisfies the problem requirements.

```

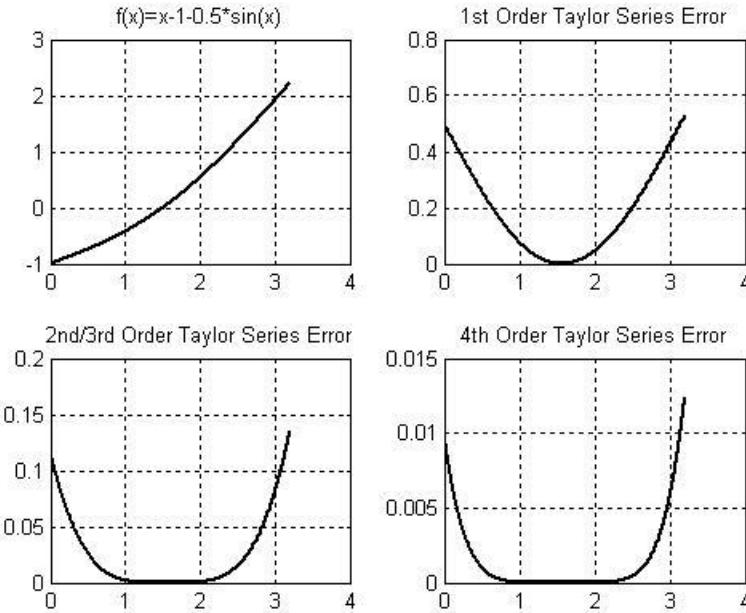
x=0:0.001:3.2;
f=x-1-0.5*sin(x);
subplot(2,2,1);
plot(x,f);grid;title('f(x)=x-1-0.5*sin(x)');hold on

f1=x-1.5;
e1=abs(f-f1);           %Calculates the absolute value of the
difference/error
subplot(2,2,2);
plot(x,e1);grid;title('1st Order Taylor Series Error');

f2=x-1.5+0.25.*((x-0.5*pi).^2);
e2=abs(f-f2);
subplot(2,2,3);
plot(x,e2);grid;title('2nd/3rd Order Taylor Series Error');

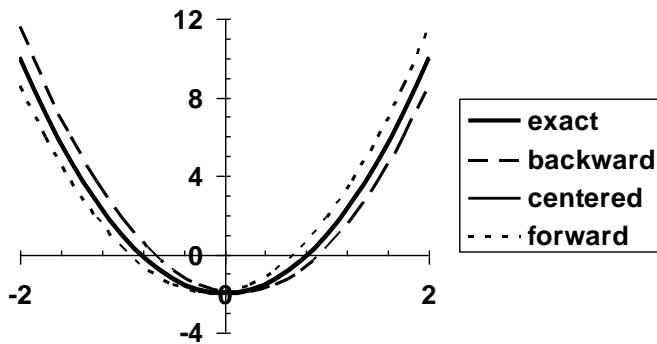
f4=x-1.5+0.25.*((x-0.5*pi).^2)-(1/48)*((x-0.5*pi).^4);
e4=abs(f4-f);
subplot(2,2,4);
plot(x,e4);grid;title('4th Order Taylor Series Error');hold off

```

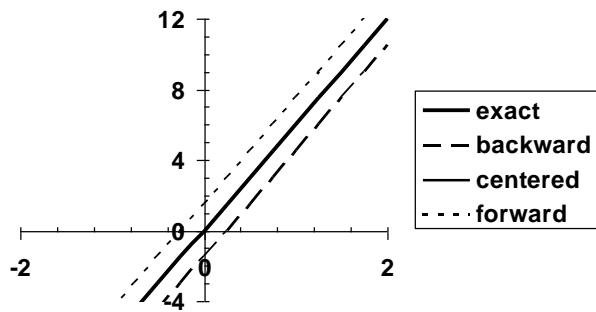


4.19 Here are Excel worksheets and charts that have been set up to solve this problem:

	A	B	C	D	E	F	G	H
1	dx	0.25						
2								
3	First	f(x)	f(x-dx)	f(x+dx)	f(x)-exact	f(x)-back	f(x)-cent	f(x)-forw
4	-2	0	-2.89063	2.140625	10	11.5625	10.0625	8.5625
5	-1.75	2.140625	0	3.625	7.1875	8.5625	7.25	5.9375
6	-1.5	3.625	2.140625	4.546875	4.75	5.9375	4.8125	3.6875
7	-1.25	4.546875	3.625	5	2.6875	3.6875	2.75	1.8125
8	-1	5	4.546875	5.078125	1	1.8125	1.0625	0.3125
9	-0.75	5.078125	5	4.875	-0.3125	0.3125	-0.25	-0.8125
10	-0.5	4.875	5.078125	4.484375	-1.25	-0.8125	-1.1875	-1.5625
11	-0.25	4.484375	4.875	4	-1.8125	-1.5625	-1.75	-1.9375
12	0	4	4.484375	3.515625	-2	-1.9375	-1.9375	-1.9375
13	0.25	3.515625	4	3.125	-1.8125	-1.9375	-1.75	-1.5625
14	0.5	3.125	3.515625	2.921875	-1.25	-1.5625	-1.1875	-0.8125
15	0.75	2.921875	3.125	3	-0.3125	-0.8125	-0.25	0.3125
16	1	3	2.921875	3.453125	1	0.3125	1.0625	1.8125
17	1.25	3.453125	3	4.375	2.6875	1.8125	2.75	3.6875
18	1.5	4.375	3.453125	5.859375	4.75	3.6875	4.8125	5.9375
19	1.75	5.859375	4.375	8	7.1875	5.9375	7.25	8.5625
20	2	8	5.859375	10.89063	10	8.5625	10.0625	11.5625



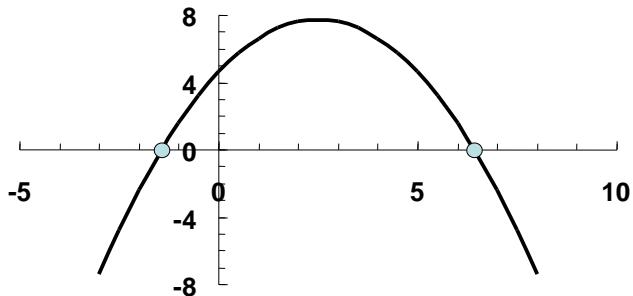
	A	B	C	D	E	F	G	H	I	J
1	dx	0.25								
2										
3	x	f(x)	f(x-dx)	f(x+dx)	f(x-2dx)	f(x+2dx)	f''(x)-exact	f''(x)-back	f''(x)-cent	f''(x)-forw
4	-2	0	-2.89063	2.140625	-6.625	3.625	-12	-13.5	-12	-10.5
5	-1.75	2.140625	0	3.625	-2.89063	4.546875	-10.5	-12	-10.5	-9
6	-1.5	3.625	2.140625	4.546875	0	5	-9	-10.5	-9	-7.5
7	-1.25	4.546875	3.625	5	2.140625	5.078125	-7.5	-9	-7.5	-6
8	-1	5	4.546875	5.078125	3.625	4.875	-6	-7.5	-6	-4.5
9	-0.75	5.078125	5	4.875	4.546875	4.484375	-4.5	-6	-4.5	-3
10	-0.5	4.875	5.078125	4.484375	5	4	-3	-4.5	-3	-1.5
11	-0.25	4.484375	4.875	4	5.078125	3.515625	-1.5	-3	-1.5	0
12	0	4	4.484375	3.515625	4.875	3.125	0	-1.5	0	1.5
13	0.25	3.515625	4	3.125	4.484375	2.921875	1.5	0	1.5	3
14	0.5	3.125	3.515625	2.921875	4	3	3	1.5	3	4.5
15	0.75	2.921875	3.125	3	3.515625	3.453125	4.5	3	4.5	6
16	1	3	2.921875	3.453125	3.125	4.375	6	4.5	6	7.5
17	1.25	3.453125	3	4.375	2.921875	5.859375	7.5	6	7.5	9
18	1.5	4.375	3.453125	5.859375	3	8	9	7.5	9	10.5
19	1.75	5.859375	4.375	8	3.453125	10.89063	10.5	9	10.5	12
20	2	8	5.859375	10.89063	4.375	14.625	12	10.5	12	13.5



PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

CHAPTER 5

5.1 (a) A plot indicates that roots occur at about $x = -1.4$ and 6.4 .



(b)

$$x = \frac{-2.5 \pm \sqrt{(2.5)^2 - 4(-0.5)(4.5)}}{2(-0.5)} = \frac{-1.40512}{6.40512}$$

(c) First iteration:

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

$$\varepsilon_t = \left| \frac{6.40512 - 7.5}{6.40512} \right| \times 100\% = 17.09\% \quad \varepsilon_a = \left| \frac{10 - 5}{10 + 5} \right| \times 100\% = 33.33\%$$

$$f(5)f(7.5) = 4.5(-4.875) = -21.9375$$

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

Second iteration:

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

$$\varepsilon_t = \left| \frac{6.40512 - 6.25}{6.40512} \right| \times 100\% = 2.42\% \quad \varepsilon_a = \left| \frac{7.5 - 5}{7.5 + 5} \right| \times 100\% = 20.00\%$$

$$f(5)f(6.25) = 4.5(0.59375) = 2.672$$

Consequently, the new bracket is $x_l = 6.25$ and $x_u = 7.5$.

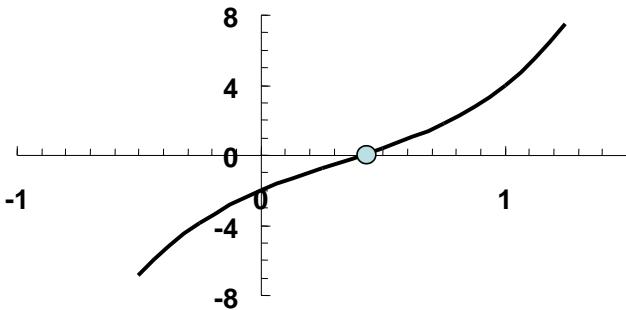
Third iteration:

$$x_r = \frac{6.25 + 7.5}{2} = 6.875$$

$$\varepsilon_t = \left| \frac{6.40512 - 6.875}{6.40512} \right| \times 100\% = 7.34\%$$

$$\varepsilon_a = \left| \frac{7.5 - 6.25}{7.5 + 6.25} \right| \times 100\% = 9.09\%$$

5.2 (a) A plot indicates that a single real root occurs at about $x = 0.42$.



(b) First iteration:

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

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

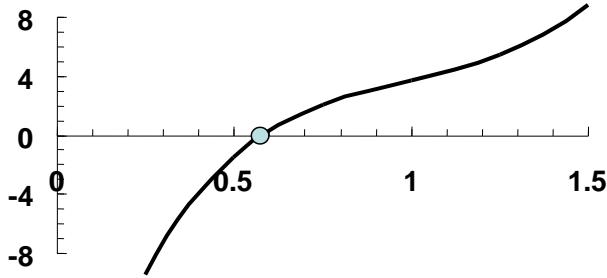
$$f(0)f(0.5) = -2(-0.375) = -0.75$$

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 0.40625.

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	0.375	-0.75	
2	0	0.5	0.25	-2	-0.73438	1.46875	100.00%
3	0.25	0.5	0.375	-0.73438	-0.18945	0.13913	33.33%
4	0.375	0.5	0.4375	-0.18945	0.08667	-0.01642	14.29%
5	0.375	0.4375	0.40625	-0.18945	-0.05246	0.009939	7.69%

5.3 (a) A plot indicates that a single real root occurs at about $x = 0.58$.



(b) Bisection:

First iteration:

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

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

$$f(0.5)f(0.75) = -1.47813(2.07236) = -3.06321$$

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 0.59375.

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	-1.47813	2.07236	-3.06321	
2	0.50000	0.75000	0.62500	-1.47813	0.68199	-1.00807	20.00%
3	0.50000	0.62500	0.56250	-1.47813	-0.28199	0.41682	11.11%
4	0.56250	0.62500	0.59375	-0.28199	0.22645	-0.06386	5.26%

(c) False position:

First iteration:

$$\begin{aligned} x_l &= 0.5 & f(x_l) &= -1.47813 \\ x_u &= 1 & f(x_u) &= 3.7 \end{aligned}$$

$$x_r = 1 - \frac{3.7(0.5 - 1)}{-1.47813 - 3.7} = 0.64273$$

$$f(0.5)f(0.64273) = -1.47813(0.91879) = -1.35808$$

Therefore, the bracket is $x_l = 0.5$ and $x_u = 0.64273$.

Second iteration:

$$\begin{array}{ll} x_l = 0.5 & f(x_l) = -1.47813 \\ x_u = 0.64273 & f(x_u) = 0.91879 \end{array}$$

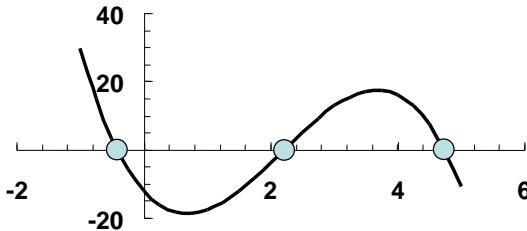
$$x_r = 0.64273 - \frac{0.91879(0.5 - 0.64273)}{-1.47813 - 0.91879} = 0.58802$$

$$\varepsilon_a = \left| \frac{0.58802 - 0.64273}{0.58802} \right| \times 100\% = 9.304\%$$

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 0.57956.

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	-1.47813	3.70000	0.64273	0.91879	-1.35808	
2	0.5	0.64273	-1.47813	0.91879	0.58802	0.13729	-0.20293	9.304%
3	0.5	0.58802	-1.47813	0.13729	0.58054	0.01822	-0.02693	1.289%
4	0.5	0.58054	-1.47813	0.01822	0.57956	0.00238	-0.00351	0.169%

5.4 (a) The graph indicates that roots are located at about -0.4 , 2.25 and 4.7 .



(b) Using bisection, the first iteration is

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

$$f(-1)f(-0.5) = 29.75(3.34375) = 99.47656$$

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

$$x_r = \frac{-0.5 + 0}{2} = -0.25$$

$$\varepsilon_a = \left| \frac{-0.25 - (-0.5)}{-0.25} \right| 100\% = 100\%$$

$$f(-0.5)f(-0.25) = 3.34375(-5.5820313) = -18.66492$$

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>i</i>	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	-1	29.75	0	-12	-0.5	3.34375	
2	-0.5	3.34375	0	-12	-0.25	-5.5820313	100.00%
3	-0.5	3.34375	-0.25	-5.5820313	-0.375	-1.4487305	33.33%
4	-0.5	3.34375	-0.375	-1.4487305	-0.4375	0.8630981	14.29%
5	-0.4375	0.863098	-0.375	-1.4487305	-0.40625	-0.3136673	7.69%
6	-0.4375	0.863098	-0.40625	-0.3136673	-0.421875	0.2694712	3.70%
7	-0.42188	0.269471	-0.40625	-0.3136673	-0.414063	-0.0234052	1.89%
8	-0.42188	0.269471	-0.41406	-0.0234052	-0.417969	0.1227057	0.93%

Thus, after eight iterations, we obtain a root estimate of **-0.417969** with an approximate error of 0.93%, which is below the stopping criterion of 1%.

(c) Using false position, the first iteration is

$$x_r = 0 - \frac{-12(-1-0)}{29.75 - (-12)} = -0.287425$$

$$f(-1)f(-0.287425) = 29.75(-4.4117349) = -131.2491$$

Therefore, the root is in the first interval and the upper guess is redefined as $x_u = -0.287425$. The second iteration is

$$x_r = -0.287425 - \frac{-4.4117349(-1 - (-0.287425))}{29.75 - (-4.4117349)} = -0.3794489$$

$$\varepsilon_a = \left| \frac{-0.3794489 - (-0.287425)}{-0.3794489} \right| 100\% = 24.25\%$$

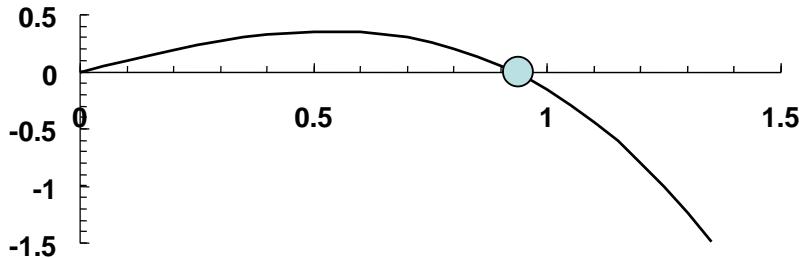
$$f(-1)f(-0.3794489) = 29.75(-1.2896639) = -38.3675$$

Consequently, the root is in the first interval and the upper guess is redefined as $x_u = -0.3794489$. All the iterations are displayed in the following table:

<i>i</i>	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	-1	29.75	0	-12	-0.287425	-4.4117349	
2	-1	29.75	-0.28743	-4.4117349	-0.379449	-1.2896639	24.25%
3	-1	29.75	-0.37945	-1.2896639	-0.405232	-0.3512929	6.36%
4	-1	29.75	-0.40523	-0.3512929	-0.412173	-0.0938358	1.68%
5	-1	29.75	-0.41217	-0.0938358	-0.414022	-0.0249338	0.45%

Therefore, after five iterations we obtain a root estimate of **-0.414022** with an approximate error of 0.45%, 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

$$x_r = \frac{0.75 + 1}{2} = 0.875$$

$$\varepsilon_a = \left| \frac{0.875 - 0.75}{0.875} \right| 100\% = 14.29\%$$

$$f(0.75)f(0.875) = 0.259764(0.0976216) = 0.025359$$

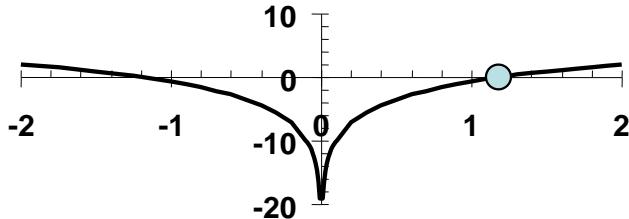
Because the product is positive, the root is in the second interval and the lower guess is redefined as $x_l = 0.875$. All the iterations are displayed in the following table:

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>i</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:

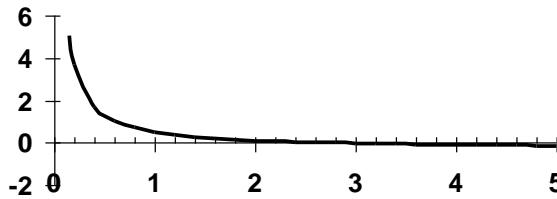
iteration	x_l	x_u	$f(x_l)$	$f(x_u)$	x_r	$f(x_r)$	$f(x_l) \times f(x_r)$	ε_a
1	0.5	2.00000	-3.47259	2.07259	1.43935	0.75678	-2.62797	
2	0.5	1.43935	-3.47259	0.75678	1.27127	0.26007	-0.90312	13.222%
3	0.5	1.27127	-3.47259	0.26007	1.21753	0.08731	-0.30319	4.414%

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>i</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

$$x_r = 5 - \frac{7(4-5)}{-2-7} = 4.22222$$

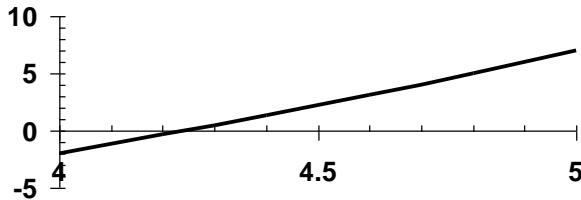
$$f(4)f(4.22222) = -2(-0.17284) = 0.34568$$

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

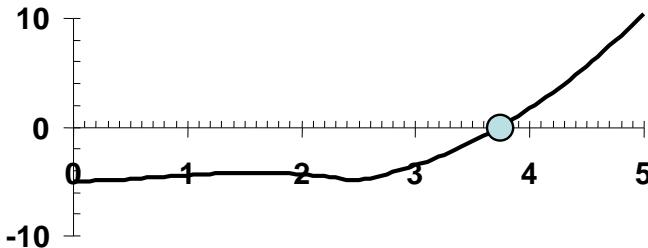
$$x_r = 4.22222 - \frac{7(4.22222 - 5)}{-0.17284 - 7} = 4.24096$$

$$\varepsilon_a = \left| \frac{4.24096 - 4.22222}{4.24096} \right| 100\% = 0.442\%$$

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

$$x_r = 6 - \frac{7(5.01754 - 6)}{-1.00147 - 7} = 5.14051$$

$$\varepsilon_t = \left| \frac{5.60979 - 5.14051}{5.60979} \right| 100\% = 8.37\%$$

$$\varepsilon_a = \left| \frac{5.14051 - 5.01754}{5.14051} \right| 100\% = 2.392\%$$

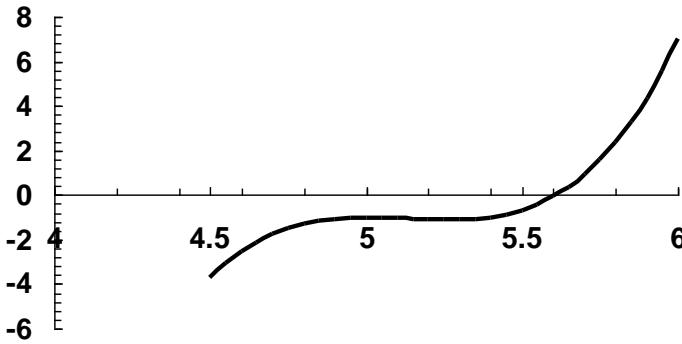
$$f(5.01754)f(5.14051) = -1.00147(-1.06504) = 1.06661$$

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	$f(x_r)$	$f(x_l) \times f(x_r)$	$\varepsilon_a, \%$	$\varepsilon_t, \%$
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).



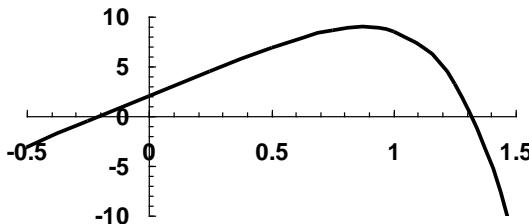
5.11 (a)

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

(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>i</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 0.87.



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

$$f'(x) = -12x^5 - 6x^3 + 10$$

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

<i>i</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	10.00000	8.87500	88.75000	

2	0.50000	1.00000	0.75000	8.87500	4.62109	41.01221	33.33%
3	0.75000	1.00000	0.87500	4.62109	-0.17444	-0.80610	14.29%
4	0.75000	0.87500	0.81250	4.62109	2.53263	11.70351	7.69%
5	0.81250	0.87500	0.84375	2.53263	1.26437	3.20217	3.70%

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

$$f(0.84375) = -2(0.84375)^6 - 1.5(0.84375)^4 + 10(0.84375) + 2 = 8.956$$

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 Solve for the reactions:

$$R_1 = 265 \text{ lbs.} \quad R_2 = 285 \text{ lbs.}$$

Write beam equations:

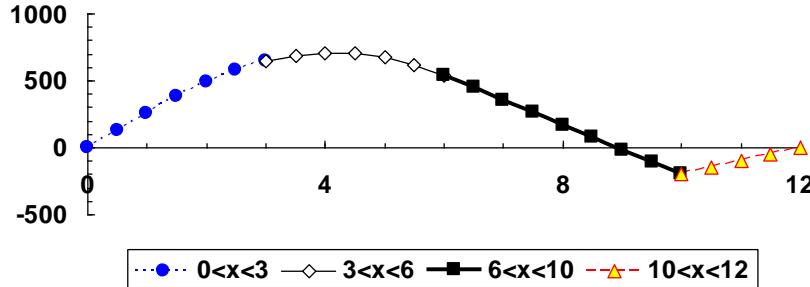
$$\begin{aligned} 0 < x < 3 \\ M + (16.667x^2) \frac{x}{3} - 265x &= 0 \\ (1) \quad M &= 265x - 5.5555556x^3 \end{aligned}$$

$$\begin{aligned} 3 < x < 6 \\ M + 100(x-3)\left(\frac{x-3}{2}\right) + 150\left(x-\frac{2}{3}(3)\right) - 265x &= 0 \\ (2) \quad M &= -50x^2 + 415x - 150 \end{aligned}$$

$$\begin{aligned} 6 < x < 10 \\ M + 150\left(x-\frac{2}{3}(3)\right) + 300(x-4.5) - 265x &= 0 \\ (3) \quad M &= -185x + 1650 \end{aligned}$$

$$\begin{aligned} 10 < x < 12 \\ M + 100(12-x) &= 0 \\ (4) \quad M &= 100x - 1200 \end{aligned}$$

A plot of these equations can be generated:



Combining Equations:

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

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

$$\text{Set } x_L = 6; x_U = 10$$

$$\begin{aligned} M(x_L) &= 540 & x_r &= \frac{x_L + x_U}{2} = 8 \\ M(x_U) &= -200 \end{aligned}$$

$$M(x_R) = 170 \rightarrow \text{replaces } x_L$$

$$\begin{aligned} M(x_L) &= 170 & x_r &= \frac{8+10}{2} = 9 \\ M(x_U) &= -200 \end{aligned}$$

$$M(x_R) = -15 \rightarrow \text{replaces } x_U$$

$$\begin{aligned} M(x_L) &= 170 & x_r &= \frac{8+9}{2} = 8.5 \\ M(x_U) &= -15 \end{aligned}$$

$$M(x_R) = 77.5 \rightarrow \text{replaces } x_L$$

$$\begin{aligned} M(x_L) &= 77.5 & x_r &= \frac{8.5+9}{2} = 8.75 \\ M(x_U) &= -15 \end{aligned}$$

$$M(x_R) = 31.25 \rightarrow \text{replaces } x_L$$

$$\begin{aligned} M(x_L) &= 31.25 & x_r &= \frac{8.75+9}{2} = 8.875 \\ M(x_U) &= -15 \end{aligned}$$

$$M(x_R) = 8.125 \rightarrow \text{replaces } x_L$$

$$\begin{aligned} M(x_L) &= 8.125 & x_r &= \frac{8.875+9}{2} = 8.9375 \\ M(x_U) &= -15 \end{aligned}$$

$$M(x_R) = -3.4375 \rightarrow \text{replaces } x_U$$

$$\begin{aligned} M(x_L) &= 8.125 & x_r &= \frac{8.875 + 8.9375}{2} = 8.90625 \\ M(x_U) &= -3.4375 \\ M(x_R) &= 2.34375 \rightarrow \text{replaces } x_L \end{aligned}$$

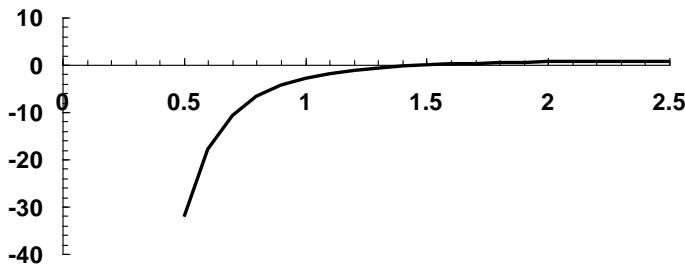
$$\begin{aligned} M(x_L) &= 2.34375 & x_r &= \frac{8.90625 + 8.9375}{2} = 8.921875 \\ M(x_U) &= -3.4375 \\ M(x_R) &= -0.546875 \rightarrow \text{replaces } x_U \end{aligned}$$

$$\begin{aligned} M(x_L) &= 2.34375 & x_r &= \frac{8.90625 + 8.921875}{2} = 8.9140625 \\ M(x_U) &= -0.546875 \\ M(x_R) &= 0.8984 \quad \text{Therefore, } x = 8.91 \text{ feet} \end{aligned}$$

5.15 (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

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

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

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

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.

5.16 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

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

5.17 (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 °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).
x1 = 0 + 273.15;
xu = 40 + 273.15;
if fTa(x1,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:

osf	Approximation	Exact	Error
8	26.75781	26.78017	0.0224
10	15.35156	15.38821	0.0366
12	7.46094	7.46519	0.0043

5.18 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

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

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:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

	A	B	C	D	E
1	Bisection Example				
2					
3					
4	xl	12			
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 $6.6 \times 10^{-3}\%$. These results are all displayed as message boxes. For example, the solution check is displayed as



5.19 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 ($f_{l1} = f(x_l)$), there is a single additional evaluation per iteration ($f_{lr} = f(x_r)$). 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.

5.20 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)
    Loop While ea > es And iter < imax
End Function

```

```

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				
5	xu	16			RUN	
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



5.21 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

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

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 ($5n$).

5.22 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:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

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

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.

CHAPTER 6

6.1 The function can be formulated as a fixed-point iteration as

$$x_{i+1} = 2 \sin(\sqrt{x_i})$$

Using an initial guess of $x_0 = 0.5$, the first iteration is

$$x_1 = 2 \sin(\sqrt{0.5}) = 1.299274$$

$$\varepsilon_a = \left| \frac{1.299274 - 0.5}{1.299274} \right| \times 100\% = 61.517\%$$

The remaining iterations are summarized below. As can be seen, 7 iterations are required to drop below the specified stopping criterion of 0.01%.

i	x	ε_a	ratio
0	0.5		
1	1.299274	61.517%	
2	1.817148	28.499%	0.46327
3	1.950574	6.840%	0.24002
4	1.969743	0.973%	0.14227
5	1.972069	0.118%	0.12122
6	1.972344	0.014%	0.11832
7	1.972377	0.002%	0.11797

The table also includes a column showing the ratio of the relative errors between iterations:

$$\text{ratio} = \frac{\varepsilon_{a,i}}{\varepsilon_{a,i-1}}$$

As can be seen, after the first few iterations this ratio is converging on a constant value of about 0.118. Recall that the error of fixed-point iteration is

$$E_{t,i+1} = g'(\xi)E_{t,i}$$

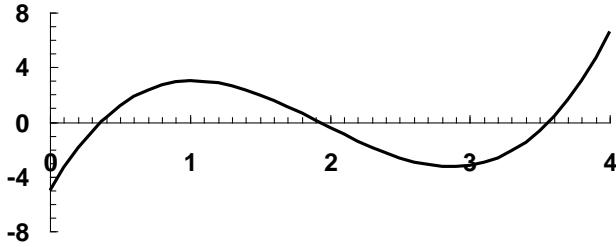
For our problem

$$g'(x) = \frac{d}{dx} 2 \sin(\sqrt{x}) = \frac{1}{\sqrt{x}} \cos(\sqrt{x})$$

The value of this quantity in the vicinity of the true root (1.9724) agrees with the ratio obtained in the table confirming that the convergence is linear and conforms to the theory.

$$g'(1.9724) = \frac{1}{\sqrt{1.9724}} \cos(\sqrt{1.9724}) = 0.1179$$

6.2 (a) Graphical



Root ≈ 3.58

(b) Fixed point

The equation can be solved in numerous ways. A simple way that converges is to solve for the x that is not raised to a power to yield

$$x = \frac{5 - 2x^3 + 11.7x^2}{17.7}$$

The resulting iterations are

<i>i</i>	x_i	ϵ_a
0	3	
1	3.180791	5.68%
2	3.333959	4.59%
3	3.442543	3.15%

(c) Newton-Raphson

<i>i</i>	x_i	$f(x)$	$f'(x)$	ϵ_a
0	3	-3.2	1.5	
1	5.133333	48.09007	55.68667	41.56%
2	4.26975	12.95624	27.17244	20.23%
3	3.792934			12.57%

(d) Secant

<i>i</i>	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ϵ_a
0	3	-3.2	4	6.6	
1	4	6.6	3.326531	-1.9688531	20.25%
2	3.326531	-1.96885	3.481273	-0.7959153	4.44%
3	3.481273	-0.79592	3.586275	0.2478695	2.93%

(e) Modified secant ($\delta = 0.01$)

<i>i</i>	<i>x</i>	<i>f(x)</i>	<i>dx</i>	<i>x+dx</i>	<i>f(x+dx)</i>	<i>f(x)</i>	ϵ_a
0	3	-3.2	0.03	3.03	-3.14928	1.6908	
1	4.892595	35.7632	0.048926	4.9415212	38.09731	47.7068	38.68%
2	4.142949	9.73047	0.041429	4.1843789	10.7367	24.28771	18.09%
3	3.742316	2.203063	0.037423	3.7797391	2.748117	14.56462	10.71%

6.3 (a) Fixed point. The equation can be solved in two ways. The way that converges is

$$x_{i+1} = \sqrt{1.8x_i + 2.5}$$

The resulting iterations are

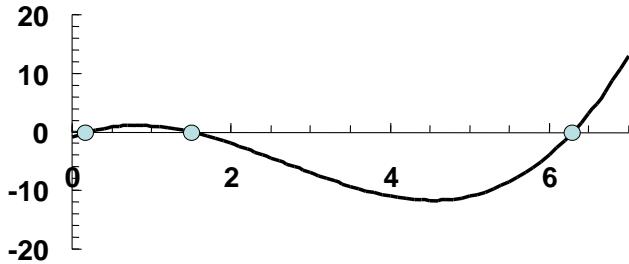
<i>i</i>	<i>x_i</i>	ϵ_a
0	5	
1	3.391165	47.44%
2	2.933274	15.61%
3	2.789246	5.16%
4	2.742379	1.71%
5	2.726955	0.57%
6	2.721859	0.19%
7	2.720174	0.06%
8	2.719616	0.02%

(b) Newton-Raphson

$$x_{i+1} = x_i - \frac{-x_i^2 + 1.8x_i + 2.5}{-2x_i + 1.8}$$

<i>i</i>	<i>x_i</i>	<i>f(x)</i>	<i>f'(x)</i>	ϵ_a
0	5	-13.5	-8.2	
1	3.353659	-2.71044	-4.90732	49.09%
2	2.801332	-0.30506	-3.80266	19.72%
3	2.721108	-0.00644	-3.64222	2.95%
4	2.719341	-3.1E-06	-3.63868	0.06%
5	2.719341	-7.4E-13	-3.63868	0.00%

6.4 (a) A graph of the function indicates that there are 3 real roots at approximately 0.2, 1.5 and 6.3.



(b) The Newton-Raphson method can be set up as

$$x_{i+1} = x_i - \frac{-1 + 5.5x_i - 4x_i^2 + 0.5x_i^3}{5.5 - 8x_i + 1.5x_i^2}$$

This formula can be solved iteratively to determine the three roots as summarized in the following tables:

i	x_i	$f(x)$	$f'(x)$	ϵ_a
0	0	-1	5.5	
1	0.181818	-0.12923	4.095041100.000000%	
2	0.213375	-0.0037	3.861294	14.789338%
3	0.214332	-3.4E-06	3.85425	0.446594%
4	0.214333	-2.8E-12	3.854244	0.000408%

i	x_i	$f(x)$	$f'(x)$	ϵ_a
0	2	-2	-4.5	
1	1.555556	-0.24143	-3.3148128.571429%	
2	1.482723	-0.00903	-3.06408	4.912085%
3	1.479775	-1.5E-05	-3.0536	0.199247%
4	1.479769	-4.6E-11	-3.05358	0.000342%

i	x_i	$f(x)$	$f'(x)$	ϵ_a
0	6	-4	11.5	
1	6.347826	0.625955	15.15974	5.479452%
2	6.306535	0.009379	14.7063	0.654728%
3	6.305898	2.22E-06	14.69934	0.010114%
4	6.305898	1.42E-13	14.69934	0.000002%

Therefore, the roots are 0.214333, 1.479769, and 6.305898.

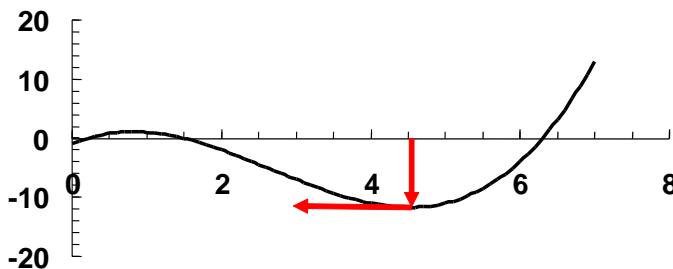
6.5 (a) The Newton-Raphson method can be set up as

$$x_{i+1} = x_i - \frac{-1 + 5.5x_i - 4x_i^2 + 0.5x_i^3}{5.5 - 8x_i + 1.5x_i^2}$$

Using an initial guess of 4.52, this formula jumps around and eventually converges on the root at 0.214333 after 21 iterations:

<i>i</i>	x_i	$f(x)$	$f'(x)$	ϵ_a
0	4.52	-11.6889	-0.0144	
1	-807.209	-2.7E+08	983842.5	100.56%
2	-537.253	-7.9E+07	437265	50.25%
3	-357.284	-2.3E+07	194341.7	50.37%
4	-237.307	-6908482	86375.8	50.56%
5	-157.325	-2046867	38390.94	50.84%
6	-104.009	-606419	17064.32	51.26%
7	-68.4715	-179640	7585.799	51.90%
8	-44.7904	-53200.9	3373.094	52.87%
9	-29.0183	-15746.4	1500.736	54.35%
10	-18.5258	-4654.8	668.5155	56.64%
11	-11.5629	-1372.39	298.5553	60.22%
12	-6.96616	-402.448	134.0203	65.99%
13	-3.96327	-116.755	60.76741	75.77%
14	-2.04193	-33.1655	28.08972	94.09%
15	-0.86123	-9.02308	13.50246	137.09%
16	-0.19298	-2.21394	7.099696	346.28%
17	0.118857	-0.40195	4.570334	262.36%
18	0.206806	-0.02922	3.909707	42.53%
19	0.21428	-0.00021	3.854637	3.49%
20	0.214333	-1E-08	3.854244	0.025%
21	0.214333	-7.3E-17	3.854244	0.000%

The reason for the behavior is depicted in the following plot. As can be seen, the guess of $x = 4.52$ corresponds to a near-zero negative slope of the function. Hence, the first iteration shoots to a large negative value that is far from a root.

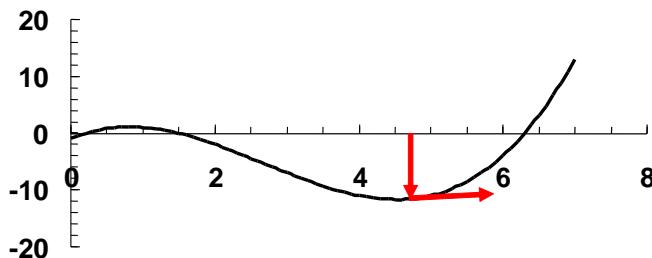


(b) Using an initial guess of 4.54, this formula jumps around and eventually converges on the root at 6.305898 after 14 iterations:

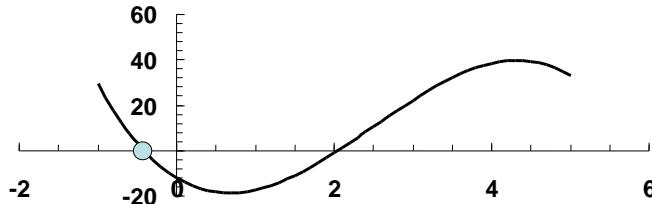
<i>i</i>	x_i	$f(x)$	$f'(x)$	ϵ_a
0	4.54	-11.6881	0.0974	
1	124.5407	904479.6	22274.75	96.35%
2	83.9351	267945.9	9901.672	48.38%
3	56.87443	79358.89	4402.556	47.58%
4	38.84879	23491.6	1958.552	46.40%
5	26.85442	6945.224	872.4043	44.66%
6	18.8934	2047.172	389.7938	42.14%

7	13.64147	598.9375	175.5027	38.50%
8	10.22877	171.854	80.61149	33.36%
9	8.096892	46.70903	39.06436	26.33%
10	6.901198	10.79053	21.73022	17.33%
11	6.40463	1.505003	15.79189	7.75%
12	6.309328	0.050492	14.73681	1.51%
13	6.305902	6.41E-05	14.69938	0.05%
14	6.305898	1.04E-10	14.69934	0.00%

The reason for the behavior is depicted in the following plot. As can be seen, the guess of $x = 4.54$ corresponds to a near-zero positive slope. Hence, the first iteration shoots to a large positive value that is far from a root.



6.6 (a) A graph of the function indicates that the lowest real root is approximately -0.4 :



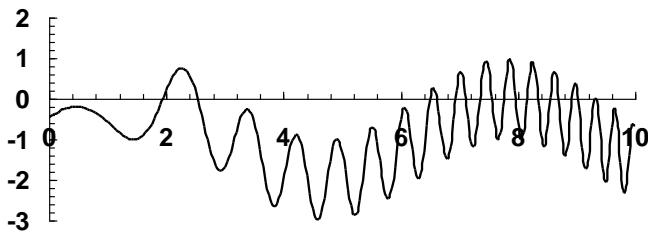
(b) The stopping criterion corresponding to 3 significant figures can be determined with Eq. 3.7 is

$$\varepsilon_s = (0.5 \times 10^{-3})\% = 0.05\%$$

Using initial guesses of $x_{i-1} = -1$ and $x_i = -0.6$, the secant method can be iterated to this level as summarized in the following table:

i	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ε_a
0	-1	29.4	-0.6	7.5984	
1	-0.6	7.5984	-0.46059	1.72547499	30.268%
2	-0.46059	1.725475	-0.41963	0.15922371	9.761%
3	-0.41963	0.159224	-0.41547	0.00396616	1.002%
4	-0.41547	0.003966	-0.41536	9.539E-06	0.026%

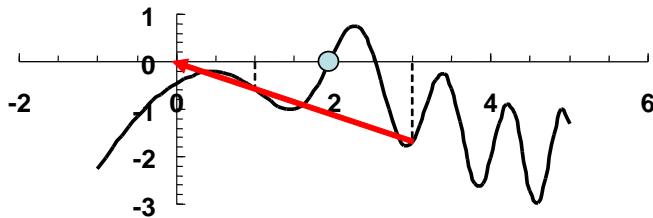
6.7 A graph of the function indicates that the first positive root occurs at about 1.9. However, the plot also indicates that there are many other positive roots.



(a) For initial guesses of $x_{i-1} = 1.0$ and $x_i = 3.0$, four iterations of the secant method yields

i	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ϵ_a
0	1	-0.57468	3	-1.697951521	
1	3	-1.69795	-0.02321	-0.483363437	13023.081%
2	-0.02321	-0.48336	-1.22635	-2.744750012	98.107%
3	-1.22635	-2.74475	0.233951	-0.274717273	624.189%
4	0.233951	-0.27472	0.396366	-0.211940326	40.976%

The result jumps to a negative value due to the poor choice of initial guesses as illustrated in the following plot:

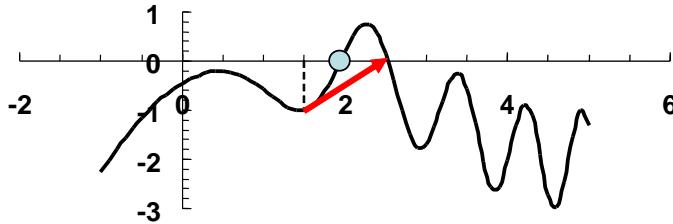


Thereafter, it seems to be converging slowly towards the lowest positive root. However, if the iterations are continued, the technique again runs into trouble when the near-zero slope at 0.5 is approached. At that point, the solution shoots far from the lowest root with the result that it eventually converges to a root at 177.26!

(b) For initial guesses of $x_{i-1} = 1.0$ and $x_i = 3.0$, four iterations of the secant method yields

i	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ϵ_a
0	1.5	-0.99663	2.5	0.1663963	
1	2.5	0.166396	2.356929	0.6698423	6.070%
2	2.356929	0.669842	2.547287	-0.0828279	7.473%
3	2.547287	-0.08283	2.526339	0.0314711	0.829%
4	2.526339	0.031471	2.532107	0.0005701	0.228%

For these guesses, the result jumps to the vicinity of the second lowest root at 2.5 as illustrated in the following plot:

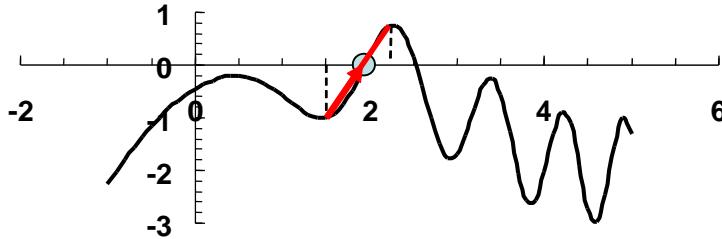


Thereafter, although the two guesses bracket the lowest root, the way that the secant method sequences the iteration results in the technique missing the lowest root.

(c) For initial guesses of $x_{i-1} = 1.5$ and $x_i = 2.25$, four iterations of the secant method yields

i	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ε_a
0	1.5	-0.996635	2.25	0.753821	
1	2.25	0.753821	1.927018	-0.061769	16.761%
2	1.927018	-0.061769	1.951479	0.024147	1.253%
3	1.951479	0.024147	1.944604	-0.000014	0.354%
4	1.944604	-0.000014	1.944608	0.000000	0.000%

For this case, the secant method converges rapidly on the lowest root at 1.9446 as illustrated in the following plot:



6.8 The modified secant method locates the root to the desired accuracy after one iteration:

$$\begin{aligned} x_0 &= 3.5 & f(x_0) &= 0.21178 \\ x_0 + \delta x_0 &= 3.535 & f(x_0 + \delta x_0) &= 3.054461 \end{aligned}$$

$$x_1 = 3.5 - \frac{0.035(0.21178)}{3.054461 - 0.21178} = 3.497392$$

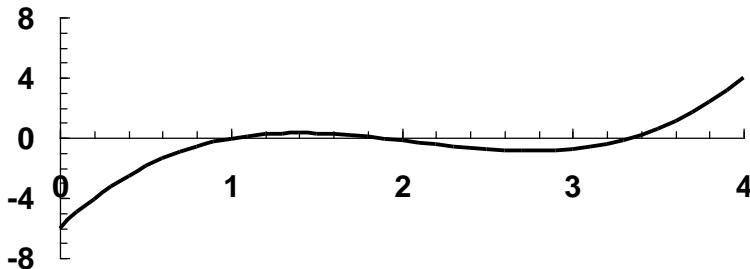
$$\varepsilon_a = \left| \frac{3.497392 - 3.5}{3.497392} \right| \times 100\% = 0.075\%$$

Note that within 3 iterations, the root is determined to 7 significant digits as summarized below:

i	x	$f(x)$	dx	$x+dx$	$f(x+dx)$	$f(x)$	ε_a
0	3.5	0.21178	0.03500	3.535	3.05446	81.2194	
1	3.497392	0.002822	0.03497	3.532366	2.83810	81.0683	0.075%

2	3.497358	3.5E-05	0.03497	3.532331	2.83521	81.0662	0.001%
3	3.497357	4.35E-07	0.03497	3.532331	2.83518	81.0662	0.000%

6.9 (a) Graphical



Highest real root ≈ 3.3

(b) Newton-Raphson

i	x_i	$f(x)$	$f'(x)$	ϵ_a
0	3.5	0.60625	4.5125	
1	3.365651	0.071249	3.468997	3.992%
2	3.345112	0.001549	3.318537	0.614%
3	3.344645	7.92E-07	3.315145	0.014%

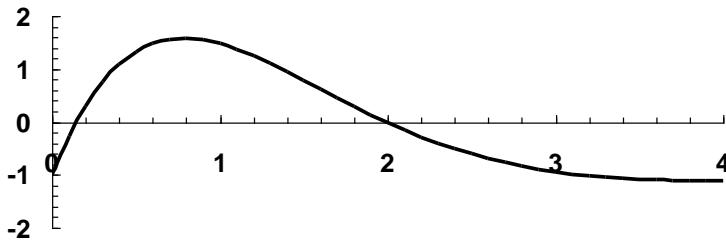
(c) Secant

i	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ϵ_a
0	2.5	-0.78125	3.5	0.60625	
1	3.5	0.60625	3.063063	-0.6667	14.265%
2	3.063063	-0.6667	3.291906	-0.16487	6.952%
3	3.291906	-0.16487	3.367092	0.076256	2.233%

(d) Modified secant ($\delta = 0.01$)

i	x	$x+dx$	$f(x)$	$f(x+dx)$	$f(x)$	ϵ_a
0	3.5	3.535	0.60625	0.76922	4.6563	
1	3.3698	3.4034980.085704	0.207879	3.6256	3.864%	
2	3.346161	3.3796230.005033	0.120439	3.4489	0.706%	
3	3.344702	3.3781490.000187	0.115181	3.4381	0.044%	

6.10 (a) Graphical



Lowest positive real root ≈ 0.15

(b) Newton-Raphson

<i>i</i>	x_i	$f(x)$	$f'(x)$	ϵ_a
0	0.3	0.751414	3.910431	
1	0.107844	-0.22695	6.36737	178.180%
2	0.143487	-0.00895	5.868388	24.841%
3	0.145012	-1.6E-05	5.847478	1.052%

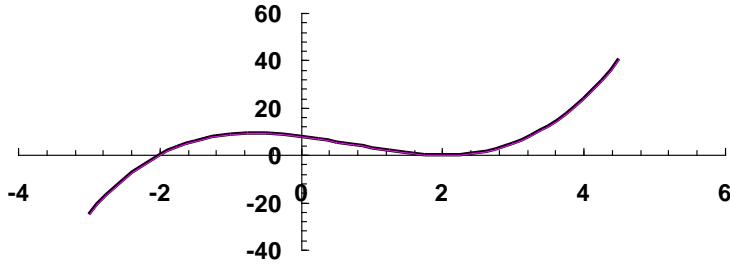
(c) Secant

<i>i</i>	x_{i-1}	$f(x_{i-1})$	x_i	$f(x_i)$	ϵ_a
0	0.5	1.32629	0.4	1.088279	
1	0.4	1.088279	-0.05724	-1.48462	798.821%
2	-0.05724	-1.48462	0.206598	0.334745	127.706%
3	0.206598	0.334745	0.158055	0.075093	30.713%
4	0.158055	0.075093	0.144016	-0.00585	9.748%
5	0.144016	-0.00585	0.14503	9E-05	0.699%

(d) Modified secant ($\delta = 0.01$)

<i>i</i>	x	$x+dx$	$f(x)$	$f(x+dx)$	$f'(x)$	ϵ_a
0	0.3	0.303	0.751414	0.763094	3.893468	
1	0.107007	0.108077	-0.23229	-0.22547	6.371687	180.357%
2	0.143463	0.144897	-0.00909	-0.00069	5.858881	25.412%
3	0.145015	0.146465	-1.2E-06	0.008464	5.83752	1.070%
4	0.145015	0.146465	2.12E-09	0.008465	5.837517	0.000%
5	0.145015	0.146465	-3.6E-12	0.008465	5.837517	0.000%

6.11 As indicated by the following plot, a double root is located at $x = 2$.



(a) The standard Newton-Raphson method can be set up as

$$x_{i+1} = x_i - \frac{x_i^3 - 2x_i^2 - 4x_i + 8}{3x_i^2 - 4x_i - 4}$$

As expected, this method converges slowly as summarized in the following table:

<i>i</i>	<i>x_i</i>	<i>f(x)</i>	<i>f'(x)</i>	<i>ε_a</i>
0	1.2	2.048	-4.48	
1	1.657143	0.429901	-2.3902	27.586%
2	1.837002	0.101942	-1.22428	9.791%
3	1.92027	0.024921	-0.61877	4.336%
4	1.960544	0.006166	-0.31097	2.054%
5	1.980371	0.001534	-0.15588	1.001%
6	1.99021	0.000382	-0.07803	0.494%
7	1.995111	9.55E-05	-0.03904	0.246%
8	1.997557	2.39E-05	-0.01953	0.122%
9	1.998779	5.96E-06	-0.00976	0.061%
10	1.99939	1.49E-06	-0.00488	0.031%
11	1.999695	3.73E-07	-0.00244	0.015%
12	1.999847	9.31E-08	-0.00122	0.008%

(b) The modified Newton-Raphson method (Eq. 6.9a) can be set up for the double root ($m = 2$) as

$$x_{i+1} = x_i - 2 \frac{x_i^3 - 2x_i^2 - 4x_i + 8}{3x_i^2 - 4x_i - 4}$$

This method converges much quicker than the standard approach in **(a)** as summarized in the following table:

<i>i</i>	<i>x_i</i>	<i>f(x)</i>	<i>f'(x)</i>	<i>ε_a</i>
0	1.2	2.048	-4.48	
1	2.114286	0.053738	0.953469	43.243%
2	2.001566	9.81E-06	0.012532	5.632%
3	2	3.75E-13	2.45E-06	0.078%

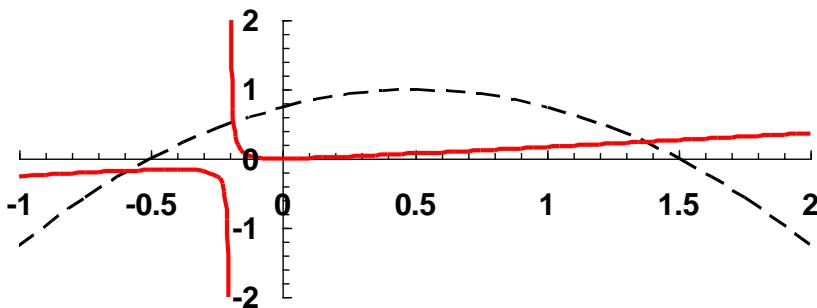
(c) The modified Newton-Raphson method (Eq. 6.13) can be set up for the double root ($m = 2$) as

$$x_{i+1} = x_i - \frac{(x_i^3 - 2x_i^2 - 4x_i + 8)(3x_i^2 - 4x_i - 4)}{(3x_i^2 - 4x_i - 4)^2 - (x_i^3 - 2x_i^2 - 4x_i + 8)(6x_i - 4)}$$

This method also converges much quicker than the standard approach in (a) as summarized in the following table:

i	x_i	$f(x)$	$f'(x)$	$f''(x)$	ϵ_a
0	1.2	2.048	-4.48	3.2	
1	1.878788	0.056989	-0.92562	7.272727	36.129%
2	1.998048	1.52E-05	-0.01561	7.988287	5.969%
3	2	9.09E-13	-3.8E-06	7.999997	0.098%

6.12 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:

$$x = \sqrt{1.2 + 0.75 - 1.2} = 0.866025$$

$$y = \frac{(0.866025)^2}{1 + 5(0.866025)} = 0.14071$$

Second iteration

$$x = \sqrt{0.866025 + 0.75 - 0.14071} = 1.214626$$

$$y = \frac{(1.214626)^2}{1 + 5(1.214626)} = 0.20858$$

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

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

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

The partial derivatives can be computed and evaluated at the initial guesses ($x = 1.2$, $y = 1.2$) as

$$\frac{\partial u}{\partial x} = -2x + 1 = -1.4 \quad \frac{\partial u}{\partial y} = -1$$

$$\frac{\partial v}{\partial x} = 2x - 5y = -3.6 \quad \frac{\partial v}{\partial y} = -1 - 5x = -7$$

The determinant of the Jacobian can be computed as

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

The values of the function at the initial guesses can be computed as

$$u(1.2, 1.2) = -(1.2)^2 + 1.2 + 0.75 - 1.2 = -0.69$$

$$v(1.2, 1.2) = (1.2)^2 - 1.2 - 5(1.2)(1.2) = -6.96$$

These values can be substituted into Eq. (6.21) to give

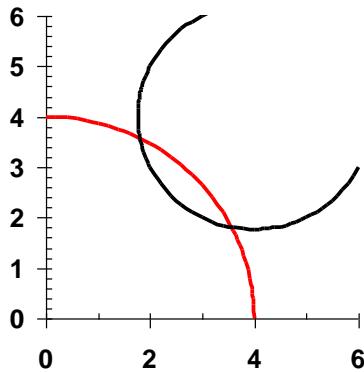
$$x = 1.2 - \frac{-0.69(-7) - (-6.96)(-1)}{6.2} = 1.543548$$

$$y = 1.2 - \frac{-6.96(-1.4) - (-0.69)(-3.6)}{6.2} = 0.0290325$$

The remaining iterations are summarized below:

<i>i</i>	<i>x_i</i>	<i>y_i</i>	<i>ε_a</i>
0	1.2	1.2	
1	1.543548	0.0290325	4033%
2	1.394123	0.2228721	86.97%
3	1.372455	0.2392925	6.86%
4	1.372066	0.2395019	0.0874%
5	1.372065	0.2395019	1.87×10 ⁻⁵ %

6.13 The functions can be plotted (*y* versus *x*). The plot indicates that there are two roots at about (1.8, 3.6) and (3.6, 1.8).



To implement the Newton-Raphson method, the equations to be solved are

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

$$v(x, y) = 16 - x^2 - y^2$$

The partial derivatives can be computed and evaluated at the first set of initial guesses (*x* = 1.8, *y* = 3.6) as

$$\frac{\partial u}{\partial x} = -2(x - 4) = 4.4$$

$$\frac{\partial u}{\partial y} = -2(y - 4) = 0.8$$

$$\frac{\partial v}{\partial x} = -2x = -3.6$$

$$\frac{\partial v}{\partial y} = -2y = -7.2$$

The determinant of the Jacobian can be computed as

$$4.4(-7.2) - 0.8(-3.6) = -28.8$$

The values of the function at the initial guesses can be computed as

$$u(1.8, 3.6) = 5 - (1.8 - 4)^2 - (3.6 - 4)^2 = 0$$

$$v(1.8, 3.6) = 16 - (1.8)^2 - (3.6)^2 = -0.2$$

These values can be substituted into Eq. (6.21) to give

$$x = 1.8 - \frac{0(-7.2) - (-0.2)(0.8)}{-28.8} = 1.805556$$

$$y = 3.6 - \frac{-0.2(4.4) - 0(-3.6)}{-28.8} = 3.569444$$

The remaining iterations are summarized below:

<i>i</i>	<i>x_i</i>	<i>y_i</i>	<i>ε_a</i>
0	1.8	3.6	
1	1.805556	3.569444	0.856%
2	1.805829	3.569171	0.0151%
3	1.805829	3.569171	2.35×10 ⁻⁶ %

For the second set of initial guesses ($x = 3.6$, $y = 1.8$), the partial derivatives can be computed and evaluated as

$$\frac{\partial u}{\partial x} = -2(x - 4) = 0.8 \quad \frac{\partial u}{\partial y} = -2(y - 4) = 4.4$$

$$\frac{\partial v}{\partial x} = -2x = -7.2 \quad \frac{\partial v}{\partial y} = -2y = -3.6$$

The determinant of the Jacobian can be computed as

$$0.8(-3.6) - 4.4(-7.2) = 28.8$$

The values of the function at the initial guesses can be computed as

$$u(1.8, 3.6) = 5 - (3.6 - 4)^2 - (1.8 - 4)^2 = 0$$

$$v(1.8, 3.6) = 16 - (3.6)^2 - (1.8)^2 = -0.2$$

These values can be substituted into Eq. (6.21) to give

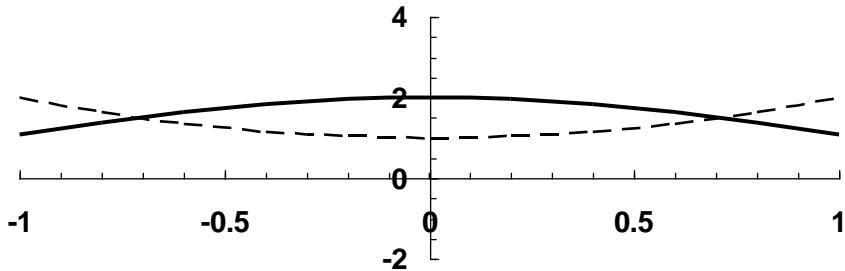
$$x = 3.6 - \frac{0(-3.6) - (-0.2)(4.4)}{28.8} = 3.569444$$

$$y = 1.8 - \frac{-0.2(0.8) - 0(-7.2)}{28.8} = 1.805556$$

The remaining iterations are summarized below:

<i>i</i>	<i>x_i</i>	<i>y_i</i>	<i>ε_a</i>
0	3.6	1.8	
1	3.569444	1.805556	0.856%
2	3.569171	1.805829	0.0151%
3	3.569171	1.805829	2.35×10 ⁻⁶ %

6.14 The functions can be plotted (*y* versus *x*). The plot indicates that there are two roots at about (-0.7, 1.5) and (0.7, 1.5).



To implement the Newton-Raphson method, the equations to be solved are

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

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

We will solve for the positive root. The partial derivatives can be computed and evaluated at the initial guesses (*x* = 0.7, *y* = 1.5) as

$$\frac{\partial u}{\partial x} = 2x = 1.4 \quad \frac{\partial u}{\partial y} = -1$$

$$\frac{\partial v}{\partial x} = -2 \sin(0.7) = -1.288435 \quad \frac{\partial v}{\partial y} = -1$$

The determinant of the Jacobian can be computed as

$$1.4(-1) - (-1)(-1.288435) = -2.688435$$

The values of the function at the initial guesses can be computed as

$$u(0.7, 1.5) = (0.7)^2 + 1 - 1.5 = -0.01$$

$$v(0.7, 1.5) = 2 \cos(0.7) - 1.5 = 0.0296844$$

These values can be substituted into Eq. (6.21) to give

$$x = 0.7 - \frac{-0.01(-1) - 0.0296844(-1)}{-2.688435} = 0.7147611$$

$$y = 1.5 - \frac{0.0296844(1.4) - (-0.01)(-1.288435)}{-2.688435} = 1.510666$$

The remaining iterations are summarized below:

<i>i</i>	<i>x_i</i>	<i>y_i</i>	<i>ε_a</i>
0	0.7	1.5	
1	0.7147611	1.510666	2.065%
2	0.7146211	1.510683	0.0196%
3	0.7146211	1.510683	1.76×10 ⁻⁶ %

6.15 The function to be evaluated is

$$f(c) = \frac{W}{V} - \frac{Q}{V} c - k\sqrt{c} = 1 - 0.1c - 0.25\sqrt{c}$$

Using an initial guess of $x_0 = 4$ and $\delta = 0.5$, the three iterations can be summarized as

<i>i</i>	<i>x</i>	<i>x+dx</i>	<i>f(x)</i>	<i>f(x+dx)</i>	<i>f'(x)</i>	<i>ε_a</i>
0	4	6	0.1	-0.21237	-0.15619	
1	4.640261	6.960392	-0.00256	-0.3556	-0.15217	13.798%
2	4.623452	6.935178	9.94E-05	-0.35189	-0.15226	0.364%
3	4.624105	6.936158	-3.8E-06	-0.35203	-0.15226	0.014%

Therefore, the root is estimated as $c = 4.624105$. This result can be checked by substituting it into the function to yield,

$$f(c) = 1 - 0.1(4.624105) - 0.25\sqrt{4.624105} = -3.8 \times 10^{-6}$$

6.16 Convergence can be evaluated in two ways. First, we can calculate the derivative of the right-hand side and determine whether it is greater than one. Second, we can develop a graphical representation as in Fig. 6.3.

For the first formulation, the derivative can be evaluated as

$$g'(c) = -\frac{2Q(W - Qc)}{(kV)^2} = -3.2 + 0.32c$$

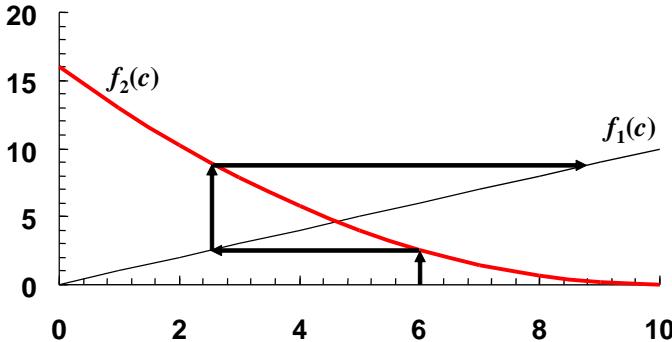
Between, $c = 2$ and 6 , this ranges from -2.56 and -1.28 . Therefore, because $|g'(c)| > 1$, we would expect that fixed-point iteration would be divergent.

The second way to assess divergence is to create a plot of

$$f_1(c) = c$$

$$f_2(c) = \left(\frac{W - Qc}{kV} \right)^2$$

The result also indicates divergence:



For the second formulation, the derivative can be evaluated as

$$g'(c) = -\frac{kV}{2Q\sqrt{c}} = -\frac{1.25}{\sqrt{c}}$$

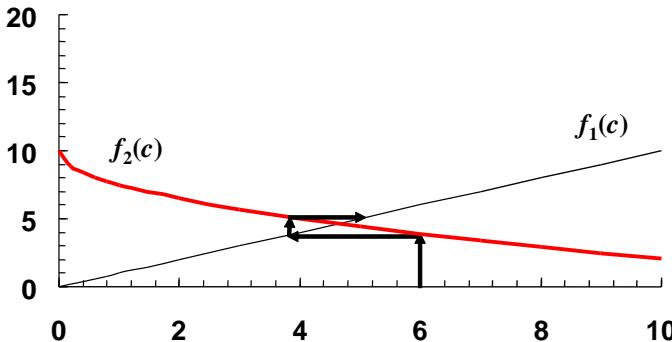
Between, $c = 2$ and 6 , this ranges from -0.883 and -0.51 . Therefore, because $|g'(c)| < 1$, we would expect that fixed-point iteration would be convergent.

The second way to assess divergence is to create a plot of

$$f_1(c) = c$$

$$f_2(c) = \frac{W - kV\sqrt{c}}{Q}$$

The result also indicates convergence:



Here are the results of using fixed-point iteration to determine the root for the second formulation.

<i>i</i>	<i>x_i</i>	<i>ε_a</i>	<i>E_{t,i}/E_{t,i-1}</i>
0	4		
1	5	20.0%	-0.60236
2	4.40983	13.4%	-0.56994
3	4.750101	7.2%	-0.58819
4	4.551318	4.4%	-0.57739
5	4.666545	2.5%	-0.5836
6	4.599453	1.5%	-0.57997
7	4.638416	0.8%	-0.58207
8	4.615754	0.5%	-0.58084
9	4.628923	0.3%	-0.58156
10	4.621267	0.2%	-0.58114

Notice that we have included the true error and the ratio of the true errors between iterations. The latter should be equal to $|g'(c)|$, which at the root is equal to

$$g'(4.624081) = -\frac{1.25}{\sqrt{4.624081}} = -0.5813$$

Thus, the computation verifies the theoretical result that was derived in Box 6.1 (p. 138).

6.17 Here is a VBA program to implement the Newton-Raphson algorithm and solve Example 6.3.

```

Option Explicit

Sub NewtRaph()
    Dim imax As Integer, iter As Integer
    Dim x0 As Double, es As Double, ea As Double
    x0 = 0#
    es = 0.01
    imax = 20
    MsgBox "Root: " & NewtR(x0, es, imax, iter, ea)
    MsgBox "Iterations: " & iter
    MsgBox "Estimated error: " & ea
End Sub

Function df(x)
    df = -Exp(-x) - 1#
End Function

Function f(x)
    f = Exp(-x) - x
End Function

Function NewtR(x0, es, imax, iter, ea)
    Dim xr As Double, xrold As Double
    xr = x0
    iter = 0
    Do
        xrold = xr

```

```

xr = xr - f(xr) / df(xr)
iter = iter + 1
If (xr <> 0) Then
    ea = Abs((xr - xrold) / xr) * 100
End If
If ea < es Or iter >= imax Then Exit Do
Loop
NewtR = xr
End Function

```

When this program is run, it yields a root of 0.5671433 after 4 iterations. The approximate error at this point is $2.21 \times 10^{-5}\%$.

6.18 Here is a VBA program to implement the secant algorithm and solve Example 6.6.

```

Option Explicit

Sub SecMain()
    Dim imax As Integer, iter As Integer
    Dim x0 As Double, x1 As Double, xr As Double
    Dim es As Double, ea As Double
    x0 = 0
    x1 = 1
    es = 0.01
    imax = 20
    MsgBox "Root: " & Secant(x0, x1, xr, es, imax, iter, ea)
    MsgBox "Iterations: " & iter
    MsgBox "Estimated error: " & ea
End Sub

Function f(x)
    f = Exp(-x) - x
End Function

Function Secant(x0, x1, xr, es, imax, iter, ea)
    xr = x1
    iter = 0
    Do
        xr = x1 - f(x1) * (x0 - x1) / (f(x0) - f(x1))
        iter = iter + 1
        If (xr <> 0) Then
            ea = Abs((xr - x1) / xr) * 100
        End If
        If ea < es Or iter >= imax Then Exit Do
        x0 = x1
        x1 = xr
    Loop
    Secant = xr
End Function

```

When this program is run, it yields a root of 0.5671433 after 4 iterations. The approximate error at this point is $4.77 \times 10^{-3}\%$.

For MATLAB users, here is an M-file to solve the same problem:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

function root = secant(func,xrold,xr,es,maxit)
% secant(func,xrold,xr,es,maxit):
%   uses secant method to find the root of a function
% input:
%   func = name of function
%   xrold, xr = initial guesses
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

% if necessary, assign default values
if nargin<5, maxit=50; end      %if maxit blank set to 50
if nargin<4, es=0.001; end       %if es blank set to 0.001
% Secant method
iter = 0;
while (1)
    xrn = xr - func(xr)*(xrold - xr)/(func(xrold) - func(xr));
    iter = iter + 1;
    if xrn ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
    xrold = xr;
    xr = xrn;
end
root = xrn;

>> secant(inline('exp(-x)-x'),0,1)

ans =
0.5671

```

6.19 Here is a VBA program to implement the modified secant algorithm and solve Example 6.8.

```

Option Explicit

Sub SecMod()
Dim imax As Integer, iter As Integer
Dim x As Double, es As Double, ea As Double
x = 1
es = 0.01
imax = 20
MsgBox "Root: " & ModSecant(x, es, imax, iter, ea)
MsgBox "Iterations: " & iter
MsgBox "Estimated error: " & ea
End Sub

Function f(x)
f = Exp(-x) - x
End Function

Function ModSecant(x, es, imax, iter, ea)
Dim xr As Double, xrold As Double, fr As Double
Const del As Double = 0.01
xr = x

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

iter = 0
Do
    xrold = xr
    fr = f(xr)
    xr = xr - fr * del * xr / (f(xr + del * xr) - fr)
    iter = iter + 1
    If (xr <> 0) Then
        ea = Abs((xr - xrold) / xr) * 100
    End If
    If ea < es Or iter >= imax Then Exit Do
Loop
ModSecant = xr
End Function

```

When this program is run, it yields a root of 0.5671433 after 4 iterations. The approximate error at this point is $2.36 \times 10^{-5}\%$.

For MATLAB users, here is an M-file to solve the same problem:

```

function root = modsec(func,xr,delta,es,maxit)
% modsec(func,xr,delta,es,maxit):
%   uses the modified secant method
%   to find the root of a function
% input:
%   func = name of function
%   xr = initial guess
%   delta = perturbation fraction
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

% if necessary, assign default values
if nargin<5, maxit=50; end      %if maxit blank set to 50
if nargin<4, es=0.001; end       %if es blank set to 0.001
if nargin<3, delta=1E-5; end     %if delta blank set to 0.00001
% Secant method
iter = 0;
while (1)
    xrold = xr;
    xr = xr - delta*xr*func(xr) / (func(xr+delta*xr)-func(xr));
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
end
root = xr;

>> modsec(inline('exp(-x)-x'),1,.01)

ans =
    0.5671

```

6.20 Here is a VBA program to implement the 2 equation Newton-Raphson method and solve Example 6.10.

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

Option Explicit

Sub NewtRaphSyst()
    Dim imax As Integer, iter As Integer
    Dim x0 As Double, y0 As Double, xr As Double
    Dim yr As Double, es As Double, ea As Double
    x0 = 1.5
    y0 = 3.5
    es = 0.01
    imax = 20
    Call NR2Eqs(x0, y0, xr, yr, es, imax, iter, ea)
    MsgBox "x, y = " & xr & ", " & yr
    MsgBox "Iterations: " & iter
    MsgBox "Estimated error: " & ea
End Sub

Sub NR2Eqs(x0, y0, xr, yr, es, imax, iter, ea)
    Dim J As Double, eay As Double
    iter = 0
    Do
        J = dudx(x0, y0) * dvdy(x0, y0) - dudy(x0, y0) * dvdx(x0, y0)
        xr = x0 - (u(x0, y0) * dvdy(x0, y0) - v(x0, y0) * dudy(x0, y0)) / J
        yr = y0 - (v(x0, y0) * dudx(x0, y0) - u(x0, y0) * dvdx(x0, y0)) / J
        iter = iter + 1
        If (xr <> 0) Then
            ea = Abs((xr - x0) / xr) * 100
        End If
        If (xr <> 0) Then
            eay = Abs((yr - y0) / yr) * 100
        End If
        If eay > ea Then ea = eay
        If ea < es Or iter >= imax Then Exit Do
        x0 = xr
        y0 = yr
    Loop
End Sub

Function u(x, y)
    u = x ^ 2 + x * y - 10
End Function

Function v(x, y)
    v = y + 3 * x * y ^ 2 - 57
End Function

Function dudx(x, y)
    dudx = 2 * x + y
End Function

Function dudy(x, y)
    dudy = x
End Function

Function dvdx(x, y)
    dvdx = 3 * y ^ 2
End Function

Function dvdy(x, y)
    dvdy = 1 + 6 * x * y
End Function

```

Its application yields roots of $x = 2$ and $y = 3$ after 4 iterations. The approximate error at this point is $1.96 \times 10^{-5}\%$.

6.21 The program from Prob. 6.20 can be set up to solve Prob. 6.11, by changing the functions to

```

Function u(x, y)
u = y + x ^ 2 - 0.75 - x
End Function

Function v(x, y)
v = x ^ 2 - 5 * x * y - y
End Function

Function dudx(x, y)
dudx = 2 * x - 1
End Function

Function dudy(x, y)
dudy = 1
End Function

Function dvdx(x, y)
dvdx = 2 * x ^ 2 - 5 * y
End Function

Function dvdy(x, y)
dvdy = -5 * x
End Function

```

Using a stopping criterion of 0.01%, the program yields $x = 1.3720655$ and $y = 0.2395017$ after 6 iterations with an approximate error of $1.89 \times 10^{-3}\%$.

The program from Prob. 6.20 can be set up to solve Prob. 6.12, by changing the functions to

```

Function u(x, y)
u = (x - 4) ^ 2 + (y - 4) ^ 2 - 5
End Function

Function v(x, y)
v = x ^ 2 + y ^ 2 - 16
End Function

Function dudx(x, y)
dudx = 2 * (x - 4)
End Function

Function dudy(x, y)
dudy = 2 * (y - 4)
End Function

Function dvdx(x, y)
dvdx = 2 * x
End Function

```

```
Function dvdy(x, y)
dvdy = 2 * y
End Function
```

Using a stopping criterion of 0.01% and initial guesses of 1.8 and 3.6, the program yields $x = 1.805829$ and $y = 3.569171$ after 3 iterations with an approximate error of 2.35×10^{-6} .

Using a stopping criterion of 0.01% and initial guesses of 3.6 and 1.8, the program yields $x = 3.569171$ and $y = 1.805829$ after 3 iterations with an approximate error of 2.35×10^{-6} .

6.22 Determining the square root of a number can be formulated as a roots problem:

$$x = \sqrt{a}$$

$$x^2 = a$$

$$f(x) = x^2 - a = 0 \quad (1)$$

The derivative of this function is

$$f'(x) = 2x \quad (2)$$

Substituting (1) and (2) into the Newton Raphson formula (Eq. 6.6) gives

$$x = x - \frac{x^2 - a}{2x}$$

Combining terms yields the “divide and average” method,

$$x = \frac{2x(x) - x^2 + a}{2} = \frac{x^2 + a/x}{2}$$

6.23 (a) The formula for Newton-Raphson is

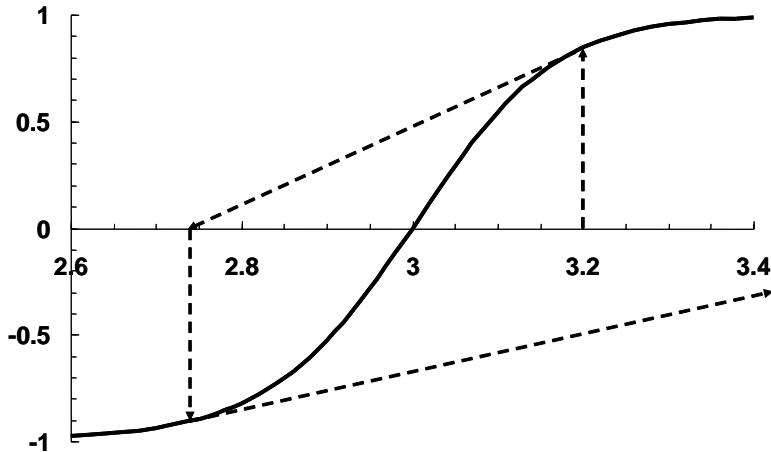
$$x_{i+1} = x_i - \frac{\tanh(x_i^2 - 9)}{2x_i \operatorname{sech}^2(x_i^2 - 9)}$$

Using an initial guess of 3.2, the iterations proceed as

iteration	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	3.2	0.845456	1.825311	
1	2.736816	-0.906910	0.971640	16.924%
2	3.670197	0.999738	0.003844	25.431%
3	-256.413			101.431%

Note that on the fourth iteration, the computation should go unstable.

(b) The solution diverges from its real root of $x = 3$. Due to the concavity of the slope, the next iteration will always diverge. The following graph illustrates how the divergence evolves.



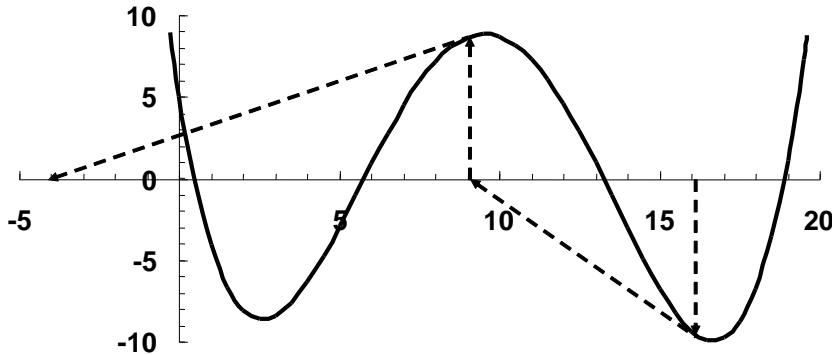
6.24 The formula for Newton-Raphson is

$$x_{i+1} = x_i - \frac{0.0074x_i^4 - 0.284x_i^3 + 3.355x_i^2 - 12.183x_i + 5}{0.0296x_i^3 - 0.852x_i^2 + 6.71x_i - 12.183}$$

Using an initial guess of 16.15, the iterations proceed as

iteration	x_i	$f(x_i)$	$f'(x_i)$	ϵ_a
0	16.15	-9.57445	-1.35368	
1	9.077102	8.678763	0.662596	77.920%
2	-4.02101	128.6318	-54.864	325.742%
3	-1.67645	36.24995	-25.966	139.852%
4	-0.2804	8.686147	-14.1321	497.887%
5	0.334244	1.292213	-10.0343	183.890%
6	0.463023	0.050416	-9.25584	27.813%
7	0.46847	8.81E-05	-9.22351	1.163%
8	0.46848	2.7E-10	-9.22345	0.002%

As depicted below, the iterations involve regions of the curve that have flat slopes. Hence, the solution is cast far from the roots in the vicinity of the original guess.



6.25

$$f(x) = \pm \sqrt{16 - (x + 1)^2} + 2$$

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

1st iteration

$$x_{i-1} = 0.5 \Rightarrow f(x_{i-1}) = -1.708$$

$$x_i = 3 \Rightarrow f(x_i) = 2$$

$$x_{i+1} = 3 - \frac{2(0.5 - 3)}{(-1.708 - 2)} = 1.6516$$

2nd iteration

$$x_i = 1.6516 \Rightarrow f(x_i) = -0.9948$$

$$x_{i-1} = 0.5 \Rightarrow f(x_{i-1}) = -1.46$$

$$x_{i+1} = 1.6516 - \frac{-0.9948(0.5 - 1.6516)}{(-1.46 - -0.9948)} = 4.1142$$

The solution diverges because the secant created by the two x -values yields a solution outside the function's domain.

6.26

The equation to be solved is

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

Because this equation is easy to differentiate, the Newton-Raphson is the best choice to achieve results efficiently. It can be formulated as

$$x_{i+1} = x_i - \frac{\pi R x_i^2 - \left(\frac{\pi}{3}\right) x_i^3 - V}{2\pi R x_i - \pi x_i^2}$$

or substituting the parameter values,

$$x_{i+1} = x_i - \frac{\pi(3)x_i^2 - \left(\frac{\pi}{3}\right)x_i^3 - 30}{2\pi(3)x_i - \pi x_i^2}$$

The iterations can be summarized as

iteration	x_i	$f(x_i)$	$f'(x_i)$	$ \varepsilon_a $
0	3	26.54867	28.27433	
1	2.061033	0.866921	25.50452	45.558%
2	2.027042	0.003449	25.30035	1.677%
3	2.026906	5.68E-08	25.29952	0.007%

Thus, after only three iterations, the root is determined to be 2.026906 with an approximate relative error of 0.007%.

CHAPTER 7

7.1 In a fashion similar to Example 7.1, $n = 4$, $a_0 = -20$, $a_1 = 3$, $a_2 = 14.5$, $a_3 = -7.5$, $a_4 = 1$ and $t = 2$. These can be used to compute

$$\begin{aligned} r &= a_4 = 1 \\ a_4 &= 0 \end{aligned}$$

For $i = 3$,

$$\begin{aligned} s &= a_3 = -7.5 \\ a_3 &= r = 1 \\ r &= s + rt = -7.5 + 1(2) = -5.5 \end{aligned}$$

For $i = 2$,

$$\begin{aligned} s &= a_2 = 14.5 \\ a_2 &= r = -5.5 \\ r &= s + rt = 14.5 - 5.5(2) = 3.5 \end{aligned}$$

For $i = 1$,

$$\begin{aligned} s &= a_1 = 3 \\ a_1 &= r = 3.5 \\ r &= s + rt = 3 + 3.5(2) = 10 \end{aligned}$$

For $i = 0$,

$$\begin{aligned} s &= a_0 = -20 \\ a_0 &= r = 10 \\ r &= s + rt = -20 + 10(2) = 0 \end{aligned}$$

Therefore, the quotient is $x^3 - 5.5x^2 + 3.5x + 10$ with a remainder of zero. Thus, 2 is a root. This result can be easily verified with MATLAB,

```
>> a = [1 -7.5 14.5 3 -20];
>> b = [1 -2];
>> [d,e] = deconv(a,b)

d =
    1.0000   -5.5000    3.5000   10.0000
e =
    0         0         0         0         0
```

7.2 In a fashion similar to Example 7.1, $n = 5$, $a_0 = 10$, $a_1 = -7$, $a_2 = -6$, $a_3 = 1$, $a_4 = -5$, $a_5 = 1$, and $t = 2$. These can be used to compute

$$\begin{aligned} r &= a_4 = 1 \\ a_4 &= 0 \end{aligned}$$

For $i = 4$,

$$\begin{aligned} s &= a_4 = -5 \\ a_3 &= r = 1 \end{aligned}$$

$$r = s + rt = -5 + 1(2) = -3$$

For $i = 3$,

$$s = a_3 = 1$$

$$a_3 = r = -3$$

$$r = s + rt = 1 - 3(2) = -5$$

For $i = 2$,

$$s = a_2 = -6$$

$$a_2 = r = -5$$

$$r = s + rt = -6 - 5(2) = -16$$

For $i = 1$,

$$s = a_1 = -7$$

$$a_1 = r = -16$$

$$r = s + rt = -7 - 16(2) = -39$$

For $i = 0$,

$$s = a_0 = 10$$

$$a_0 = r = -39$$

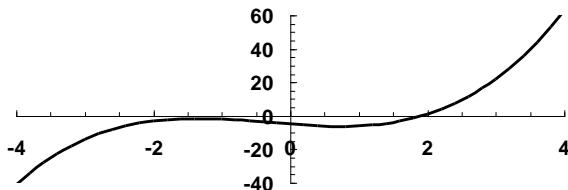
$$r = s + rt = 10 - 39(2) = -68$$

Therefore, the quotient is $x^4 - 3x^3 - 5x^2 - 16x - 39$ with a remainder of -68 . Thus, 2 is not a root. This result can be easily verified with MATLAB,

```
>> a=[1 -5 1 -6 -7 10];
>> b = [1 -2];
>> [d,e] = deconv(a,b)

d =
    1     -3     -5    -16    -39
e =
    0      0      0      0      0     -68
```

7.3 (a) A plot indicates a root at about $x = 2$.



Try initial guesses of $x_0 = 1$, $x_1 = 1.5$, and $x_2 = 2.5$. Using the same approach as in Example 7.2,

First iteration:

$$f(1) = -6$$

$$h_0 = 0.5$$

$$f(1.5) = -3.875$$

$$h_1 = 1$$

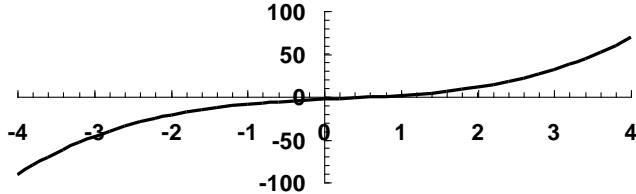
$$f(2.5) = 9.375$$

$$\begin{aligned}\delta_0 &= 4.25 & \delta_1 &= 13.25 \\ a &= \frac{13.25 - 4.25}{1 + 0.5} = 6 & b &= 6(1) + 13.25 = 19.25 & c &= 9.375 \\ x_3 &= 2.5 + \frac{-2(9.375)}{19.25 + \sqrt{19.25^2 - 4(6)(9.375)}} = 1.901244 \\ \varepsilon_a &= \left| \frac{1.901244 - 2.5}{1.901244} \right| \times 100\% = 31.49\%\end{aligned}$$

The iterations can be continued as tabulated below:

<i>i</i>	x_3	ε_a
0	1.901244	31.4929%
1	1.919270	0.9392%
2	1.919639	0.0192%
3	1.919640	0.0000%

(b) A plot indicates a root at about $x = 0.7$.



Try initial guesses of $x_0 = 0.5$, $x_1 = 1$, and $x_2 = 1.5$. Using the same approach as in Example 7.2,

First iteration:

$$\begin{aligned}f(0.5) &= -1 & f(1) &= 1.5 & f(1.5) &= 5.25 \\ h_0 &= 0.5 & h_1 &= 0.5 & & \\ \delta_0 &= 5 & \delta_1 &= 7.5 & & \\ a &= \frac{7.5 - 5}{0.5 + 0.5} = 2.5 & b &= 2.5(0.5) + 7.5 = 8.75 & c &= 5.25 \\ x_3 &= 1.5 + \frac{-2(5.25)}{8.75 + \sqrt{8.75^2 - 4(2.5)(5.25)}} = 0.731071 \\ \varepsilon_a &= \left| \frac{0.731071 - 1.5}{0.731071} \right| \times 100\% = 105.18\%\end{aligned}$$

The iterations can be continued as tabulated below:

<i>i</i>	x_3	ε_a
0	0.731071	105.1785%
1	0.720767	1.4296%

2	0.721231	0.0643%
3	0.721230	0.0001%

7.4 Here are MATLAB sessions to determine the roots:

(a)

```
>> a=[1 -1 3 -2];
>> roots(a)

ans =
0.1424 + 1.6661i
0.1424 - 1.6661i
0.7152
```

(b)

```
>> a=[2 0 6 0 10];
>> roots(a)

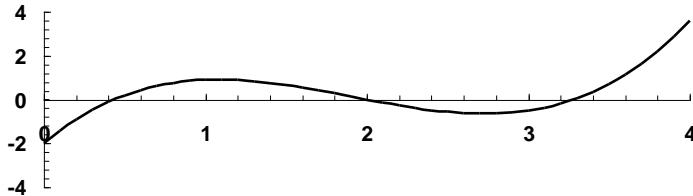
ans =
-0.6067 + 1.3668i
-0.6067 - 1.3668i
0.6067 + 1.3668i
0.6067 - 1.3668i
```

(c)

```
>> a=[1 -2 6 -8 8];
>> roots(a)

ans =
-0.0000 + 2.0000i
-0.0000 - 2.0000i
1.0000 + 1.0000i
1.0000 - 1.0000i
```

7.5 (a) A plot suggests 3 real roots: 0.44, 2 and 3.3.



Try $r = 1$ and $s = -1$, and follow Example 7.3

1st iteration:

$$\begin{array}{ll} \Delta r = 1.085 & \Delta s = 0.887 \\ r = 2.085 & s = -0.1129 \end{array}$$

2nd iteration:

$$\Delta r = 0.4019 \quad \Delta s = -0.5565$$

$$r = 2.487 \quad s = -0.6694$$

3rd iteration:

$$\begin{aligned}\Delta r &= -0.0605 & \Delta s &= -0.2064 \\ r &= 2.426 & s &= -0.8758\end{aligned}$$

4th iteration:

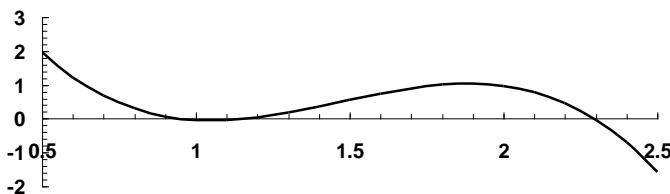
$$\begin{aligned}\Delta r &= 0.00927 & \Delta s &= 0.00432 \\ r &= 2.436 & s &= -0.8714\end{aligned}$$

$$\text{root}_1 = \frac{2.436 + \sqrt{2.436^2 + 4(-0.8714)}}{2} = 2$$

$$\text{root}_2 = \frac{2.436 - \sqrt{2.436^2 + 4(-0.8714)}}{2} = 0.4357$$

The remaining root₃ = 3.279.

(b) Plot suggests 3 real roots at approximately 0.9, 1.2 and 2.3.



Try $r = 2$ and $s = -0.5$, and follow Example 7.3

1st iteration:

$$\begin{aligned}\Delta r &= 0.2302 & \Delta s &= -0.5379 \\ r &= 2.2302 & s &= -1.0379\end{aligned}$$

2nd iteration:

$$\begin{aligned}\Delta r &= -0.1799 & \Delta s &= -0.0422 \\ r &= 2.0503 & s &= -1.0801\end{aligned}$$

3rd iteration:

$$\begin{aligned}\Delta r &= 0.0532 & \Delta s &= -0.01641 \\ r &= 2.1035 & s &= -1.0966\end{aligned}$$

4th iteration:

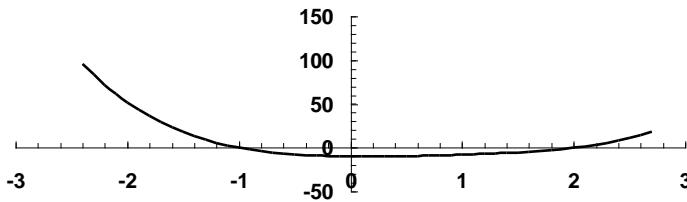
$$\begin{aligned}\Delta r &= 0.00253 & \Delta s &= -0.00234 \\ r &= 2.106 & s &= -1.099\end{aligned}$$

$$\text{root}_1 = \frac{2.106 + \sqrt{2.106^2 + 4(-1.099)}}{2} = 1.1525$$

$$\text{root}_2 = \frac{2.106 - \sqrt{2.106^2 + 4(-1.099)}}{2} = 0.9535$$

The remaining root₃ = 2.2947

(c) Plot suggests 2 real roots at approximately -1 and 2.2. This means that there should also be 2 complex roots



Try r = -1 and s = 1, and follow Example 7.3

1st iteration:

$$\begin{aligned}\Delta r &= 2.171 & \Delta s &= 3.947 \\ r &= 1.171 & s &= 4.947\end{aligned}$$

2nd iteration:

$$\begin{aligned}\Delta r &= -0.0483 & \Delta s &= -2.260 \\ r &= 1.123 & s &= 2.688\end{aligned}$$

3rd iteration:

$$\begin{aligned}\Delta r &= -0.0931 & \Delta s &= -0.6248 \\ r &= 1.030 & s &= 2.063\end{aligned}$$

4th iteration:

$$\begin{aligned}\Delta r &= -0.0288 & \Delta s &= -0.0616 \\ r &= 1 & s &= 2\end{aligned}$$

$$\text{root}_1 = \frac{1 + \sqrt{1^2 + 4(2)}}{2} = 2$$

$$\text{root}_2 = \frac{1 - \sqrt{1^2 + 4(2)}}{2} = -1$$

The remaining roots are 1 + 2i and 1 - 2i.

7.6 Here is a VBA program to implement the Müller algorithm and solve Example 7.2.

```

Option Explicit

Sub TestMull()
    Dim maxit As Integer, iter As Integer
    Dim h As Double, xr As Double, eps As Double
    h = 0.1
    xr = 5
    eps = 0.001
    maxit = 20
    Call Muller(xr, h, eps, maxit, iter)
    MsgBox "root = " & xr
    MsgBox "Iterations: " & iter
End Sub

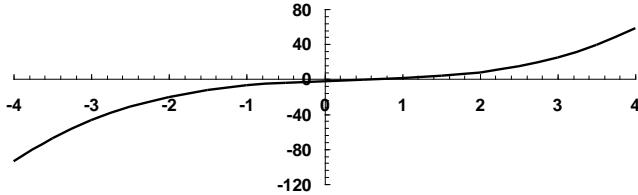
Sub Muller(xr, h, eps, maxit, iter)
    Dim x0 As Double, x1 As Double, x2 As Double
    Dim h0 As Double, h1 As Double, d0 As Double, d1 As Double
    Dim a As Double, b As Double, c As Double
    Dim den As Double, rad As Double, dxr As Double
    x2 = xr
    x1 = xr + h * xr
    x0 = xr - h * xr
    Do
        iter = iter + 1
        h0 = x1 - x0
        h1 = x2 - x1
        d0 = (f(x1) - f(x0)) / h0
        d1 = (f(x2) - f(x1)) / h1
        a = (d1 - d0) / (h1 + h0)
        b = a * h1 + d1
        c = f(x2)
        rad = Sqr(b * b - 4 * a * c)
        If Abs(b + rad) > Abs(b - rad) Then
            den = b + rad
        Else
            den = b - rad
        End If
        dxr = -2 * c / den
        xr = x2 + dxr
        If Abs(dxr) < eps * xr Or iter >= maxit Then Exit Do
        x0 = x1
        x1 = x2
        x2 = xr
    Loop
End Sub

Function f(x)
    f = x ^ 3 - 13 * x - 12
End Function

```

When this program is run, it yields the correct result of 4 in 3 iterations.

7.7 The plot suggests a real root at 0.7.



Using initial guesses of $x_0 = 0.63$, $x_1 = 0.77$ and $x_2 = 0.7$, the software developed in Prob. 7.6 yields a root of 0.715225 in 2 iterations.

7.8 Here is a VBA program to implement the Bairstow algorithm to solve Example 7.3.

```

Option Explicit

Sub PolyRoot()
    Dim n As Integer, maxit As Integer, ier As Integer, i As Integer
    Dim a(10) As Double, re(10) As Double, im(10) As Double
    Dim r As Double, s As Double, es As Double
    n = 5
    a(0) = 1.25: a(1) = -3.875: a(2) = 2.125: a(3) = 2.75: a(4) = -3.5: a(5) = 1
    maxit = 20
    es = 0.0001
    r = -1
    s = -1
    Call Bairstow(a(), n, es, r, s, maxit, re(), im(), ier)
    For i = 1 To n
        If im(i) >= 0 Then
            MsgBox re(i) & " + " & im(i) & "i"
        Else
            MsgBox re(i) & " - " & Abs(im(i)) & "i"
        End If
    Next i
End Sub

Sub Bairstow(a, nn, es, rr, ss, maxit, re, im, ier)
    Dim iter As Integer, n As Integer, i As Integer
    Dim r As Double, s As Double, ea1 As Double, ea2 As Double
    Dim det As Double, dr As Double, ds As Double
    Dim r1 As Double, i1 As Double, r2 As Double, i2 As Double
    Dim b(10) As Double, c(10) As Double
    r = rr
    s = ss
    n = nn
    ier = 0
    ea1 = 1
    ea2 = 1
    Do
        If n < 3 Or iter >= maxit Then Exit Do
        iter = 0
        Do
            iter = iter + 1
            b(n) = a(n)
            b(n - 1) = a(n - 1) + r * b(n)
            c(n) = b(n)
            c(n - 1) = b(n - 1) + r * c(n)
            For i = n - 2 To 0 Step -1
                b(i) = a(i) + r * b(i + 1) + s * b(i + 2)
                c(i) = b(i) + r * c(i + 1) + s * c(i + 2)
            Next i
        Loop While Abs(ea1 - ea2) > es
        ea1 = ea2
        ea2 = Abs(re(n))
    Loop
    re = r
    im = s
End Sub

```

```

Next i
det = c(2) * c(2) - c(3) * c(1)
If det <> 0 Then
    dr = (-b(1) * c(2) + b(0) * c(3)) / det
    ds = (-b(0) * c(2) + b(1) * c(1)) / det
    r = r + dr
    s = s + ds
    If r <> 0 Then ea1 = Abs(dr / r) * 100
    If s <> 0 Then ea2 = Abs(ds / s) * 100
Else
    r = r + 1
    s = s + 1
    iter = 0
End If
If ea1 <= es And ea2 <= es Or iter >= maxit Then Exit Do
Loop
Call Quadroot(r, s, r1, i1, r2, i2)
re(n) = r1
im(n) = i1
re(n - 1) = r2
im(n - 1) = i2
n = n - 2
For i = 0 To n
    a(i) = b(i + 2)
Next i
Loop
If iter < maxit Then
    If n = 2 Then
        r = -a(1) / a(2)
        s = -a(0) / a(2)
        Call Quadroot(r, s, r1, i1, r2, i2)
        re(n) = r1
        im(n) = i1
        re(n - 1) = r2
        im(n - 1) = i2
    Else
        re(n) = -a(0) / a(1)
        im(n) = 0
    End If
Else
    ier = 1
End If
End Sub

Sub Quadroot(r, s, r1, i1, r2, i2)
Dim disc
disc = r ^ 2 + 4 * s
If disc > 0 Then
    r1 = (r + Sqr(disc)) / 2
    r2 = (r - Sqr(disc)) / 2
    i1 = 0
    i2 = 0
Else
    r1 = r / 2
    r2 = r1
    i1 = Sqr(Abs(disc)) / 2
    i2 = -i1
End If
End Sub

```

When this program is run, it yields the correct result of -1 , 0.5 , 2 , $1 + 0.5i$, and $1 - 0.5i$.

7.9 Using the software developed in Prob. 7.8 the following results should be generated for the three parts of Prob. 7.5:

- (a) 3.2786, 2.0000, 0.4357
- (b) 2.2947, 1.1525, 0.9535
- (c) 2.0000, $1.0000 + 2.0000i$, $1.0000 - 2.0000i$, -1

7.10 The goal seek set up is

	B1	=B1^3.5
1	x	3.759703
2	$x^{3.5}$	102.9516
3		
4		
5		
6		
7		
8		
9		

Goal Seek
 Set cell: \$B\$2
 To value: 80
 By changing cell: \$B\$1

The result is

	B2	=B1^3.5
1	x	3.497367
2	$x^{3.5}$	80.00077
3		
4		
5		
6		
7		
8		
9		
10		

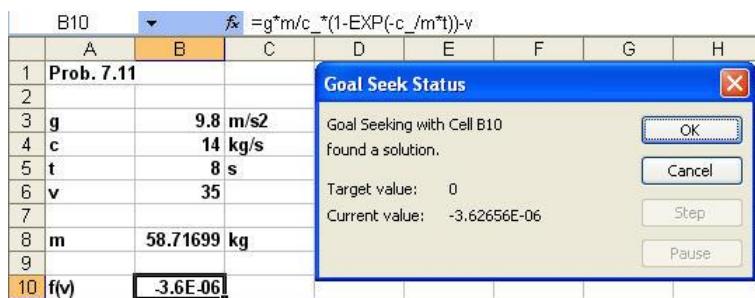
Goal Seek Status
 Goal Seeking with Cell B2 found a solution.
 Target value: 80
 Current value: 80.00077448

7.11 The goal seek set up is shown below. Notice that we have named the cells containing the parameter values with the labels in column A.

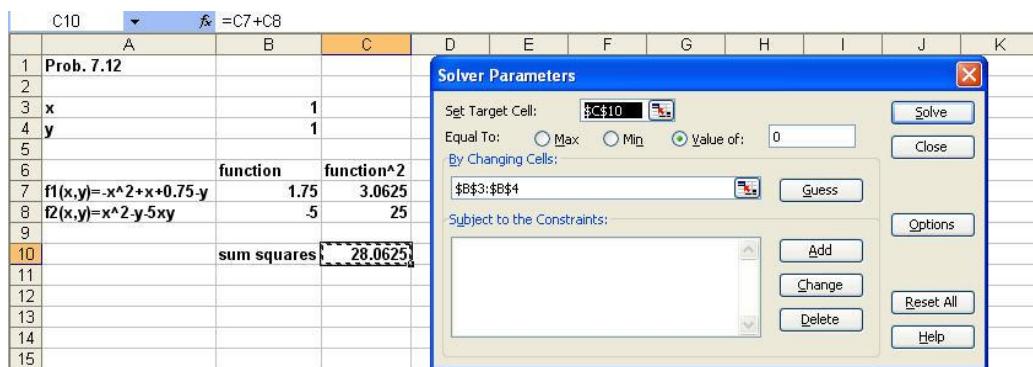
	m	=g*m/c_* (1-EXP(-c_/m*t))-v
1	Prob. 7.11	
2		
3	g	9.8 m/s ²
4	c	14 kg/s
5	t	8 s
6	v	35
7	m	50 kg
8		
9		
10	f(v)	3.72605

Goal Seek
 Set cell: B10
 To value: 0
 By changing cell: \$B\$8

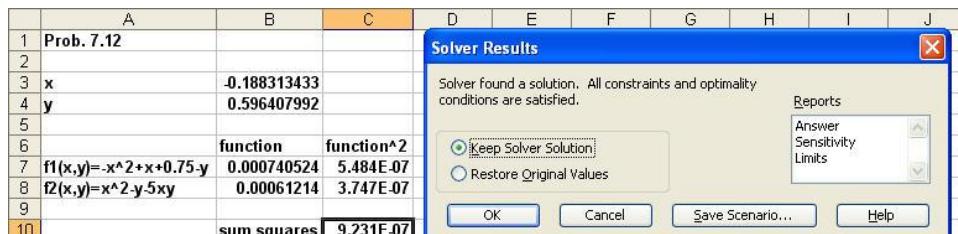
The result is 58.717 kg as shown here:



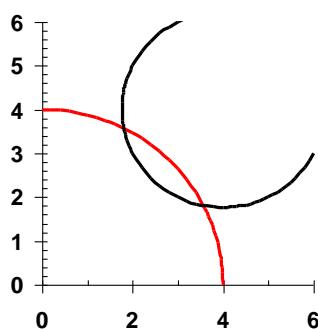
7.12 The Solver set up is shown below using initial guesses of $x = y = 1$. Notice that we have rearranged the two functions so that the correct values will drive both to zero. We then drive the sum of their squared values to zero by varying x and y . This is done because a straight sum would be zero if $f_1(x,y) = -f_2(x,y)$.



The result is



7.13 A plot of the functions indicates two real roots at about (1.8, 3.6) and (3.6, 1.8).



The Solver set up is shown below using initial guesses of (2, 4). Notice that we have rearranged the two functions so that the correct values will drive both to zero. We then drive the sum of their squared values to zero by varying x and y. This is done because a straight sum would be zero if $f_1(x,y) = -f_2(x,y)$.

The screenshot shows a Microsoft Excel spreadsheet with the Solver Parameters dialog box open. The Solver Parameters dialog box has the following settings:

- Set Target Cell:** \$C\$10 (containing '=C7+C8')
- Equal To:** Value of: 0
- By Changing Cells:** \$B\$3:\$B\$4
- Subject to the Constraints:** None (empty)

The worksheet (Prob. 7.13) contains the following data:

	A	B	C
1	Prob. 7.13		
2			
3	x	2	
4	y	4	
6		function	function^2
7	f1(x,y)=5-(x-4)^2-(y-4)^2	1	1
8	f2(x,y)=16-x^2-y^2	-4	16
10		sum squares	17

The result is

The screenshot shows the Solver Results dialog box with the following message:

Solver found a solution. All constraints and optimality conditions are satisfied.

Reports

Keep Solver Solution
 Restore Original Values

Buttons: OK, Cancel, Save Scenario..., Help

The worksheet (Prob. 7.13) now shows the solved values:

	A	B	C
1	Prob. 7.13		
2			
3	x	1.805916437	
4	y	3.569117141	
6		function	function^2
7	f1(x,y)=5-(x-4)^2-(y-4)^2	0.00037279	1.138E-07
8	f2(x,y)=16-x^2-y^2	6.86609E-05	4.714E-09
10		sum squares	1.185E-07

For guesses of (4, 2) the result is (3.5691, 1.8059).

7.14 MATLAB session:

```
>> a = poly([4 -2 1 -5 7])
a =
    1     -5    -35    125    194   -280

>> polyval(a,1)
ans =
    0

>> polyder(a)
ans =
    5    -20   -105    250    194

>> b = poly([4 -2])
b =
    1     -2     -8

>> [d,e] = deconv(a,b)
d =
    1     -3    -33     35
```

```

e =
    0      0      0      0      0      0

>> roots(d)
ans =
    7.0000
   -5.0000
    1.0000

>> conv(d,b)
ans =
    1     -5    -35    125    194   -280

>> r = roots(a)
r =
    7.0000
   -5.0000
    4.0000
   -2.0000
   -2.0000
    1.0000

```

7.15 MATLAB sessions:

Prob. 7.5a:

```

>> a=[.7 -4 6.2 -2];
>> roots(a)
ans =
    3.2786
    2.0000
    0.4357

```

Prob. 7.5b:

```

>> a=[-3.704 16.3 -21.97 9.34];
>> roots(a)
ans =
    2.2947
    1.1525
    0.9535

```

Prob. 7.5c:

```

>> a=[1 -3 5 -1 -10];
>> roots(a)
ans =
    2.0000
    1.0000 + 2.0000i
    1.0000 - 2.0000i
   -1.0000

```

7.16 Here is a program written in Fortran 90:

```

PROGRAM Root
Use IMSL !This establishes the link to the IMSL libraries
Implicit None !forces declaration of all variables
Integer::nroot

```

```

Parameter(nroot=1)
Integer::itmax=50
Real::errabs=0.,errrel=1.E-5,eps=0.,eta=0.
Real::f,x0(nroot) ,x(nroot)
External f
Integer::info(nroot)
Print *, "Enter initial guess"
Read *, x0
Call ZReal(f,errabs,errrel,eps,eta,nroot,itmax,x0,x,info)
Print *, "root = ", x
Print *, "iterations = ", info
End

Function f(x)
Implicit None
Real::f,x
f = x**3-x**2+3.*x-2.
End

```

The output for Prob. 7.4a would look like

```

Enter initial guess
.5
root = 0.7152252
iterations = 6
Press any key to continue

```

The other parts of Probs 7.4 have complex roots and therefore cannot be evaluated with ZReal. The roots for Prob. 7.5 can be evaluated by changing the function and obtaining the results:

- 7.5 (a)** 2, 0.4357, 3.279
- 7.5 (b)** 1.1525, 0.9535, 2.295
- 7.5 (c)** 2, -1

$$\mathbf{7.17} \quad x_2 = 0.62, x_1 = 0.64, x_0 = 0.60$$

$$h_0 = 0.64 - 0.60 = 0.04$$

$$h_1 = 0.62 - 0.64 = -0.02$$

$$\delta_0 = \frac{60 - 20}{0.64 - 0.60} = 1000$$

$$\delta_1 = \frac{50 - 60}{0.62 - 0.64} = 500$$

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} = \frac{500 - 1000}{-0.02 + 0.04} = 25000$$

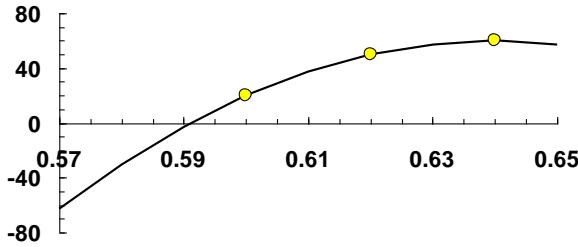
$$b = ah_1 + \delta_1 = -25000(-0.02) + 500 = 1000$$

$$c = 50$$

$$\sqrt{b^2 - 4ac} = \sqrt{1000^2 - 4(-25000)50} = 2449.49$$

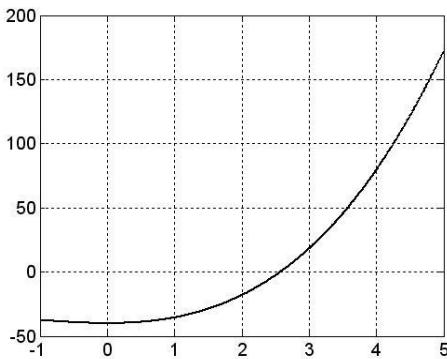
$$t_0 = 0.62 + \frac{-2(50)}{1000 + 2449.49} = 0.591$$

Therefore, the pressure was zero at 0.591 seconds. The result is graphically displayed below:

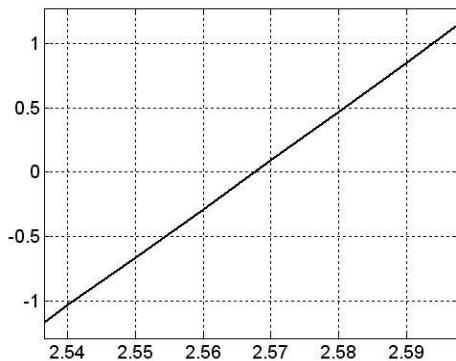


7.18 (a) First we will determine the root graphically

```
>> x=-1:0.01:5;
>> f=x.^3+3.5.*x.^2-40;
>> plot(x,f);grid
```



The zoom in tool can be used several times to home in on the root. For example, as shown in the following plot, a real root appears to occur at $x = 2.567$:



(b) The roots function yields both real and complex roots:

```
>> a=[1 3.5 0 -40];
>> roots(a)
ans =
-3.0338 + 2.5249i
-3.0338 - 2.5249i
2.5676
```

7.19 (a) Excel Solver Solution: The 3 functions can be set up as a roots problems:

$$f_1(a, u, v) = a^2 - u^2 + 2v^2 = 0$$

$$f_2(a, u, v) = u + v - 2 = 0$$

$$f_3(a, u, v) = a^2 - 2a - u = 0$$

C11		=SUM(C7:C9)
A	B	C
1 Problem 7.19		
2		
3 a	1	
4 u	1	
5 v	1	
6 Function	Function ²	
7 Func1	2	4
8 Func2	0	0
9 Func3	-2	4
10		
11 Sum Square		8
12		
13		
14		
15		

Solver Parameters

Set Target Cell:

Equal To: Max Min Value of:

By Changing Cells:

Subject to the Constraints:

A	B	C
1 Problem 7.19		
2		
3 a	-0.48785879	
4 u	1.213986399	
5 v	0.78613415	
6 Function	Function ²	
7 Func1	0.00025561	6.534E-08
8 Func2	0.00012114	1.467E-08
9 Func3	-0.00026321	6.928E-08
10		
11 Sum Square		1.493E-07

Solver Results

Solver found a solution. All constraints and optimality conditions are satisfied.

Keep Solver Solution Restore Original Values

If you use initial guesses of $a = -1$, $u = 1$, and $v = -1$, the Solver finds another solution at $a = -1.6951$, $u = 6.2634$, and $v = -4.2636$

(b) Symbolic Manipulator Solution:

MATLAB:

```
>> syms a u v
>> S=solve(u^2-2*v^2-a^2,u+v-2,a^2-2*a-u);
>> double(S.a)
ans =
    3.0916 + 0.3373i
    3.0916 - 0.3373i
   -0.4879
   -1.6952

>> double(S.u)
ans =
    3.2609 + 1.4108i
    3.2609 - 1.4108i
    1.2140
    6.2641

>> double(S.v)
ans =
   -1.2609 - 1.4108i
   -1.2609 + 1.4108i
    0.7860
   -4.2641
```

Mathcad:

Problem 7.19 (Mathcad)

$$f(a, u, v) := u^2 - 2v^2 - a^2 \quad g(a, u, v) := u + v - 2 \quad h(a, u, v) := a^2 - 2a - u$$

$$a := -1 \quad u := 1 \quad v := 1 \quad \text{Initial Guesses}$$

Given

$$f(a, u, v) = 0 \quad g(a, u, v) = 0 \quad h(a, u, v) = 0$$

$$\text{Find}(a, u, v) = \begin{pmatrix} -0.4879 \\ 1.214 \\ 0.786 \end{pmatrix}$$

$$a := -1 \quad u := 6 \quad v := -4$$

Given

$$f(a, u, v) = 0 \quad g(a, u, v) = 0 \quad h(a, u, v) = 0$$

$$\text{Find}(a, u, v) = \begin{pmatrix} -1.6952 \\ 6.2641 \\ -4.2641 \end{pmatrix}$$

$$a := 3 + i \quad u := 3 + i \quad v := -1 - i$$

Given

$$f(a, u, v) = 0 \quad g(a, u, v) = 0 \quad h(a, u, v) = 0$$

$$\text{Find}(a, u, v) = \begin{pmatrix} 3.0916 + 0.3373i \\ 3.2609 + 1.4108i \\ -1.2609 - 1.4108i \end{pmatrix}$$

Therefore, we see that the two real-valued solutions for a , u , and v are $(-0.4879, 1.2140, 0.7860)$ and $(-1.6952, 6.2641, -4.2641)$. In addition, MATLAB and Mathcad also provide the complex solutions as well.

7.20 MATLAB can be used to determine the roots of the numerator and denominator:

```
>> n=[1 12.5 50.5 66];
>> roots(n)
ans =
-5.5000
-4.0000
-3.0000

>> d=[1 19 122 296 192];
>> roots(d)
ans =
-8.0000
-6.0000
-4.0000
-1.0000
```

The transfer function can be written as

$$G(s) = \frac{(s+5.5)(s+4)(s+3)}{(s+8)(s+6)(s+4)(s+1)}$$

7.21

```
function root = bisection(func,xl,xu,es,maxit)
% root = bisection(func,xl,xu,es,maxit):
%   uses bisection method to find the root of a function
% input:
%   func = name of function
%   xl, xu = lower and upper guesses
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root
```

```

if func(xl)*func(xu)>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<5, maxit=50; end %if maxit blank set to 50
if nargin<4, es=0.001; end %if es blank set to 0.001
% bisection
iter = 0;
xr = xl;
while (1)
    xrold = xr;
    xr = (xl + xu)/2;
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    test = func(xl)*func(xr);
    if test < 0
        xu = xr;
    elseif test > 0
        xl = xr;
    else
        ea = 0;
    end
    if ea <= es | iter >= maxit, break, end
end
root = xr;

```

The following is a MATLAB session that uses the function to solve Example 5.3 with $\varepsilon_s = 0.0001$.

```

>> fcd=inline('(9.81*68.1/cd)*(1-exp(-0.146843*cd))-40','cd');
>> format long
>> bisection(fcd,5,15,0.0001)
ans =
    14.80114936828613

```

7.22

```

function root = falsepos(func,xl,xu,es,maxit)
% falsepos(func,xl,xu,es,maxit):
%   uses the false position method to find the root of the function func
% input:
%   func = name of function
%   xl, xu = lower and upper guesses
%   es = (optional) stopping criterion (%) (default = 0.001)
%   maxit = (optional) maximum allowable iterations (default = 50)
% output:
%   root = real root

if func(xl)*func(xu)>0 %if guesses do not bracket a sign change
    error('no bracket') %display an error message and terminate
end
% default values
if nargin<5, maxit=50; end
if nargin<4, es=0.001; end
% false position
iter = 0;
xr = xl;
while (1)
    xrold = xr;

```

```

xr = xu - func(xu)*(xl - xu)/(func(xl) - func(xu));
iter = iter + 1;
if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
test = func(xl)*func(xr);
if test < 0
    xu = xr;
elseif test > 0
    xl = xr;
else
    ea = 0;
end
if ea <= es | iter >= maxit, break, end
end
root = xr;

```

The following is a MATLAB session that uses the function to solve Example 5.5:

```

>> fcd=inline('(9.81*68.1/cd)*(1-exp(-0.146843*cd))-40','cd');
>> format long
>> falsepos(fcd,5,15,0.0001)
ans =
    14.80114660933235

```

7.23

```

function root = newtraph(func,dfunc,xr,es,maxit)
% root = newtraph(func,dfunc,xguess,es,maxit):
%   uses Newton-Raphson method to find the root of a function
% input:
%   func = name of function
%   dfunc = name of derivative of function
%   xguess = initial guess
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

% if necessary, assign default values
if nargin<5, maxit=50; end      %if maxit blank set to 50
if nargin<4, es=0.001; end      %if es blank set to 0.001
% Newton-Raphson
iter = 0;
while (1)
    xrold = xr;
    xr = xr - func(xr)/dfunc(xr);
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
    if ea <= es | iter >= maxit, break, end
end
root = xr;

```

The following is a MATLAB session that uses the function to solve Example 6.3 with $\epsilon_s = 0.0001$.

```

>> format long
>> f=inline('exp(-x)-x','x');
>> df=inline('-exp(-x)-1','x');
>> newtraph(f,df,0)
ans =
    0.56714329040978

```

7.24

```

function root = secant(func,xrold,xr,es,maxit)
% secant(func,xrold,xr,es,maxit):
%   uses secant method to find the root of a function
% input:
%   func = name of function
%   xrold, xr = initial guesses
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

% if necessary, assign default values
if nargin<5, maxit=50; end      %if maxit blank set to 50
if nargin<4, es=0.001; end      %if es blank set to 0.001
% Secant method
iter = 0;
while (1)
    xrn = xr - func(xr)*(xrold - xr)/(func(xrold) - func(xr));
    iter = iter + 1;
    if xrn ~= 0, ea = abs((xr - xrn)/xrn) * 100; end
    if ea <= es | iter >= maxit, break, end
    xrold = xr;
    xr = xrn;
end
root = xrn;

```

Test by solving Example 6.6:

```

>> format long
>> f=inline('exp(-x)-x','x');
>> secant(f,0,1)
ans =
0.56714329040970

```

7.25

```

function root = modsec(func,xr,delta,es,maxit)
% modsec(func,xr,delta,es,maxit):
%   uses modified secant method to find the root of a function
% input:
%   func = name of function
%   xr = initial guess
%   delta = perturbation fraction
%   es = (optional) stopping criterion (%)
%   maxit = (optional) maximum allowable iterations
% output:
%   root = real root

% if necessary, assign default values
if nargin<5, maxit=50; end      %if maxit blank set to 50
if nargin<4, es=0.001; end      %if es blank set to 0.001
if nargin<3, delta=1E-5; end    %if delta blank set to 0.00001
% Secant method
iter = 0;
while (1)
    xrold = xr;
    xr = xr - delta*xr*func(xr) / (func(xr+delta*xr)-func(xr));
    iter = iter + 1;
    if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end

```

```
if ea <= es | iter >= maxit, break, end  
end  
root = xr;
```

Test by solving Example 6.8:

```
>> format long  
>> f=inline('exp(-x)-x','x');  
>> modsec(f,1,0.01)  
ans =  
0.56714329027265
```

CHAPTER 8

8.1 Ideal gas law:

$$v = \frac{RT}{p} = \frac{0.082054(400)}{2.5} = 13.12864$$

van der Waals equation:

Determine the root of

$$f(v) = \left(p + \frac{a}{v^2} \right)(v - b) - RT$$

$$f(v) = \left(2.5 + \frac{12.02}{v^2} \right)(v - 0.08407) - 0.082054(400)$$

Any of the techniques in Chaps 5 or 6 can be used to determine the root as $v = 12.8407$ L/mol. The Newton-Raphson method would be a good choice because (a) the equation is relatively simple to differentiate and (b) the ideal gas law provides a good initial guess. The Newton-Raphson method can be formulated as

$$v_{i+1} = v_i - \frac{\left(p + \frac{a}{v_i^2} \right)(v_i - b) - RT}{\left(p + \frac{a}{v_i^2} \right) - (v_i - b) \frac{2a}{v_i^3}}$$

Using the ideal gas law for the initial guess results in an accurate root determination in a few iterations:

<i>i</i>	<i>x_i</i>	<i>f(x_i)</i>	<i>f'(x_i)</i>	<i>ε_a</i>
0	13.12864	0.699518	2.431156	
1	12.84091	0.000441	2.428057	2.2407%
2	12.84073	1.84E-10	2.428055	0.0014%
3	12.84073	0	2.428055	0.0000%

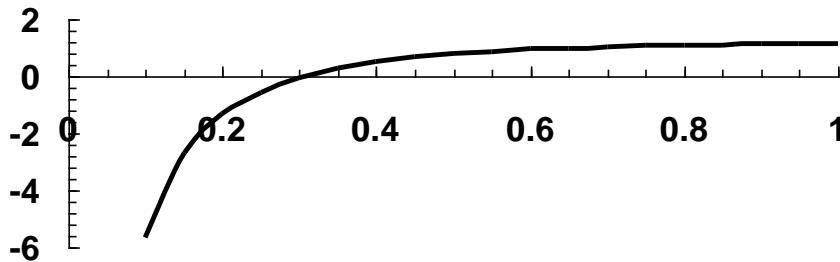
8.2 The function to be solved is

$$f(R) = \ln \frac{1 + R(1 - X_{Af})}{R(1 - X_{Af})} - \frac{R + 1}{R[1 + R(1 - X_{Af})]} = 0$$

or substituting $X_{Af} = 0.95$,

$$f(R) = \ln \frac{1+0.05R}{R(0.05)} - \frac{R+1}{R(1+0.05R)} = 0$$

A plot of the function indicates a root at about $R = 0.3$

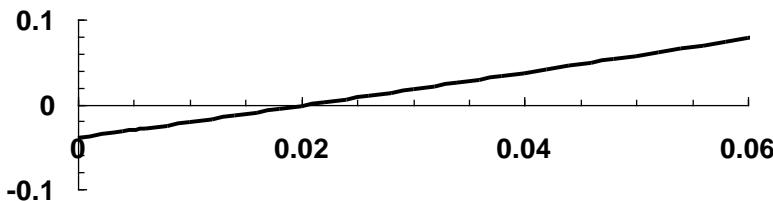


Bisection with initial guesses of 0.01 and 1 can be used to determine a root of 0.30715 after 16 iterations with $\varepsilon_a = 0.005\%$.

8.3 The function to be solved is

$$f(x) = \frac{x}{1-x} \sqrt{\frac{7}{2+x}} - 0.04 = 0$$

A plot of the function indicates a root at about $x = 0.02$.



Because the function is so linear, false position is a good choice. Using initial guesses of 0.01 and 0.03, the first iteration is

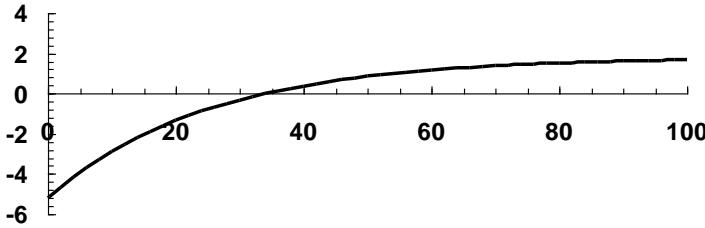
$$x_r = 0.03 - \frac{0.017432(0.01 - 0.03)}{-0.02115 - 0.017432} = 0.020964$$

After 3 iterations, the result is 0.021041 with $\varepsilon_a = 0.003\%$.

8.4 The function to be solved is

$$f(t) = 12(1 - e^{-0.04t}) + 5e^{-0.04t} - 10.2 = 0$$

A plot of the function indicates a root at about $t = 34$.

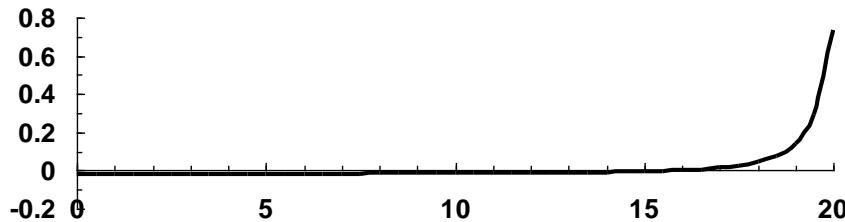


Bisection with initial guesses of 0 and 50 can be used to determine a root of 33.95309 after 16 iterations with $\epsilon_a = 0.002\%$.

8.5 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 $\epsilon_a = 0.493\%$. Note that false position would have required 68 iterations to attain comparable accuracy.

<i>i</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%

8.6 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

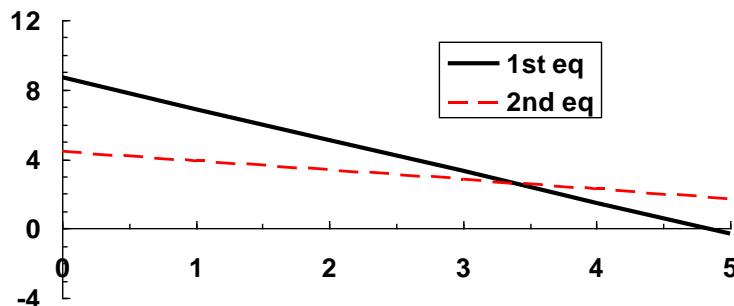
$$f_1(x_1, x_2) = \frac{5 + x_1 + x_2}{(50 - 2x_1 - x_2)^2(20 - x_1)} - 4 \times 10^{-4}$$

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

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

$$f_1(3.3, 2.7) = -2.36 \times 10^{-6}$$

$$f_2(3.3, 2.7) = 2.33 \times 10^{-5}$$

$$\frac{\partial f_1}{\partial x_1} = 9.9 \times 10^{-5} \quad \frac{\partial f_2}{\partial x_1} = 5.185 \times 10^{-3}$$

$$\frac{\partial f_1}{\partial x_2} = 5.57 \times 10^{-5} \quad \frac{\partial f_2}{\partial x_2} = 9.35 \times 10^{-3}$$

$$|J| = 6.37 \times 10^{-7}$$

$$x_1 = 3.3 - \frac{-2.36 \times 10^{-6} (9.35 \times 10^{-3}) - 2.33 \times 10^{-5} (5.57 \times 10^{-5})}{6.37 \times 10^{-7}} = 3.3367$$

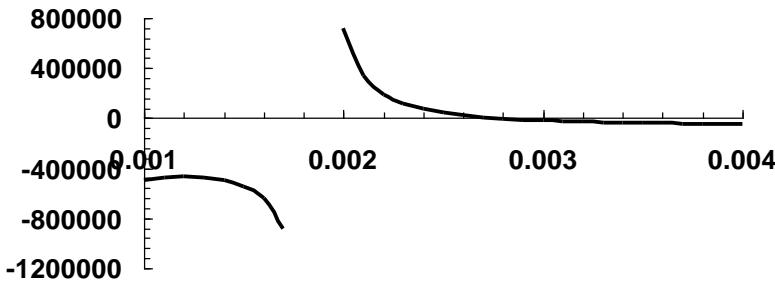
$$x_2 = 2.7 - \frac{2.33 \times 10^{-5} (9.9 \times 10^{-5}) - (-2.36 \times 10^{-6})(5.185 \times 10^{-3})}{6.37 \times 10^{-7}} = 2.677$$

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

8.7 Using the given values, $a = 12.6126$ and $b = 0.0018707$. Therefore, the roots problem to be solved is

$$f(v) = \frac{0.518(223)}{(v - 0.0018707)} - \frac{12.6126}{v(v + 0.0018707)\sqrt{223}} - 65000$$

A plot indicates a root at about 0.0028.

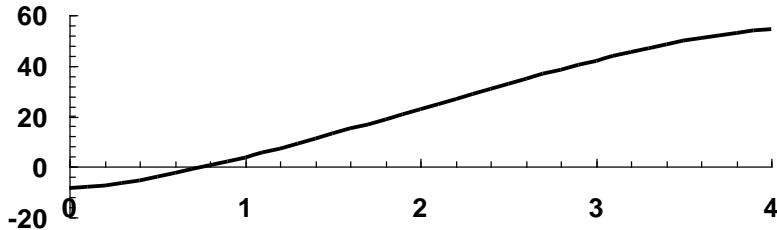


Using initial guesses of 0.002 and 0.004, bisection can be employed to determine the root as 0.00275 after 12 iterations with $\varepsilon_a = 0.018\%$. The mass of methane contained in the tank can be computed as $3/0.00275 = 1091$ kg.

8.8 Using the given values, the roots problem to be solved is

$$f(h) = \left[4 \cos^{-1}\left(\frac{2-h}{2}\right) - (2-h)\sqrt{4h-h^2} \right] 5 - 8.5 = 0$$

A plot indicates a root at about 0.8.

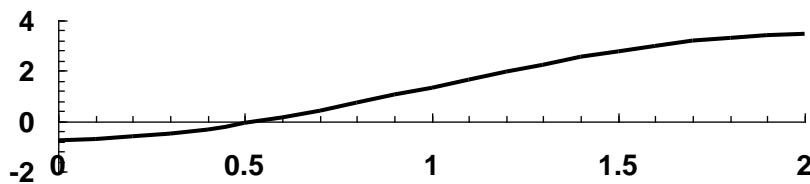


A numerical method can be used to determine that the root is 0.77194.

8.9 Using the given values, the roots problem to be solved is

$$f(h) = \frac{\pi h^2(3-h)}{3} - 0.75 = 0$$

A plot indicates a root at about 0.52.

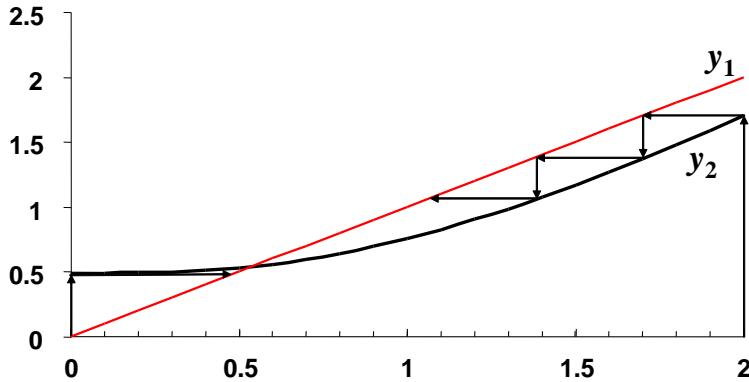


A numerical method can be used to determine that the root is 0.53952.

8.10 The best way to approach this problem is to use the graphical method displayed in Fig. 6.3. For the first version, we plot

$$y_1 = h \quad \text{and} \quad y_2 = \sqrt{\frac{h^3 + 0.7162}{3}}$$

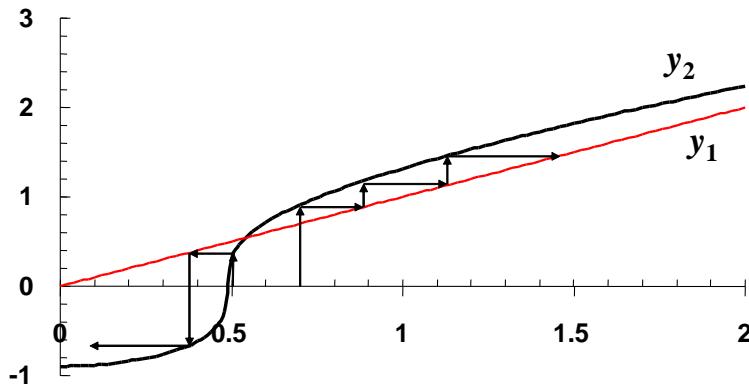
versus the range of h . Note that for the sphere, h ranges from 0 to $2r$. As displayed below, this version will always converge.



For the second version, we plot

$$y_1 = h \quad \text{and} \quad y_2 = \sqrt[3]{3h^2 - 0.7162}$$

versus the range of h . As displayed below, this version is not convergent.



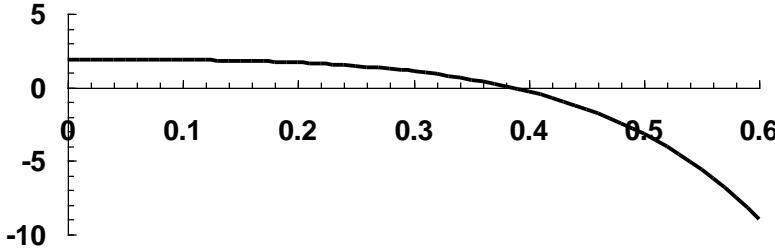
8.11 Substituting the parameter values yields

$$20 \frac{\varepsilon^3}{1-\varepsilon} = 150 \frac{1-\varepsilon}{1000} + 1.75$$

This can be rearranged and expressed as a roots problem

$$f(\varepsilon) = 0.15(1-\varepsilon) + 1.75 - 20 \frac{\varepsilon^3}{1-\varepsilon} = 0$$

A plot of the function suggests a root at about 0.38.



But suppose that we do not have a plot. How do we come up with a good initial guess? The void fraction (the fraction of the volume that is not solid; i.e. consists of voids) varies between 0 and 1. As can be seen, a value of 1 (which is physically unrealistic) causes a division by zero. Therefore, two physically-based initial guesses can be chosen as 0 and 0.99. Note that the zero is not physically realistic either, but since it does not cause any mathematical difficulties, it is OK. Applying bisection yields a result of $\varepsilon = 0.384211$ in 15 iterations with an absolute approximate relative error of $7.87 \times 10^{-3} \%$.

8.12 (a) The Reynolds number can be computed as

$$\text{Re} = \frac{\rho V D}{\mu} = \frac{1.23(40)0.005}{1.79 \times 10^{-5}} = 13743$$

In order to find f , we must determine the root of the function $g(f)$

$$g(f) = -2.0 \log \left(\frac{0.0000015}{3.7(0.005)} + \frac{2.51}{13743\sqrt{f}} \right) - \frac{1}{\sqrt{f}} = 0$$

As mentioned in the problem a good initial guess can be obtained from the Blasius formula

$$f = \frac{0.316}{13743^{0.25}} = 0.029185$$

Using this guess, a root of 0.028968 can be obtained with an approach like the modified secant method. This result can then be used to compute the pressure drop as

$$\Delta p = 0.028968 \frac{0.2(1.23)(40)^2}{2(0.005)} = 1140.17 \text{ Pa}$$

(b) For the rougher steel pipe, we must determine the root of

$$g(f) = -2.0 \log \left(\frac{0.000045}{3.7(0.005)} + \frac{2.51}{13743\sqrt{f}} \right) - \frac{1}{\sqrt{f}} = 0$$

Using the same initial guess as in **(a)**, a root of 0.04076 can be obtained. This result can then be used to compute the pressure drop as

$$\Delta p = 0.04076 \frac{0.2(1.23)(40)^2}{2(0.005)} = 1604.25 \text{ Pa}$$

Thus, as would be expected, the pressure drop is higher for the rougher pipe.

8.13 There are a variety of ways to solve this system of 5 equations

$$K_1 = \frac{[\text{H}^+][\text{HCO}_3^-]}{[\text{CO}_2]} \quad (1)$$

$$K_2 = \frac{[\text{H}^+][\text{CO}_3^{2-}]}{[\text{HCO}_3^-]} \quad (2)$$

$$K_w = [\text{H}^+][\text{OH}^-] \quad (3)$$

$$c_T = [\text{CO}_2] + [\text{HCO}_3^-] + [\text{CO}_3^{2-}] \quad (4)$$

$$\text{Alk} = [\text{HCO}_3^-] + 2[\text{CO}_3^{2-}] + [\text{OH}^-] - [\text{H}^+] \quad (5)$$

One way is to combine the equations to produce a single polynomial. Equations 1 and 2 can be solved for

$$[\text{CO}_2] = \frac{[\text{H}^+][\text{HCO}_3^-]}{K_1} \quad [\text{CO}_3^{2-}] = \frac{K_2[\text{HCO}_3^-]}{[\text{H}^+]} \quad (6)$$

These results can be substituted into Eq. 4, which can be solved for

$$[\text{CO}_2] = F_0 c_T \quad [\text{HCO}_3^-] = F_1 c_T \quad [\text{CO}_3^{2-}] = F_2 c_T$$

where F_0 , F_1 , and F_2 are the fractions of the total inorganic carbon in carbon dioxide, bicarbonate and carbonate, respectively, where

$$F_0 = \frac{[\text{H}^+]^2}{[\text{H}^+]^2 + K_1[\text{H}^+] + K_1 K_2} \quad F_1 = \frac{K_1[\text{H}^+]}{[\text{H}^+]^2 + K_1[\text{H}^+] + K_1 K_2} \quad F_2 = \frac{K_1 K_2}{[\text{H}^+]^2 + K_1[\text{H}^+] + K_1 K_2}$$

Now these equations, along with the Eq. 3 can be substituted into Eq. 5 to give

$$0 = F_1 c_T + 2F_2 c_T + K_w / [\text{H}^+] - [\text{H}^+] - \text{Alk}$$

Although it might not be apparent, this result is a fourth-order polynomial in $[\text{H}^+]$.

$$\begin{aligned} & [\text{H}^+]^4 + (K_1 + \text{Alk})[\text{H}^+]^3 + (K_1 K_2 + \text{Alk}K_1 - K_w - K_1 c_T)[\text{H}^+]^2 \\ & + (\text{Alk}K_1 K_2 - K_1 K_w - 2K_1 K_2 c_T)[\text{H}^+] - K_1 K_2 K_w = 0 \end{aligned}$$

Substituting parameter values gives

$$[\text{H}^+]^4 + 2.001 \times 10^{-3} [\text{H}^+]^3 - 5.012 \times 10^{-10} [\text{H}^+]^2 - 1.055 \times 10^{-19} [\text{H}^+] - 2.512 \times 10^{-31} = 0$$

This equation can be solved for $[\text{H}^+] = 2.51 \times 10^{-7}$ (pH = 6.6). This value can then be used to compute

$$\begin{aligned} [\text{OH}^-] &= \frac{10^{-14}}{2.51 \times 10^{-7}} = 3.98 \times 10^{-8} \\ [\text{CO}_2] &= \frac{(2.51 \times 10^{-7})^2}{(2.51 \times 10^{-7})^2 + 10^{-6.3}(2.51 \times 10^{-7}) + 10^{-6.3}10^{-10.3}} 3 \times 10^{-3} = 0.33304(3 \times 10^{-3}) = 0.001 \\ [\text{HCO}_3^-] &= \frac{10^{-6.3}(2.51 \times 10^{-7})}{(2.51 \times 10^{-7})^2 + 10^{-6.3}(2.51 \times 10^{-7}) + 10^{-6.3}10^{-10.3}} 3 \times 10^{-3} = 0.666562(3 \times 10^{-3}) = 0.002 \\ [\text{CO}_3^{2-}] &= \frac{10^{-6.3}10^{-10.3}}{(2.51 \times 10^{-7})^2 + 10^{-6.3}(2.51 \times 10^{-7}) + 10^{-6.3}10^{-10.3}} 3 \times 10^{-3} = 0.000133(3 \times 10^{-3}) = 4 \times 10^{-7} M \end{aligned}$$

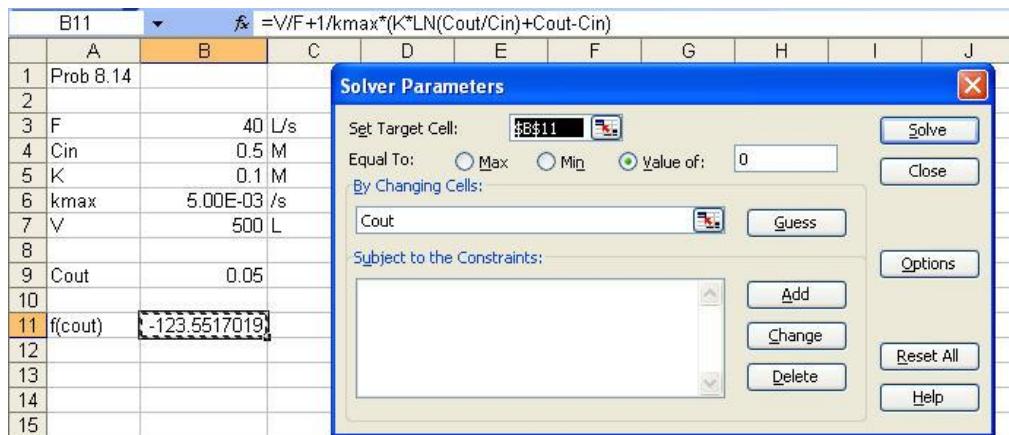
8.14 The integral can be evaluated as

$$-\int_{C_{\text{in}}}^{C_{\text{out}}} \frac{K}{k_{\max} C} + \frac{1}{k_{\max}} dC = -\frac{1}{k_{\max}} \left[K \ln\left(\frac{C_{\text{out}}}{C_{\text{in}}}\right) + C_{\text{out}} - C_{\text{in}} \right]$$

Therefore, the problem amounts to finding the root of

$$f(C_{\text{out}}) = \frac{V}{F} + \frac{1}{k_{\max}} \left[K \ln\left(\frac{C_{\text{out}}}{C_{\text{in}}}\right) + C_{\text{out}} - C_{\text{in}} \right]$$

Excel solver can be used to find the root:

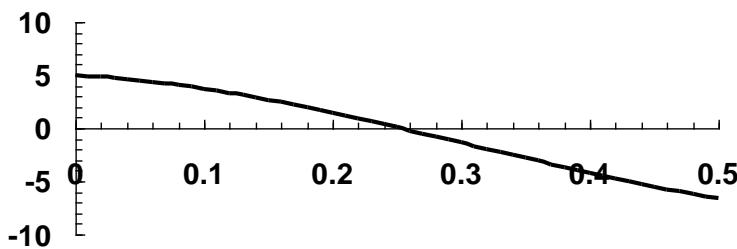


	A	B	C	D	E	F	G	H	I	J
1	Prob 8.14									
2										
3	F	40 L/s								
4	Cin	0.5 M								
5	K	0.1 M								
6	kmax	5.00E-03 /s								
7	V	500 L								
8										
9	Cout	0.448393658								
10										
11	f(cout)	6.89467E-07								

8.15 (a) The function to be solved is

$$f(t) = 9e^{-0.7t} \cos(4t) - 3.5$$

A plot of the function indicates a root at about $t = 0.25$



(b) The Newton-Raphson method can be set up as

$$t_{i+1} = t_i - \frac{9e^{-0.7t_i} \cos(4t_i) - 3.5}{-36e^{-0.7t_i} \sin(4t_i) - 6.3\cos(4t_i)e^{-0.7t_i}}$$

Using an initial guess of 0.3,

i	t	f(t)	f'(t)	ε _a
0	0.3	-0.85651	-29.0483	
1	0.270514	-0.00335	-28.7496	10.899824%
2	0.270398	-1.2E-07	-28.7476	0.043136%
3	0.270398	0	-28.7476	0.000002%

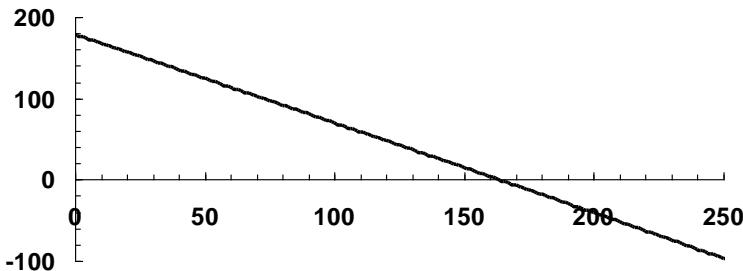
(c) The secant method can be implemented with initial guesses of 0.3,

i	t _{i-1}	f(t _{i-1})	t _i	f(t _i)	ε _a
0	0.2	1.951189	0.4	-3.69862	
1	0.4	-3.69862	0.269071	0.038125	48.66%
2	0.269071	0.038125	0.270407	-0.00026	0.49%
3	0.270407	-0.00026	0.270398	1.07E-07	0.0034%

8.16 The function to be solved is

$$f(P/A) = \frac{250}{1 + 0.4/\cos[25\sqrt{(P/A)/200,000}]} - \frac{P}{A}$$

A plot of the function indicates a root at about $P/A = 163$.

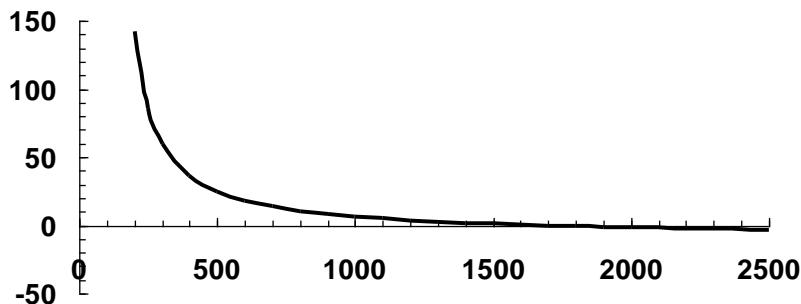


A numerical method can be used to determine that the root is 163.4429.

8.17 The function to be solved is

$$f(T_A) = \frac{T_A}{12} \cosh\left(\frac{600}{T_A}\right) + 6 - \frac{T_A}{12} - 15$$

A plot of the function indicates a root at about $T_A = 1700$.



A numerical method can be used to determine that the root is 1684.365.

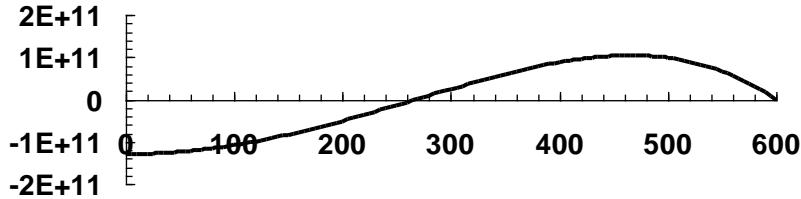
8.18 This problem can be solved by determining the root of the derivative of the elastic curve

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

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>i</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%

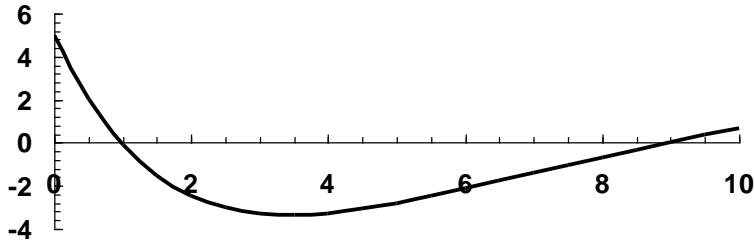
After 20 iterations, the root is determined as $x = 268.328$. This value can be substituted into Eq. (P8.18) to compute the maximum deflection as

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

8.19 (a) This problem can be solved by determining the root of

$$f(x) = 10 - 20(e^{-0.15x} - e^{-0.5x}) - 5 = 0$$

A plot of the function indicates a root at about $x = 1$ km.



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

<i>i</i>	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l) \times f(x_r)$	ε_a
1	0	5	2.5	5	-3.01569	-15.0784	

2	0	2.5	1.25	5	-0.87535	-4.37677	100.00%
3	0	1.25	0.625	5	1.422105	7.110527	100.00%
4	0.625	1.25	0.9375	1.422105	0.139379	0.198212	33.33%
5	0.9375	1.25	1.09375	0.139379	-0.39867	-0.05557	14.29%

After 10 iterations, the root is determined as $x = 0.971679688$ with an approximate error of 0.5%.

(b) The location of the minimum can be determined by differentiating the original function to yield

$$f'(x) = -0.15e^{-0.15x} + 0.5e^{-0.5x} = 0$$

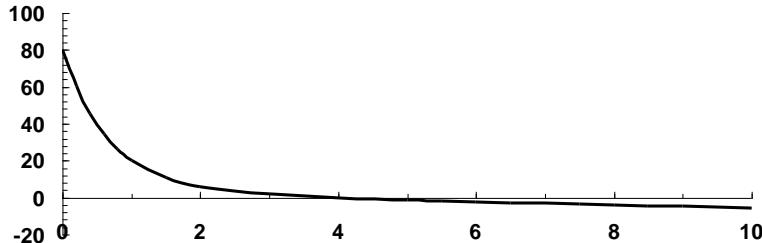
The root of this function can be determined as $x = 3.44$ km. The value of the minimum concentration can then be computed as

$$c = 10 - 20(e^{-0.15(3.44)} - e^{-0.5(3.44)}) = 1.6433$$

8.20 (a) This problem can be solved by determining the root of

$$f(t) = 75e^{-1.5t} + 20e^{-0.075t} - 15 = 0$$

A plot of the function indicates a root at about $t = 4$.



The Newton-Raphson method can be formulated as

$$t_{i+1} = t_i - \frac{75e^{-1.5t_i} + 20e^{-0.075t_i} - 15}{-112.5e^{-1.5t_i} - 1.5e^{-0.075t_i}}$$

Using the initial guess of $t = 6$, an accurate root determination can be obtained in a few iterations:

<i>i</i>	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	6	-2.23818	-0.97033	
1	3.693371	0.455519	-1.57879	62.45%
2	3.981896	0.02752	-1.39927	7.25%
3	4.001563	9.84E-05	-1.3893	0.49%

The result can be checked by substituting it back into the original equation to yield a prediction close to 15:

$$c = 75e^{-1.5(4.001563)} + 20e^{-0.075(4.001563)} = 15.0001$$

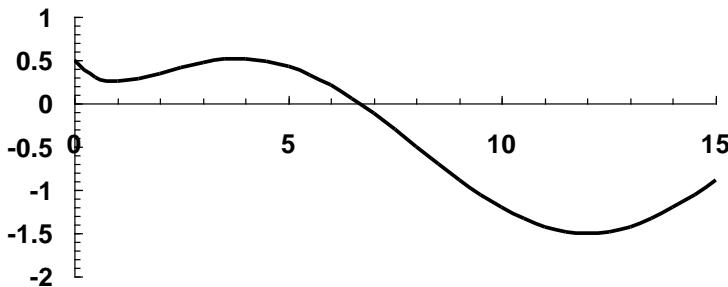
8.21 The solution can be formulated as

$$0.5 = \sin\left(\frac{2\pi x}{16}\right) \cos\left(\frac{2\pi(12)48}{16}\right) + e^{-x}$$

or

$$f(x) = \sin\left(\frac{\pi}{8}x\right) + e^{-x} - 0.5$$

A plot of this function suggests a root at about 6.7:

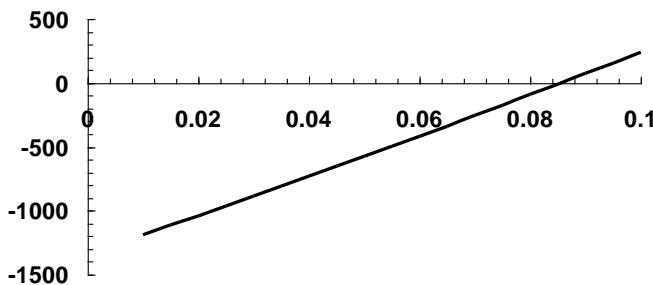


A numerical method can be used to determine that the root is 6.6704.

8.22 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:

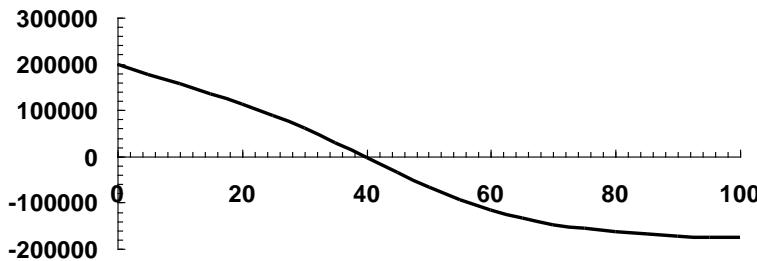


A numerical method can be used to determine that the root is 0.085595.

8.23 (a) The solution can be formulated as

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

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



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

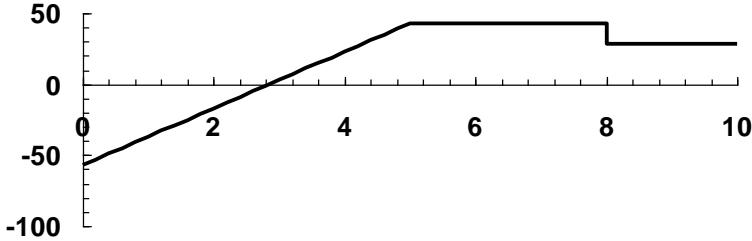
i	t_i	t_u	$f(t_i)$	$f(t_u)$	t_r	$f(t_r)$	$f(t_i) \times f(t_r)$	ϵ_a
1	0	100.0000	200000	-176110	53.1760	-84245	-1.685E+10	
2	0	53.1760	200000	-84245	37.4156	14442.8	2.889E+09	42.123%
3	37.4156	53.1760	14443	-84245	39.7221	-763.628	-1.103E+07	5.807%
4	37.4156	39.7221	14443	-763.628	39.6063	3.545288	5.120E+04	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 $135,142.5/112,618.7 = 1.2$.

8.24 First, we can generate a plot of the function:



Thus, a zero value occurs at approximately $x = 2.8$.

A numerical solution can be developed in a number of ways. Using MATLAB, we would first formulate an M-file for the shear function as:

```
function f = V(x)
f=20*(sing(x,0,1)-sing(x,5,1))-15*sing(x,8,0)-57;
```

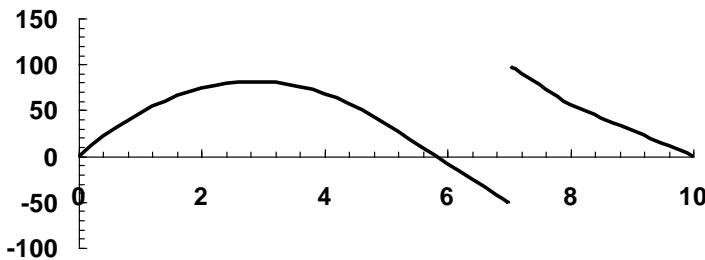
In addition, the singularity function can be set up as

```
function s = sing(x, a, n)
if x > a
    s = (x - a) ^ n;
else
    s = 0;
end
```

We can then either design our own M-file or use MATLAB's built-in capabilities like the `fzero` function. A session using the `fzero` function would yield a root of 2.85 as shown here,

```
>> x=fzero(@V, 5)
x =
2.8500
```

8.25 First, we can generate a plot of the moment function:



Thus, a zero value occurs at approximately $x = 5.8$.

A numerical solution can be developed in a number of ways. Using MATLAB, we would first formulate an M-file for the moment function as:

```
function f = Mx(x)
f=-10*(sing(x,0,2)-sing(x,5,2))+15*sing(x,8,1)+150*sing(x,7,0)+57*x;
```

In addition, the singularity function can be set up as

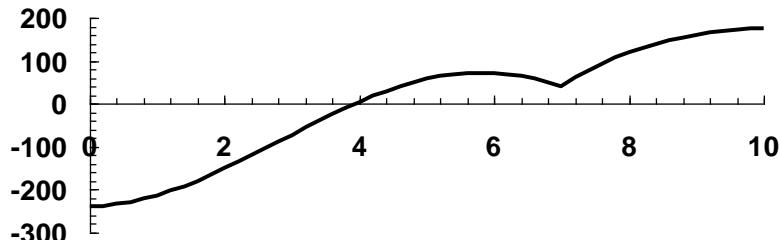
```
function s = sing(x, a, n)
if x > a
    s = (x - a) ^ n;
else
    s = 0;
end
```

We can then either design our own M-file implementing one of the numerical methods in the book or use MATLAB's built-in capabilities like the `fzero` function. A session using the `fzero` function would yield a root of 5.814 as shown here,

```
>> x=fzero (@Mx, 5)

x =
5.8140
```

8.26 First, we can generate a plot of the slope function:



Thus, a zero value occurs at approximately $x = 3.9$.

A numerical solution can be developed in a number of ways. Using MATLAB, we would first formulate an M-file for the slope function as:

```
function f = duydx(x)
f=-10/3*(sing(x,0,3)-sing(x,5,3))+7.5*sing(x,8,2)+150*sing(x,7,1)+57/2*x^2-238.25;
```

In addition, the singularity function can be set up as

```
function s = sing(x, a, n)
if x > a
    s = (x - a) ^ n;
else
    s = 0;
end
```

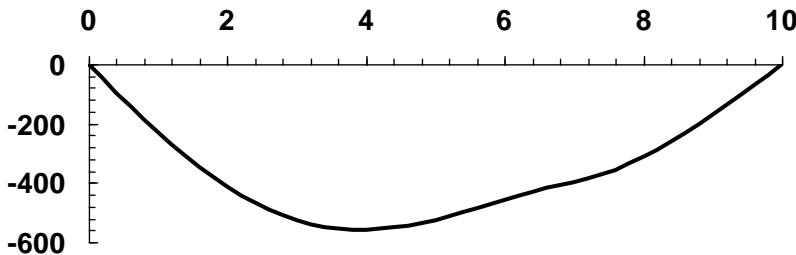
PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

We can then either design our own M-file implementing one of the numerical methods in the book or use MATLAB's built-in capabilities like the `fzero` function. A session using the `fzero` function would yield a root of 3.9357 as shown here,

```
>> x=fzero(@duydx, 5)

x =
3.9357
```

8.27 (a) First, we can generate a plot of the slope function:



Therefore, other than the end supports, there are no points of zero displacement.

(b) The location of the minimum can be determined by locating the zero of the slope function as described in Prob. 8.26.

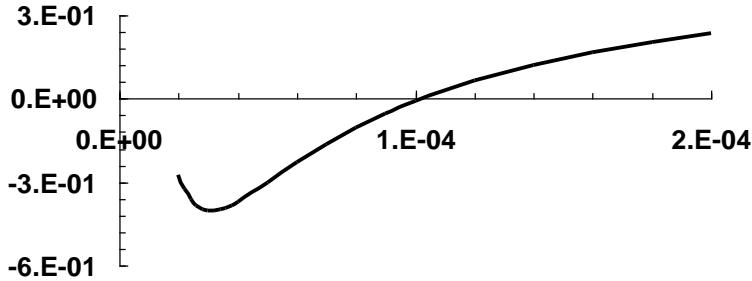
8.28 (a) The solution can be formulated as

$$f(C) = e^{-280(0.05)/(2(7.5))} \cos \left[\sqrt{\frac{1}{7.5C}} - \left(\frac{280}{2(7.5)} \right)^2 (0.05) \right] - 0.01$$

or

$$f(C) = 0.393241 \cos \left[\sqrt{\frac{1}{7.5C}} - 348.4444 (0.05) \right] - 0.01$$

A plot of this function indicates a root at about $C = 1 \times 10^{-4}$.



(b) Bisection:

<i>i</i>	C_l	C_u	C_r	$f(C_l)$	$f(C_r)$	$f(C_l) \times f(C_r)$	ε_a
1	5.0000E-05	1.5000E-04	1.0000E-04	-3.02E-01	-9.35E-03	0.002823	
2	1.0000E-04	1.5000E-04	1.2500E-04	-9.35E-03	8.00E-02	-0.00075	20.00%
3	1.0000E-04	1.2500E-04	1.1250E-04	-9.35E-03	3.88E-02	-0.00036	11.11%
4	1.0000E-04	1.1250E-04	1.0625E-04	-9.35E-03	1.57E-02	-0.00015	5.88%
5	1.0000E-04	1.0625E-04	1.0313E-04	-9.35E-03	3.44E-03	-3.2E-05	3.03%

After 14 iterations, the root is determined as 0.000102277 with an approximate error of 0.006%.

(c) In order to use MATLAB, we can first set up a function to hold the equation to be solved,

```
function f = prob0828(C)
t = 0.05; R = 280; L = 7.5; goal = 0.01;
f=exp(-R*t/(2*L))*cos(sqrt(1/(L*C)-(R/(2*L))^2)*t)-goal;
```

Here is the session that then determines the root,

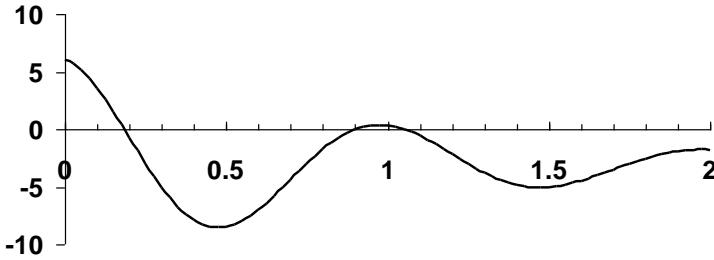
```
>> format long
>> fzero(@prob0828,1e-4)

ans =
1.022726852565315e-004
```

8.29 The solution can be formulated as

$$f(t) = 9e^{-t} \cos(2\pi t) - 3$$

A plot of this function indicates roots at about $t = 0.18, 0.9$ and 1.05 .



Using the Excel Solver and initial guesses of 0, 0.7 and 1 yields roots of $t = 0.184363099$, 0.903861928 , and 1.049482051 , respectively.

8.30 The solution can be formulated as

$$f(N) = 0 = \frac{2}{q(N + \sqrt{N^2 + 4n_i^2})\mu} - \rho$$

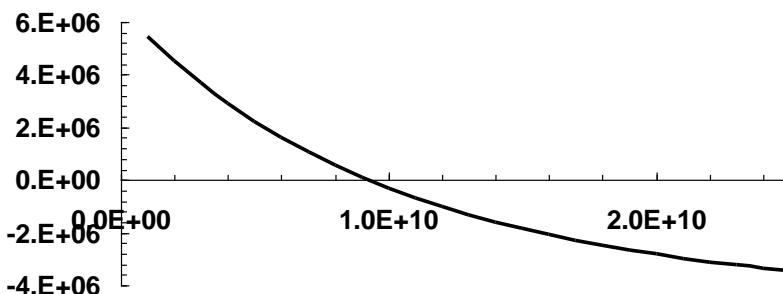
where

$$\mu = 1350 \left(\frac{1000}{300} \right)^{-2.42} = 73.2769$$

Substituting this value along with the other parameters gives

$$f(N) = 0 = \frac{2}{1.24571 \times 10^{-17} (N + \sqrt{N^2 + 1.54256 \times 10^{20}})} - 6.5 \times 10^6$$

A plot of this function indicates a root at about $N = 9 \times 10^9$.



(b) The bisection method can be implemented with the results for the first 5 iterations summarized as

<i>i</i>	<i>N_l</i>	<i>N_u</i>	<i>N_r</i>	<i>f(N)</i>	<i>f(N_r)</i>	<i>f(N)_×f(N_r)</i>	ε_a
1	5.000E+09	1.500E+10	1.000E+10	2.23E+06	-3.12E+05	-7E+11	

2	5.000E+09	1.000E+10	7.500E+09	2.23E+06	7.95E+05	1.77E+12	33.333%
3	7.500E+09	1.000E+10	8.750E+09	7.95E+05	2.06E+05	1.63E+11	14.286%
4	8.750E+09	1.000E+10	9.375E+09	2.06E+05	-6.15E+04	-1.3E+10	6.667%
5	8.750E+09	9.375E+09	9.063E+09	2.06E+05	6.99E+04	1.44E+10	3.448%

After 15 iterations, the root is 9.228×10^9 with a relative error of 0.003%.

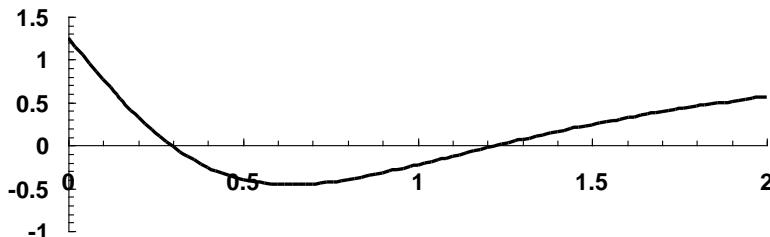
(c) The modified secant method (with $\delta = 0.01$) can be implemented with the results summarized as

i	N_i	$f(N_i)$	δN_i	$N_i + \delta N_i$	$f(N_i + \delta N_i)$	$f'(N_i)$	ϵ_a
0	9.000E+09	9.672E+04	9.000E+07	9.09E+09	5.819E+04	-0.0004	
1	9.226E+09	6.749E+02	9.226E+07	9.32E+09	-3.791E+04	-0.0004	2.449%
2	9.228E+09	-3.160E+00	9.228E+07	9.32E+09	-3.858E+04	-0.0004	0.017%
3	9.228E+09	1.506E-02	9.228E+07	9.32E+09	-3.858E+04	-0.0004	0.000%

8.31 Using the given values, the roots problem to be solved is

$$f(x) = 0 = 1.25 - 3.59672 \frac{x}{(x^2 + 0.81)^{3/2}}$$

A plot indicates roots at about 0.3 and 1.23.

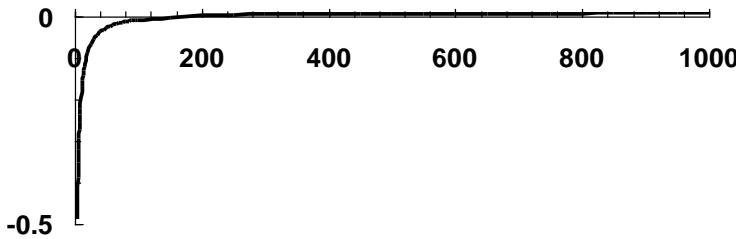


A numerical method can be used to determine that the roots are 0.295372 and 1.229573.

8.32 The solution can be formulated as

$$f(\omega) = 0 = 0.01333333 - \sqrt{1.97531 \times 10^{-5} + \left(6 \times 10^{-7} \omega - \frac{2}{\omega}\right)^2}$$

A plot of this function indicates a root at about $\omega = 150$.



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). This conclusion is borne out by the following results for bisection and false position.

(b) The bisection method can be implemented with the results for the first 5 iterations summarized as

i	ω_l	ω_u	ω_r	$f(\omega_l)$	$f(\omega_r)$	$f(\omega_l) \times f(\omega_r)$	ϵ_a
1	1	1000	500.5	-1.98667	0.007553	-0.01501	
2	1	500.5	250.75	-1.98667	0.004334	-0.00861	99.601%
3	1	250.75	125.875	-1.98667	-0.00309	0.006144	99.206%
4	125.875	250.75	188.3125	-0.00309	0.001924	-6E-06	33.156%
5	125.875	188.3125	157.0938	-0.00309	-6.2E-05	1.93E-07	19.873%

After 13 iterations, the root is 157.9474 with an approximate relative error of 0.077%.

(c) The false-position method can be implemented with the results for the first 5 iterations summarized as

i	ω_l	ω_u	$f(\omega_l)$	$f(\omega_u)$	ω_r	$f(\omega_r)$	$f(\omega_l) \times f(\omega_r)$	ϵ_a
1	1	1000.0	-1.98667	0.008674	995.7	0.00867	-0.01722	
2	1	995.7	-1.98667	0.00867	991.3	0.008667	-0.01722	0.436%
3	1	991.3	-1.98667	0.008667	987.0	0.008663	-0.01721	0.436%
4	1	987.0	-1.98667	0.008663	982.8	0.00866	-0.01720	0.436%
5	1	982.8	-1.98667	0.00866	978.5	0.008656	-0.01720	0.435%

After 578 iterations, the root is 189.4316 with an approximate error of 0.0998%. Note that the true error is actually about 20%. Therefore, the false position method is a very poor choice for this problem.

8.33 The solution can be formulated as

$$f(f) = 4 \log_{10} \left(\operatorname{Re} \sqrt{f} \right) - 0.4 - \frac{1}{\sqrt{f}}$$

We want our program to work for Reynolds numbers between 2,500 and 1,000,000.

Therefore, we must determine the friction factors corresponding to these limits. This can be done with any root location method to yield 0.011525 and 0.002913. Therefore, we can set

our initial guesses as $x_l = 0.0028$ and $x_u = 0.012$. Equation (5.5) can be used to determine the number of bisection iterations required to attain an absolute error less than 0.000005,

$$n = \log_2 \left(\frac{\Delta x^0}{E_{a,d}} \right) = \log_2 \left(\frac{0.012 - 0.0028}{0.000005} \right) = 10.8454$$

which can be rounded up to 11 iterations. Here is a VBA function that is set up to implement 11 iterations of bisection to solve the problem. Note that because VBA does not have a built-in function for the common logarithm, we have developed a user-defined function for this purpose.

```

Function Bisect(xl, xu, Re)
Dim xrold As Double, test As Double
Dim xr As Double, iter As Integer, ea As Double
Dim i As Integer
iter = 0
For i = 1 To 11
    xrold = xr
    xr = (xl + xu) / 2
    iter = iter + 1
    If xr <> 0 Then
        ea = Abs((xr - xrold) / xr) * 100
    End If
    test = f(xl, Re) * f(xr, Re)
    If test < 0 Then
        xu = xr
    ElseIf test > 0 Then
        xl = xr
    Else
        ea = 0
    End If
Next i
Bisect = xr
End Function

Function f(x, Re)
f = 4 * log10(Re * Sqr(x)) - 0.4 - 1 / Sqr(x)
End Function

Function log10(x)
log10 = Log(x) / Log(10)
End Function

```

This can be implemented in Excel. Here are the results for a number of values within the desired range. We have included the true value and the resulting error to verify that the results are within the desired error criterion of $E_a < 5 \times 10^{-6}$.

Re	Root	Truth	E_t
2500	0.011528320	0.011524764	3.56E-06
3000	0.010890430	0.010890229	2.01E-07
10000	0.007727930	0.007727127	8.02E-07

30000	0.005877148	0.005875048	2.10E-06
100000	0.004502539	0.004500376	2.16E-06
300000	0.003622070	0.003617895	4.18E-06
1000000	0.002912305	0.002912819	5.14E-07

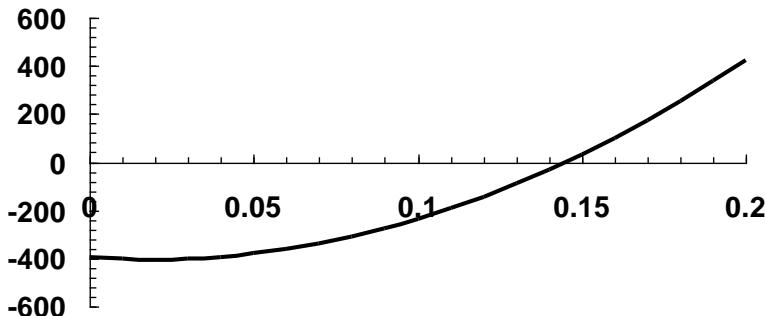
8.34 The solution can be formulated as

$$f(d) = 0 = \frac{2k_2 d^{5/2}}{5} + \frac{1}{2} k_1 d^2 - mgd - mgh$$

Substituting the parameter values gives

$$f(d) = 0 = 16d^{5/2} + 25,000d^2 - 882.9d - 397.305$$

A plot of this function indicates a root at about $d = 0.145$.



A numerical method can be used to determine that the root is $d = 1.44933$.

8.35 The solution can be formulated as

$$f(T) = 0 = -0.10597 + 1.671 \times 10^{-4} T + 9.7215 \times 10^{-8} T^2 - 9.5838 \times 10^{-11} T^3 + 1.9520 \times 10^{-14} T^4$$

MATLAB can be used to determine all the roots of this polynomial,

```
>> format long
>> x=[1.952e-14 -9.5838e-11 9.7215e-8 1.671e-4 -0.10597];
>> roots(x)

ans =
1.0e+003 *
2.74833708474921 + 1.12628559147229i
2.74833708474921 - 1.12628559147229i
-1.13102810059654
0.54408753765551
```

The only realistic value is 544.0875. This value can be checked using the `polyval` function,

```
>> polyval(x, 544.08753765551)
ans =
3.191891195797325e-016
```

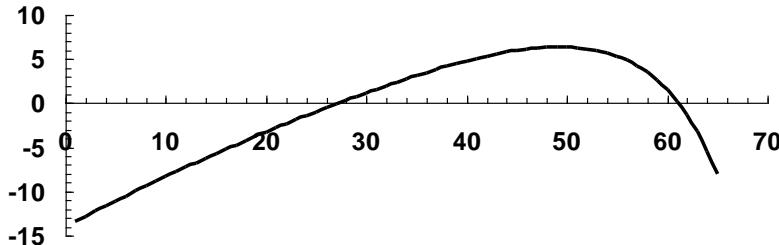
8.36 The solution can be formulated as

$$f(\theta_0) = (\tan \theta_0)x - \frac{g}{2v_0^2 \cos^2 \theta_0} x^2 + y_0 - y$$

Substituting the parameter values gives

$$f(\theta_0) = 0 = 35 \tan(\pi \theta_0 / 180) - \frac{15.0215625}{\cos^2(\pi \theta_0 / 180)} + 1$$

where θ_0 is expressed in degrees. A plot of this function indicates roots at about $\theta_0 = 27^\circ$ and 61° .



The Excel solver can then be used to determine the roots to higher accuracy. Using an initial guesses of 27° and 61° yields $\theta_0 = 27.2036^\circ$ and 61.1598° , respectively. Therefore, two angles result in the desired outcome. Note that the lower angle would probably be preferred as the ball would arrive at the catcher faster.

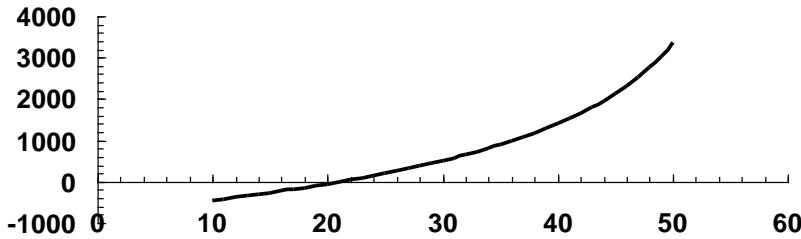
8.37 The solution can be formulated as

$$f(t) = u \ln \frac{m_0}{m_0 - qt} - gt - v$$

Substituting the parameter values gives

$$f(t) = 2,000 \ln \frac{150,000}{150,000 - 2,700t} - 9.81t - 750$$

A plot of this function indicates a root at about $t = 21$.



Because two initial guesses are given, a bracketing method like bisection can be used to determine the root,

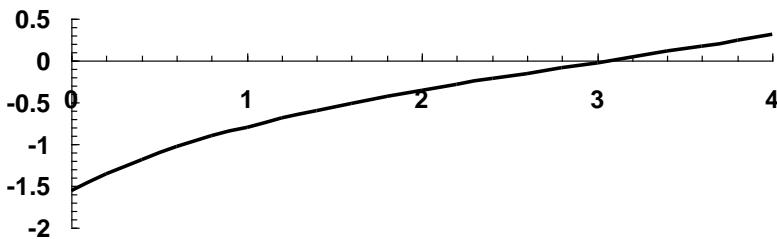
<i>i</i>	<i>t_l</i>	<i>t_u</i>	<i>t_r</i>	<i>f(t_l)</i>	<i>f(t_u)</i>	<i>f(t_l) × f(t_u)</i>	<i>ε_a</i>
1	10	50	30	-451.198	508.7576	-229550	
2	10	30	20	-451.198	-53.6258	24195.86	50.00%
3	20	30	25	-53.6258	200.424	-10747.9	20.00%
4	20	25	22.5	-53.6258	67.66275	-3628.47	11.11%
5	20	22.5	21.25	-53.6258	5.689921	-305.127	5.88%
6	20	21.25	20.625	-53.6258	-24.2881	1302.471	3.03%
7	20.625	21.25	20.9375	-24.2881	-9.3806	227.8372	1.49%
8	20.9375	21.25	21.09375	-9.3806	-1.8659	17.50322	0.74%

Thus, after 8 iterations, the approximate error falls below 1% with a result of $t = 21.09375$. Note that if the computation is continued, the root can be determined as 21.13242.

8.38 The solution can be formulated as

$$f(\omega) = \tan(\omega/3 - 1) - \frac{0.007158\omega}{1 - (\omega/34.119887)^2}$$

A plot of this function indicates a root at about $\omega = 3.1$.



A numerical method can be used to determine that the root is $\omega = 3.06637$.

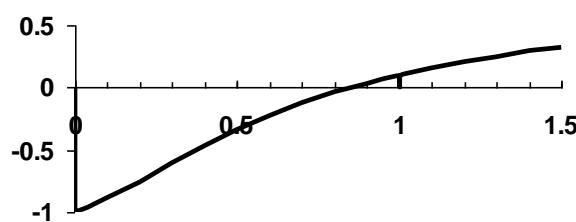
8.39 Excel Solver solution:

	A	B	C	D	E	F	G
1	Prob.	8.39					
2							
3	In:						
4	TA	400					
5	CpA	9.909	<-----	=3.381+0.01804*TA-0.0000043*TA^2			
6	FlowA	2					
7							
8	TB	700					
9	CpB	78.9098	<-----	=8.592+0.0129*TB-0.00004078*TB^2			
10	FlowB	1					
11							
12	Heatin	63164.06	<-----	=FlowA*CpA*TA+FlowB*CpB*TB			
13							
14	T	631.9315					
15	CpAout	13.06399	<-----	=3.381+0.01804*T-0.0000043*T^2			
16	CpBout	73.82618	<-----	=8.592+0.0129*T-0.00004078*T^2			
17							
18	Heatout:	63164.06	<-----	=FlowA*CpAout*T+FlowB*CpBout*T			
19							
20	Net	1.27E-07	<-----	=Heatin-Heatout			

8.40 The problem reduces to finding the value of n that drives the second part of the equation to 1. In other words, finding the root of

$$f(n) = \frac{n}{n-1} \left(R_c^{(n-1)/n} - 1 \right) - 1 = 0$$

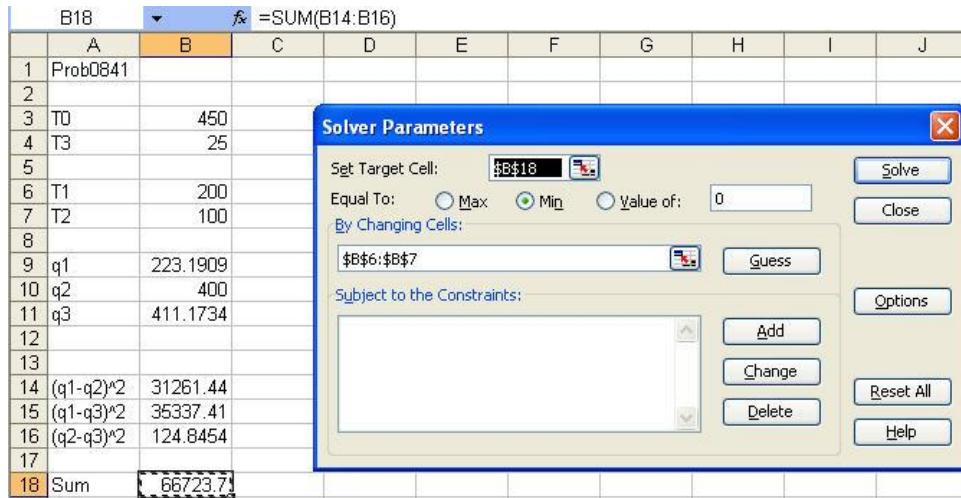
Inspection of the equation indicates that singularities occur at $x = 0$ and 1 . A plot indicates that otherwise, the function is smooth.



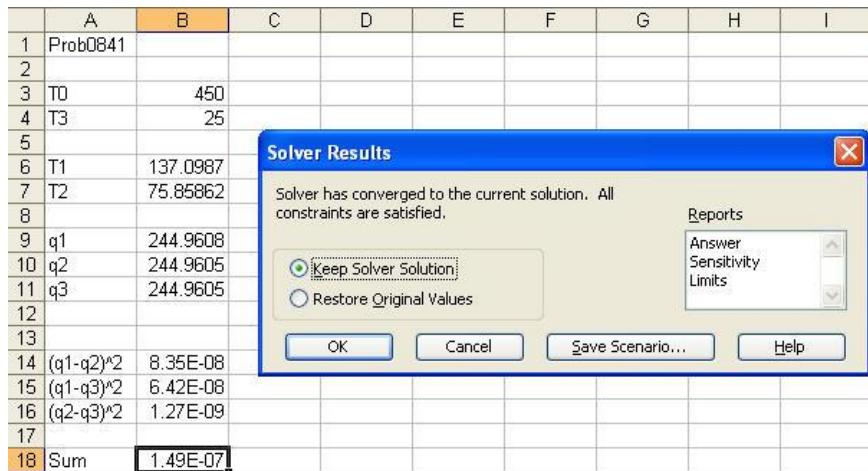
A tool such as the Excel Solver can be used to locate the root at $n = 0.8518$.

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

8.41 The following application of Excel Solver can be set up:



The solution is:



8.42 This problem was solved using the roots command in MATLAB.

```
c =
1           -33          -704         -1859
roots(c)

ans =
48.3543
-12.2041
-3.1502
```

Therefore,

$$\sigma_1 = 48.4 \text{ MPa} \quad \sigma_2 = -3.15 \text{ MPa} \quad \sigma_3 = -12.20 \text{ MPa}$$

8.43 For this problem, two continuity conditions must hold. First, the flows must balance,

$$Q_1 = Q_2 + Q_3 \quad (1)$$

Second, the energy balance must hold. That is, the head losses in pipes 1 and 3 must balance the elevation drop between reservoirs A and C,

$$H_{L,1} + H_{L,3} = E_A - E_C \quad (2)$$

The head losses for each pipe can be computed with

$$H_{L,i} = f_i \frac{L_i}{D_i} \frac{V_i^2}{2g} \quad (3)$$

The flows and velocities are related by the continuity equation, which for a circular pipe is

$$V_i = \frac{4Q_i}{\pi D_i^2} \quad (4)$$

Finally, the Colebrook equation relates the friction factor to the pipe characteristics as in

$$\frac{1}{\sqrt{f_i}} = -2.0 \log \left(\frac{\varepsilon_i}{3.7D_i} + \frac{2.51}{Re_i \sqrt{f_i}} \right)$$

where ε = the roughness (m), and Re = the *Reynolds number*,

$$Re_i = \frac{V_i D_i}{\nu}$$

where ν = kinematic viscosity (m^2/s).

These equations can be combined to reduce the problem to two equations with 2 unknowns. First, Eq. 4 can be solved for Q and substituted into Eq. 1 to give

$$\frac{\pi D_1^2}{4} V_1 - Q_2 - \frac{\pi D_3^2}{4} V_3 = 0 \quad (5)$$

Then, Eq. 3 can be substituted into Eq. 2 to yield

$$f_1 \frac{L_1}{2gD_1} V_1^2 + f_3 \frac{L_3}{2gD_3} V_3^2 - (E_A - E_C) = 0 \quad (6)$$

Therefore, if we knew the friction factors, these are two equations with two unknowns, V_1 and V_3 . If we could solve for these velocities, we could then determine the flows in pipes 1

and 3 via Eq. 4. Further, we could then determine the head losses in each pipe and compute the elevation of reservoir B as

$$E_B = E_A - H_{L,1} - H_{L,2} \quad (7)$$

There are a variety of ways to obtain the solution. One nice way is to use the Excel Solver. First the calculation can be set up as

	A	B	C	D	E	F
1						
2	e	0.0012				
3	g	9.81				
4	nu	1.00E-06				
5						
6	Reservoir					
7	ElevA	200	ElevB	181.147169	ElevC	172.5
8						
9	Parameters:					
10	L1	1800	L2	500	L3	1400
11	D1	0.4	D2	0.25	D3	0.2
12	V1	1	V2	2.03718327	V3	1
13						
14	Q1	0.125663706	Q2	0.1	Q3	0.03141593
15						
16	f1	0.026509711	f2	0.03019179	f3	0.03253974
17						
18	Re1	400000	Re2	509295.818	Re3	200000
19						
20	hL1	6.080208857	hL2	12.7726225	hL3	11.609491
21						
22	Res	Res^2				
23	Flow	-0.00575222	3.31E-05			
24	Head	-9.81030018	96.24199			
25						
26	Target Cell	96.24202				

The shaded cells are input data and the bold cells are the unknowns. The remaining cells are computed with formulas as outlined below. Note that we have named the cells so that the formulas are easier to understand.

	A	B	C	D	E	F
1						
2	e	0.0012				
3	g	9.81				
4	nu	0.000001				
5						
6	Reservoir					
7	ElevA	200	ElevB	=ElevA-hL1_-hL2_	ElevC	172.5
8						
9	Parameters:					
10	L1	1800	L2	500	L3	1400
11	D1	0.4	D2	0.25	D3	0.2
12	V1	1.12317013769616	V2	=4*Q2_-PI()/_D2_^-2	V3	1.30927138993214
13						
14	Q1	=PI()*_D1_^-2/4*_V1_-	Q2	0.1	Q3	=PI()*_D3_^-2/4*_V3_-
15						
16	f1	=ff(e,_D1_,B18)	f2	=ff(e,_D2_,D18)	f3	=ff(e,_D3_,F18)
17						
18	Re1	=D1_^-V1_/_nu	Re2	=D2_^-V2_/_nu	Re3	=D3_^-V3_/_nu
19						
20	hL1	=f1_*L1_/_D1_*V1_^-2/(2*g)	hL2	=f2_*L2_/_D2_*V2_^-2/(2*g)	hL3	=f3_*L3_/_D3_*V3_^-2/(2*g)
21						
22	Res	Res^2				
23	Flow	=Q1_-Q2_-Q3_-	=B23^2			
24	Head	=hL1_+hL3_-(ElevA-ElevC)	=B24^2			
25						
26	Target Cell	=SUM(C23:C24)				

Notice that we have set up the flow and head loss balances (Eqs. 5 and 6) in cells b23 and b24. We form a target cell (c26) as the summation of the squares of the balances (c23 and c24). It is this target cell that must be minimized to solve the problem.

An important feature of the solution is that we use a VBA worksheet function, ff, to solve for the friction factors in cells b16, d16 and f16. This function uses the modified secant method to solve the Colebrook equation for the friction factor. As shown below, it uses the Blasius formula to obtain a good initial guess:

```

Option Explicit

Function ff(e, D, Re)
Dim imax As Integer, iter As Integer
Dim es As Double, ea As Double
Dim xr As Double, xrold As Double, fr As Double
Const del As Double = 0.01
es = 0.01
imax = 20
'Blasius equation
xr = 0.316 / Re ^ 0.25
iter = 0
Do
    xrold = xr
    fr = f(xr, e, D, Re)
    xr = xr - fr * del * xr / (f(xr + del * xr, e, D, Re) - fr)
    iter = iter + 1
    If (xr <> 0) Then
        ea = Abs((xr - xrold) / xr) * 100
    End If
    If ea < es Or iter >= imax Then Exit Do
Loop
ff = xr
End Function

Function f(x, e, D, Re)
'Colebrook equation
f = -2 * Log(e / (3.7 * D) + 2.51 / Re / Sqr(x)) / Log(10) - 1 / Sqr(x)
End Function

```

The Excel Solver can then be used to drive the target cell to a minimum by varying the cells for V_1 (cell b12) and V_3 (cell f12).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	e		0.0012											
3	g		9.81											
4	nu		1.00E-06											
5														
6	Reservoir													
7	ElevA	200	ElevB	181.147169	ElevC	172.5								
8														
9	Parameters:													
10	L1	1800	L2		500	L3	1400							
11	D1	0.4	D2		0.25	D3	0.2							
12	V1	1	V2	2.03718327	V3	1								
13														
14	Q1	0.125663706	Q2		0.1	Q3	0.03141593							
15														
16	f1	0.026509711	f2	0.03019179	f3	0.03253974								
17														
18	Re1	400000	Re2	509295.818	Re3	200000								
19														
20	hL1	6.080208857	hL2	12.7726225	hL3	11.609491								
21														
22	Res		Res^2											
23	Flow	-0.00575222	3.31E-05											
24	Head	-9.81030018	96.24199											
25														
26	Target Cell		96.24202											



The results of Solver are shown below:

	A	B	C	D	E	F
1						
2	e		0.0012			
3	g		9.81			
4	nu		1.00E-06			
5						
6	Reservoir					
7	ElevA	200	ElevB	179.564187	ElevC	172.5
8						
9	Parameters:					
10	L1	1800	L2		500	L3
11	D1	0.4	D2		0.25	D3
12	V1	1.123442064	V2	2.03718327	V3	1.30915804
13						
14	Q1	0.141175893	Q2		0.1	Q3
15						
16	f1	0.026472486	f2	0.03019179	f3	0.03244092
17						
18	Re1	449376.8256	Re2	509295.818	Re3	261831.608
19						
20	hL1	7.66319003	hL2	12.7726225	hL3	19.8370151
21						
22	Res		Res^2			
23	Flow	4.74805E-05	2.25E-09			
24	Head	0.000205128	4.21E-08			
25						
26	Target Cell		4.43E-08			

Therefore, the solution is $V_1 = 1.12344$ and $V_3 = 1.309158$. Equation 4 can then be used to compute $Q_1 = 0.14118$ and $Q_3 = 0.041128$. Finally, Eq. 7 can be used to compute the elevation of reservoir B as 179.564.

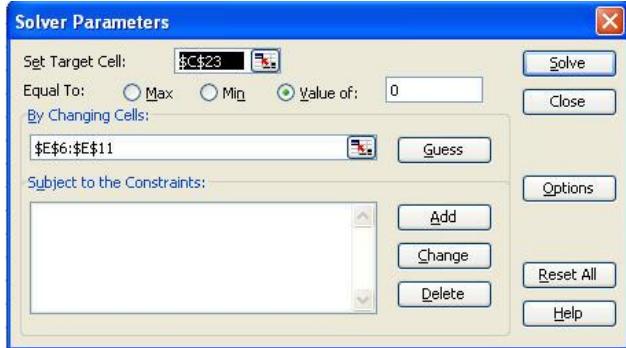
8.44 This problem can be solved in a number of ways. The following solution employs Excel and its Solver option. A worksheet is developed to solve for the pressure drop in each pipe and then determine the flow and pressure balances. Here is how the worksheet is set up,

	A	B	C	D	E	F	G	H
1	Flow1		1 m3/s					
2	rho		1.23 kg/m3					
3								
4	pipe	Length	Diameter	Area	Flow	Velocity	f	DeltaP
5	2	4	0.5	0.1963	0.5	2.546479	0.005	1.60E-01
6	3	2	0.5	0.1963	0.5	2.546479	0.005	7.98E-02
7	4	4	0.5	0.1963	0.5	2.546479	0.005	1.60E-01
8	5	2	0.5	0.1963	0.5	2.546479	0.005	7.98E-02
9	6	4	0.5	0.1963	0.5	2.546479	0.005	1.60E-01
10	7	8	0.5	0.1963	0.5	2.546479	0.005	3.19E-01
11	8	2	0.5	0.1963	0.5	2.546479	0.005	7.98E-02
12	9	2	0.5	0.1963	0.5	2.546479	0.005	7.98E-02
13								
14		Res	Res^2					
15	Flow1	0.00E+00	0.00E+00					
16	Flow2	-5.00E-01	2.50E-01					
17	Flow3	-5.00E-01	2.50E-01					
18	Head1	1.60E-01	2.54E-02					
19	Head2	1.60E-01	2.54E-02					
20	Head3	1.60E-01	2.54E-02					
21								
22		Target	5.76E-01					

The following shows the data and formulas that are entered into each cell.

	A	B	C	D	E	F	G	H
1	Flow1	1	m3/s					
2	rho	1.23	kg/m3					
3								
4	pipe	Length	Diameter	Area	Flow	Velocity	f	DeltaP
5	2	4	0.5	=PI()*(C5/2)^2	0.5	=ABS(E5/D5)	0.005	=16/PI()^2*G5*B5*rho/2/C5^5*E5^2
6	3	2	0.5	=PI()*(C6/2)^2	0.5	=ABS(E6/D6)	0.005	=16/PI()^2*G6*B6*rho/2/C6^5*E6^2
7	4	4	0.5	=PI()*(C7/2)^2	0.5	=ABS(E7/D7)	0.005	=16/PI()^2*G7*B7*rho/2/C7^5*E7^2
8	5	2	0.5	=PI()*(C8/2)^2	0.5	=ABS(E8/D8)	0.005	=16/PI()^2*G8*B8*rho/2/C8^5*E8^2
9	6	4	0.5	=PI()*(C9/2)^2	0.5	=ABS(E9/D9)	0.005	=16/PI()^2*G9*B9*rho/2/C9^5*E9^2
10	7	8	0.5	=PI()*(C10/2)^2	0.5	=ABS(E10/D10)	0.005	=16/PI()^2*G10*B10*rho/2/C10^5*E10^2
11	8	2	0.5	=PI()*(C11/2)^2	=E8	=ABS(E11/D11)	0.005	=16/PI()^2*G11*B11*rho/2/C11^5*E11^2
12	9	2	0.5	=PI()*(C12/2)^2	=E6	=ABS(E12/D12)	0.005	=16/PI()^2*G12*B12*rho/2/C12^5*E12^2
13								
14		Res	Res^2					
15	Flow1	=Flow1-E5-E6	=B15^2					
16	Flow2	=E6-E7-E8	=B16^2					
17	Flow3	=E8-E9-E10	=B17^2					
18	Head1	=H6+H12+H7-H5	=B18^2					
19	Head2	=H8+H11+H9-H7	=B19^2					
20	Head3	=H10-H9	=B20^2					
21								
22		Target	=SUM(C15:C20)					

Notice that we have set up the flow and pressure head loss balances in cells b16 through b21. We form a target cell (c23) as the summation of the squares of the residuals (c16 through c21). It is this target cell that must be minimized to solve the problem. The following shows how this was done with the Solver.



Here is the final result:

	A	B	C	D	E	F	G	H
1	Flow1		1 m ³ /s					
2	rho		1.23 kg/m ³					
3								
4	pipe	Length	Diameter	Area	Flow	Velocity	f	DeltaP
5	2	4	0.5	0.1963	0.5316	2.707495	0.005	1.80E-01
6	3	2	0.5	0.1963	0.4684	2.386504	0.005	7.00E-02
7	4	4	0.5	0.1963	0.2513	1.279907	0.005	4.03E-02
8	5	2	0.5	0.1963	0.217	1.105374	0.005	1.50E-02
9	6	4	0.5	0.1963	0.1272	0.647611	0.005	1.03E-02
10	7	8	0.5	0.1963	0.0899	0.457797	0.005	1.03E-02
11	8	2	0.5	0.1963	0.217	1.105374	0.005	1.50E-02
12	9	2	0.5	0.1963	0.4684	2.386504	0.005	7.00E-02
13								
14		Res	Res^2					
15	Flow1	-7.99E-06	6.38E-11					
16	Flow2	4.36E-05	1.90E-09					
17	Flow3	-6.55E-06	4.29E-11					
18	Head1	-4.28E-05	1.83E-09					
19	Head2	7.60E-05	5.78E-09					
20	Head3	-5.99E-06	3.59E-11					
21								
22		Target	9.66E-09					

8.45 This problem can be solved in a number of ways. The following solution employs Excel and its Solver option. A worksheet is developed to solve for the pressure drop in each pipe and then determine the flow and pressure drop balances. Here is how the worksheet is set up,

	A	B	C	D	E	F	G	H	I
1	Flow1		1 m3/s						
2	rho		1.23 kg/m3						
3	mu		1.79E-05 N s/m2						
4									
5	pipe	Length	Diameter	Area	Flow	Velocity	Re	f	DeltaP
6	2	4	0.5	0.1963	0.5000	2.5465	87491	0.004629	1.48E-01
7	3	2	0.5	0.1963	0.5000	2.5465	87491	0.004629	7.38E-02
8	4	4	0.5	0.1963	0.5000	2.5465	87491	0.004629	1.48E-01
9	5	2	0.5	0.1963	0.5000	2.5465	87491	0.004629	7.38E-02
10	6	4	0.5	0.1963	0.5000	2.5465	87491	0.004629	1.48E-01
11	7	8	0.5	0.1963	0.5000	2.5465	87491	0.004629	2.95E-01
12	8	2	0.5	0.1963	0.5000	2.5465	87491	0.004629	7.38E-02
13	9	2	0.5	0.1963	0.5000	2.5465	87491	0.004629	7.38E-02
14									
15		Res	Res^2						
16	Flow1	0.000E+00	0.00E+00						
17	Flow2	-5.00E-01	2.50E-01						
18	Flow3	-5.00E-01	2.50E-01						
19	Head1	1.48E-01	2.18E-02						
20	Head2	1.48E-01	2.18E-02						
21	Head3	1.48E-01	2.18E-02						
22									
23	Target	5.65E-01							

The following shows the data and formulas that are entered into each cell.

	A	B	C	D	E	F	G	H	I
1	Flow1	1	m3/s						
2	rho	1.23	kg/m3						
3	mu	0.0000179	N s/m2						
4									
5	pipe	Length	Diameter	Area	Flow	Velocity	Re	f	DeltaP
6	2	4	0.5	=PI()*(C6/2)^2	0.5	=ABS(E6/D6)	=rho*F6*C6/mu	=ff(G6)	=16/PI()^2*H6*B6*rho/2/C6^5*E6^2
7	3	2	0.5	=PI()*(C7/2)^2	0.5	=ABS(E7/D7)	=rho*F7*C7/mu	=ff(G7)	=16/PI()^2*H7*B7*rho/2/C7^5*E7^2
8	4	4	0.5	=PI()*(C8/2)^2	0.5	=ABS(E8/D8)	=rho*F8*C8/mu	=ff(G8)	=16/PI()^2*H8*B8*rho/2/C8^5*E8^2
9	5	2	0.5	=PI()*(C9/2)^2	0.5	=ABS(E9/D9)	=rho*F9*C9/mu	=ff(G9)	=16/PI()^2*H9*B9*rho/2/C9^5*E9^2
10	6	4	0.5	=PI()*(C10/2)^2	0.5	=ABS(E10/D10)	=rho*F10*C10/mu	=ff(G10)	=16/PI()^2*H10*B10*rho/2/C10^5*E10^2
11	7	8	0.5	=PI()*(C11/2)^2	0.5	=ABS(E11/D11)	=rho*F11*C11/mu	=ff(G11)	=16/PI()^2*H11*B11*rho/2/C11^5*E11^2
12	8	2	0.5	=PI()*(C12/2)^2	=E9	=ABS(E12/D12)	=rho*F12*C12/mu	=ff(G12)	=16/PI()^2*H12*B12*rho/2/C12^5*E12^2
13	9	2	0.5	=PI()*(C13/2)^2	=E7	=ABS(E13/D13)	=rho*F13*C13/mu	=ff(G13)	=16/PI()^2*H13*B13*rho/2/C13^5*E13^2
14									
15		Res	Res^2						
16	Flow1	=Flow1-E6-E7	=B16^2						
17	Flow2	=E7-E8-E9	=B17^2						
18	Flow3	=E9-E10-E11	=B18^2						
19	Head1	=I7+I13+B16	=B19^2						
20	Head2	=I9+I12+I10-I8	=B20^2						
21	Head3	=I11-I10	=B21^2						
22									
23	Target	=SUM(C16:C21)							

Notice that we have set up the flow and pressure head loss balances in cells b16 through b21. We form a target cell (c23) as the summation of the squares of the residuals (c16 through c21). It is this target cell that must be minimized to solve the problem.

An important feature of the solution is that we use a VBA worksheet function, ff, to solve for the friction factors in column h. This function uses the modified false position method to solve the von Karman equation for the friction factor.

Option Explicit

```
Function ff(Re)
Dim iter As Integer, imax As Integer
Dim il As Integer, iu As Integer
Dim xrold As Double, fl As Double, fu As Double, fr As Double
Dim xl As Double, xu As Double, es As Double
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

Dim xr As Double, ea As Double
xl = 0.00001
xu = 1
es = 0.01
imax = 40
iter = 0
f1 = f(xl, Re)
fu = f(xu, Re)
Do
    xrold = xr
    xr = xu - fu * (xl - xu) / (f1 - fu)
    fr = f(xr, Re)
    iter = iter + 1
    If xr <> 0 Then
        ea = Abs((xr - xrold) / xr) * 100
    End If
    If f1 * fr < 0 Then
        xu = xr
        fu = f(xu, Re)
        iu = 0
        il = il + 1
        If il >= 2 Then f1 = f1 / 2
    ElseIf f1 * fr > 0 Then
        xl = xr
        f1 = f(xl, Re)
        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
ff = xr
End Function

Function f(x, Re)
f = 4 * Log(Re * Sqr(x)) / Log(10) - 0.4 - 1 / Sqr(x)
End Function

```

The Excel Solver can then be used to drive the target cell to a minimum by varying the flows in cells e6 through e11.



Here is the final result:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

	A	B	C	D	E	F	G	H	I
1	Flow1		1 m3/s						
2	rho		1.23 kg/m3						
3	mu		1.79E-05 N s/m2						
4									
5	pipe	Length	Diameter	Area	Flow	Velocity	Re	f	DeltaP
6	2	4	0.5	0.1963	0.5411	2.7559	94687	0.004552	1.70E-01
7	3	2	0.5	0.1963	0.4588	2.3366	80278	0.004714	6.33E-02
8	4	4	0.5	0.1963	0.2512	1.2794	43958	0.005380	4.33E-02
9	5	2	0.5	0.1963	0.2075	1.0567	36306	0.005620	1.54E-02
10	6	4	0.5	0.1963	0.1239	0.6310	21681	0.006349	1.24E-02
11	7	8	0.5	0.1963	0.0836	0.4256	14622	0.007002	1.25E-02
12	8	2	0.5	0.1963	0.2075	1.0567	36306	0.005620	1.54E-02
13	9	2	0.5	0.1963	0.4588	2.3366	80278	0.004714	6.33E-02
14									
15		Res	Res^2						
16	Flow1	9.09E-05	8.26E-09						
17	Flow2	8.14E-05	6.62E-09						
18	Flow3	1.63E-05	2.65E-10						
19	Head1	-1.65E-04	2.73E-08						
20	Head2	-1.41E-05	2.00E-10						
21	Head3	3.94E-05	1.55E-09						
22									
23		Target	4.42E-08						

8.46 The horizontal and vertical components of the orbiter thruster can be computed as

$$F_H = T_S \sin \theta \quad F_V = T_S \cos \theta$$

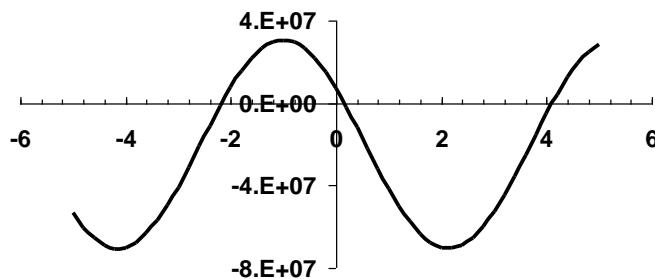
A moment balance about point G can be computed as

$$M = 4W_B - 4T_B - 24W_S + 24T_S \cos \theta - 38T_S \sin \theta$$

Substituting the parameter values yields

$$M = -20.068 \times 10^6 + 27 \times 10^6 \cos \theta - 42.75 \times 10^6 \sin \theta$$

This function can be plotted for the range of -5 to $+5$ radians



A valid root occurs at about 0.15 radians.

A MATLAB M-file called prob0846.m can be written to implement the Newton-Raphson method to solve for the root as

```
% Shuttle Liftoff Engine Angle
% Newton-Raphson Method of iteratively finding a single root
```

```

format long
% Constants
LGB = 4.0; LGS = 24.0; LTS = 38.0;
WS = 0.230E6; WB = 1.663E6;
TB = 5.3E6; TS = 1.125E6;
es = 0.5E-7; nmax = 200;
% Initial estimate in radians
x = 0.25
%Calculation loop
for i=1:nmax
    fx = LGB*WB-LGB*TB-LGS*WS+LGS*TS*cos(x)-LTS*TS*sin(x);
    dfx = -LGS*TS*sin(x)-LTS*TS*cos(x);
    xn=x-fx/dfx;
    %convergence check
    ea=abs((xn-x)/xn);
    if (ea<=es)
        fprintf('convergence: Root = %f radians \n',xn)
        theta = (180/pi)*x;
        fprintf('Engine Angle = %f degrees \n',theta)
        break
    end
    x=xn;
end

```

The program can be run with the result:

```

>> prob0846

x =
    0.150000000000000
x =
    0.15519036852630
x =
    0.15518449747863
convergence: Root = 0.155184 radians
Engine Angle = 8.891417 degrees

```

The program can be run for the case of the minimum payload, by changing W_s to 195,000 and running the M-file with the result:

```

>> prob0846

x =
    0.150000000000000
x =
    0.17333103912866
x =
    0.17321494968603
convergence: Root = 0.173215 radians
Engine Angle = 9.924486 degrees

```

CHAPTER 9

9.1

$$\begin{bmatrix} 0 & 2 & 5 \\ 1 & 0 & 1 \\ 8 & 3 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 50 \\ 10 \\ 20 \end{Bmatrix} \quad [A]^T = \begin{bmatrix} 0 & 1 & 8 \\ 2 & 0 & 3 \\ 5 & 1 & 0 \end{bmatrix}$$

9.2 (a) $[A] = 3 \times 2$ $[B] = 3 \times 3$ $[C] = 3 \times 1$ $[D] = 2 \times 4$
 $[E] = 3 \times 3$ $[F] = 2 \times 3$ $[G] = 1 \times 3$

(b) Square: $[B]$ and $[E]$

Column: $[C]$

Row: $[G]$

(c) $a_{12} = 7$ $b_{23} = 7$ $d_{32} = \text{does not exist}$
 $e_{22} = 2$ $f_{12} = 0$ $g_{12} = 6$

(d)

$$(1) [E] + [B] = \begin{bmatrix} 5 & 8 & 15 \\ 8 & 4 & 10 \\ 6 & 0 & 10 \end{bmatrix}$$

(2) $[A] + [F] = \text{not possible}$

$$(3) [B] - [E] = \begin{bmatrix} 3 & -2 & -1 \\ -6 & 0 & 4 \\ -2 & 0 & -2 \end{bmatrix}$$

$$(4) 7[B] = \begin{bmatrix} 28 & 21 & 49 \\ 7 & 14 & 49 \\ 14 & 0 & 28 \end{bmatrix}$$

$$(5) [E] \times [B] = \begin{bmatrix} 25 & 13 & 74 \\ 36 & 25 & 75 \\ 28 & 12 & 52 \end{bmatrix}$$

$$(6) \{C\}^T = \begin{bmatrix} 3 & 6 & 1 \end{bmatrix}$$

$$(7) [B] \times [A] = \begin{bmatrix} 54 & 76 \\ 41 & 53 \\ 28 & 38 \end{bmatrix}$$

$$(8) \{D\}^T = \begin{bmatrix} 9 & 2 \\ 4 & -1 \\ 3 & 7 \\ -6 & 5 \end{bmatrix}$$

(9) $[A] \times [C] = \text{not possible}$

(10) $[I] \times [B] = [B]$

$$(11) [E]^T [E] = \begin{bmatrix} 66 & 19 & 53 \\ 19 & 29 & 46 \\ 53 & 46 & 109 \end{bmatrix}$$

(12) $[C]^T [C] = 46$

9.3 (a) Possible multiplications:

$$[A][B] = \begin{bmatrix} 4 & 15 \\ 8 & 29 \\ 9 & 29 \end{bmatrix} \quad [A][C] = \begin{bmatrix} -16 & 4 \\ -24 & 4 \\ 2 & -10 \end{bmatrix} \quad [B][C] = \begin{bmatrix} -7 & 1 \\ -5 & 1 \end{bmatrix} \quad [C][B] = \begin{bmatrix} 1 & 2 \\ -2.5 & -7 \end{bmatrix}$$

Note: Some students might recognize that we can also compute $[B][B]$ and $[C][C]$:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$[B][B] = \begin{bmatrix} 2.5 & 9 \\ 1.5 & 5.5 \end{bmatrix} \quad [C][C] = \begin{bmatrix} 10 & -6 \\ -9 & 7 \end{bmatrix}$$

(b) $[B][A]$ and $[C][A]$ are impossible because the inner dimensions do not match:

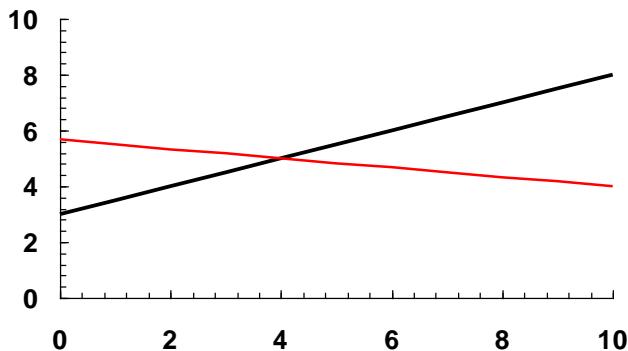
$$(2 \times 2) * (3 \times 2)$$

(c) According to **(a)**, $[B][C] \neq [C][B]$

9.4 The equations can be rearranged into a format for plotting x_2 versus x_1 :

$$x_2 = 3 + 0.5x_1$$

$$x_2 = \frac{34}{6} - \frac{1}{6}x_1$$



Therefore, the solution is $x_1 = 4$, $x_2 = 5$. The results can be checked by substituting them back into the original equations:

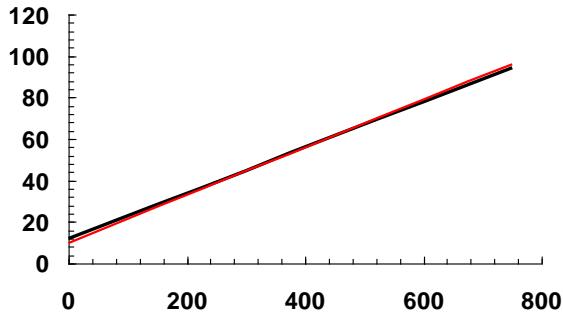
$$4(4) - 8(5) = -24$$

$$4 + 6(5) = 34$$

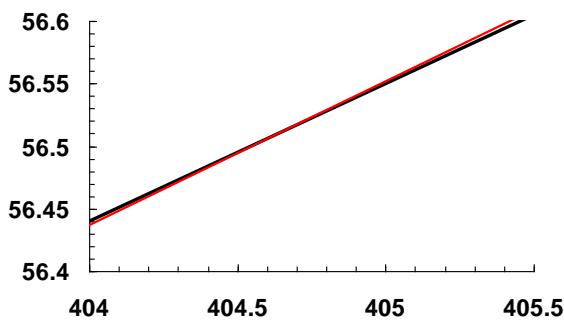
9.5 (a) The equations can be rearranged into a format for plotting x_2 versus x_1 :

$$x_2 = 12 + 0.11x_1$$

$$x_2 = 10 + \frac{2}{17.4}x_1$$



If you zoom in, it appears that there is a root at about (404.6, 56.5).



The results can be checked by substituting them back into the original equations:

$$-1.1(404.6) + 10(56.5) = 119.94 \approx 120$$

$$-2(404.6) + 17.4(56.5) = 173.9 \approx 174$$

(b) The plot suggests that the system may be ill-conditioned because the slopes are so similar.

(c) The determinant can be computed as

$$D = -1.1(17.4) - 10(-2) = 0.86$$

which is relatively small. Note that if the system is normalized first by dividing each equation by the largest coefficient,

$$-0.11x_1 + x_2 = 12$$

$$-0.11494x_1 + x_2 = 10$$

the determinant is even smaller

$$D = -0.11(1) - 1(-0.11494) = 0.00494$$

(d) Using Eqs. (9.10) and (9.11) yields

$$x_1 = \frac{17.4(120) - 10(174)}{0.86} = 404.6512$$

$$x_2 = \frac{-1.1(174) - (-2)(120)}{0.86} = 56.51163$$

9.6 (a) The determinant can be computed as:

$$A_1 = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} = 1(0) - 1(1) = -1$$

$$A_2 = \begin{vmatrix} 2 & 1 \\ 3 & 0 \end{vmatrix} = 2(0) - 1(3) = -3$$

$$A_3 = \begin{vmatrix} 2 & 1 \\ 3 & 1 \end{vmatrix} = 2(1) - 1(3) = -1$$

$$D = 0(-1) - 2(-3) + 5(-1) = 1$$

(b) Cramer's rule

$$x_1 = \frac{\begin{vmatrix} 9 & 2 & 5 \\ 9 & 1 & 1 \\ 10 & 1 & 0 \end{vmatrix}}{D} = \frac{6}{1} = 6$$

$$x_2 = \frac{\begin{vmatrix} 0 & 9 & 5 \\ 2 & 9 & 1 \\ 3 & 10 & 0 \end{vmatrix}}{D} = \frac{-8}{1} = -8$$

$$x_3 = \frac{\begin{vmatrix} 0 & 2 & 9 \\ 2 & 1 & 9 \\ 3 & 1 & 10 \end{vmatrix}}{D} = \frac{5}{1} = 5$$

(c) The results can be checked by substituting them back into the original equations:

$$2(-8) + 5(5) = 9$$

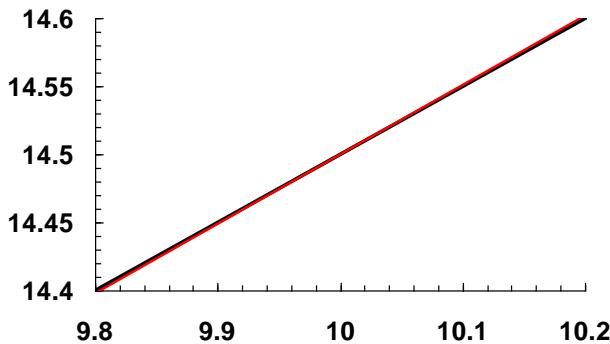
$$2(6) + (-8) + 5 = 9$$

$$3(6) + (-8) = 10$$

9.7 (a) The equations can be rearranged into a format for plotting x_2 versus x_1 :

$$x_2 = 9.5 + 0.5x_1$$

$$x_2 = 9.4 + 0.51x_1$$



The solution is $x_1 = 10$, $x_2 = 14.5$. Notice that the lines have very similar slopes.

(b) The determinant can be computed as

$$D = 0.5(-2) - (-1)1.02 = 0.02$$

(c) The plot and the low value of the determinant both suggest that the system is ill-conditioned.

(d) Using Eqs. (9.10) and (9.11) yields

$$x_1 = \frac{-9.5(-2) - (-1)(-18.8)}{0.02} = 10$$

$$x_2 = \frac{0.5(-18.8) - (-9.5)1.02}{0.02} = 14.5$$

(e) Using Eqs. (9.10) and (9.11) yields

$$x_1 = \frac{-9.5(-2) - (-1)(-18.8)}{-0.02} = -10$$

$$x_2 = \frac{0.52(-18.8) - (-9.5)1.02}{-0.02} = 4.3$$

The ill-conditioned nature of the system is illustrated by the fact that a small change in one of the coefficients results in a huge change in the results.

9.8 (a) The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 10 & 2 & -1 & 27 \\ -3 & -6 & 2 & -61.5 \\ 1 & 1 & 5 & -21.5 \end{bmatrix}$$

Forward elimination:

a_{21} is eliminated by multiplying row 1 by $-3/10$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $1/10$ and subtracting the result from row 3.

$$\begin{bmatrix} 10 & 2 & -1 & 27 \\ 0 & -5.4 & 1.7 & -53.4 \\ 0 & 0.8 & 5.1 & -24.2 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $0.8/(-5.4)$ and subtracting the result from row 3.

$$\begin{bmatrix} 10 & 2 & -1 & 27 \\ 0 & -5.4 & 1.7 & -53.4 \\ 0 & 0 & 5.351852 & -32.1111 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{-32.1111}{5.351852} = -6$$

$$x_2 = \frac{-53.4 - 1.7(-6)}{-5.4} = 8$$

$$x_1 = \frac{27 - (-1)(-6) - 2(8)}{10} = 0.5$$

(b) Check:

$$10(0.5) + 2(8) - (-6) = 27$$

$$-3(0.5) - 6(8) + 2(-6) = -61.5$$

$$0.5 + 8 + 5(-6) = -21.5$$

9.9 (a) The system is first expressed as an augmented matrix:

$$\left[\begin{array}{cccc} 8 & 2 & -2 & -2 \\ 10 & 2 & 4 & 4 \\ 12 & 2 & 2 & 6 \end{array} \right]$$

Forward elimination: First, we pivot by switching rows 1 and 3:

$$\begin{bmatrix} 12 & 2 & 2 & 6 \\ 10 & 2 & 4 & 4 \\ 8 & 2 & -2 & -2 \end{bmatrix}$$

Multiply row 1 by $10/12 = 0.83333$ and subtract from row 2 to eliminate a_{21} . Multiply row 1 by $8/12 = 0.66667$ and subtract from row 3 to eliminate a_{31} .

$$\begin{bmatrix} 12 & 2 & 2 & 6 \\ 0 & 0.33333 & 2.33333 & -1 \\ 0 & 0.66667 & -3.33333 & -6 \end{bmatrix}$$

Pivot:

$$\begin{bmatrix} 12 & 2 & 2 & 6 \\ 0 & 0.66667 & -3.33333 & -6 \\ 0 & 0.33333 & 2.33333 & -1 \end{bmatrix}$$

Multiply row 2 by $0.33333/0.66667 = 0.5$ and subtract from row 3 to eliminate a_{32} .

$$\begin{bmatrix} 12 & 2 & 2 & 6 \\ 0 & 0.66667 & -3.33333 & -6 \\ 0 & 0 & 4 & 2 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{2}{4} = 0.5$$

$$x_2 = \frac{-6 - (-3.3333)0.5}{0.66667} = -6.5$$

$$x_1 = \frac{6 - 2(0.5) - 2(-6.5)}{12} = 1.5$$

Check:

$$8(1.5) + 2(-6.5) - 2(0.5) = -2$$

$$10(1.5) + 2(-6.5) + 4(0.5) = 4$$

$$12(1.5) + 2(-6.5) + 2(0.5) = 6$$

9.10 (a) The determinant can be computed as:

$$A_1 = \begin{vmatrix} 2 & -1 \\ -2 & 0 \end{vmatrix} = 2(0) - (-1)(-2) = -2$$

$$A_2 = \begin{vmatrix} 1 & -1 \\ 5 & 0 \end{vmatrix} = 1(0) - (-1)(5) = 5$$

$$A_3 = \begin{vmatrix} 1 & 2 \\ 5 & -2 \end{vmatrix} = 1(-2) - 2(5) = -12$$

$$D = 0(-2) - (-3)5 + 7(-12) = -69$$

(b) Cramer's rule

$$x_1 = \frac{\begin{vmatrix} 2 & -3 & 7 \\ 3 & 2 & -1 \\ 2 & -2 & 0 \end{vmatrix}}{D} = \frac{-68}{-69} = 0.985507$$

$$x_2 = \frac{\begin{vmatrix} 0 & 2 & 7 \\ 1 & 3 & -1 \\ 5 & 2 & 0 \end{vmatrix}}{D} = \frac{-101}{-69} = 1.463768$$

$$x_3 = \frac{\begin{vmatrix} 0 & -3 & 2 \\ 1 & 2 & 3 \\ 5 & -2 & 2 \end{vmatrix}}{D} = \frac{-63}{-69} = 0.913043$$

(c) The system is first expressed as an augmented matrix:

$$\left[\begin{array}{cccc} 0 & -3 & 7 & 2 \\ 1 & 2 & -1 & 3 \\ 5 & -2 & 0 & 2 \end{array} \right]$$

Forward elimination: First, we pivot by switching rows 1 and 3:

$$\left[\begin{array}{cccc} 5 & -2 & 0 & 2 \\ 1 & 2 & -1 & 3 \\ 0 & -3 & 7 & 2 \end{array} \right]$$

Multiply row 1 by $1/5 = 0.2$ and subtract from row 2 to eliminate a_{21} . Because a_{31} already equals zero, it does not have to be eliminated.

$$\left[\begin{array}{cccc} 5 & -2 & 0 & 2 \\ 0 & 2.4 & -1 & 2.6 \\ 0 & -3 & 7 & 2 \end{array} \right]$$

Pivot:

$$\begin{bmatrix} 5 & -2 & 0 & 2 \\ 0 & -3 & 7 & 2 \\ 0 & 2.4 & -1 & 2.6 \end{bmatrix}$$

Multiply row 2 by $2.4/(-3) = -0.8$ and subtract from row 3 to eliminate a_{32} .

$$\begin{bmatrix} 5 & -2 & 0 & 2 \\ 0 & -3 & 7 & 2 \\ 0 & 0 & 4.6 & 4.2 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{4.2}{4.6} = 0.913043$$

$$x_2 = \frac{2 - 7(0.913043)}{-3} = 1.463768$$

$$x_1 = \frac{2 - 0(0.913043) - (-2)(1.463768)}{5} = 0.985507$$

(d) Check:

$$-3(1.463768) + 7(0.913043) = 2$$

$$(0.985507) + 2(1.463768) - (0.913043) = 3$$

$$5(0.985507) - 2(1.463768) = 2$$

9.11 (a) The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 2 & -6 & -1 & -38 \\ -3 & -1 & 7 & -34 \\ -8 & 1 & -2 & -20 \end{bmatrix}$$

Forward elimination: First, we pivot by switching rows 1 and 3:

$$\begin{bmatrix} -8 & 1 & -2 & -20 \\ -3 & -1 & 7 & -34 \\ 2 & -6 & -1 & -38 \end{bmatrix}$$

Multiply row 1 by $-3/(-8) = 0.375$ and subtract from row 2 to eliminate a_{21} . Multiply row 1 by $2/(-8) = -0.25$ and subtract from row 3 to eliminate a_{31} .

$$\begin{bmatrix} -8 & 1 & -2 & -20 \\ 0 & -1.375 & 7.75 & -26.5 \\ 0 & -5.75 & -1.5 & -43 \end{bmatrix}$$

Pivot:

$$\begin{bmatrix} -8 & 1 & -2 & -20 \\ 0 & -5.75 & -1.5 & -43 \\ 0 & -1.375 & 7.75 & -26.5 \end{bmatrix}$$

Multiply row 2 by $-1.375/-5.75 = 0.23913$ and subtract from row 3 to eliminate a_{32} .

$$\begin{bmatrix} -8 & 1 & -2 & -20 \\ 0 & -5.75 & -1.5 & -43 \\ 0 & 0 & 8.108696 & -16.2174 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{-16.2174}{8.108696} = -2$$

$$x_2 = \frac{-43 - (-1.5)(-2)}{-5.75} = 8$$

$$x_1 = \frac{-20 + 2(-2) - 1(8)}{-8} = 4$$

(b) Check:

$$2(4) - 6(8) - (-2) = -38$$

$$-3(4) - (8) + 7(-2) = -34$$

$$-8(4) + (8) - 2(-2) = -20$$

9.12 The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 2 & 1 & -1 & 1 \\ 5 & 2 & 2 & -4 \\ 3 & 1 & 1 & 5 \end{bmatrix}$$

Normalize the first row and then eliminate a_{21} and a_{31} ,

$$\begin{bmatrix} 1 & 0.5 & -0.5 & 0.5 \\ 0 & -0.5 & 4.5 & -6.5 \\ 0 & -0.5 & 2.5 & 3.5 \end{bmatrix}$$

Normalize the second row and eliminate a_{12} and a_{32} ,

$$\begin{bmatrix} 1 & 0 & 4 & -6 \\ 0 & 1 & -9 & 13 \\ 0 & 0 & -2 & 10 \end{bmatrix}$$

Normalize the third row and eliminate a_{13} and a_{23} ,

$$\begin{bmatrix} 1 & 0 & 0 & 14 \\ 0 & 1 & 0 & -32 \\ 0 & 0 & 1 & -5 \end{bmatrix}$$

Thus, the answers are $x_1 = 14$, $x_2 = -32$, and $x_3 = -5$.

Check:

$$2(14) + (-32) - (-5) = 1$$

$$5(14) + 2(-32) + 2(-5) = -4$$

$$3(14) + (-32) + (-5) = 5$$

9.13 (a) The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{bmatrix}$$

Forward elimination:

a_{21} is eliminated by multiplying row 1 by $6/1 = 6$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $-3/1 = -3$ and subtracting the result from row 3.

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 7 & -2 & -8 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $7/(-4) = -1.75$ and subtracting the result from row 3.

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 0 & 12 & 27 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{27}{12} = 2.25$$

$$x_2 = \frac{20 - 8(2.25)}{-4} = -0.5$$

$$x_1 = \frac{-3 - (-1)(2.25) - 1(-0.5)}{1} = -0.25$$

(b) The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{bmatrix}$$

Forward elimination: First, we pivot by switching rows 1 and 2:

$$\begin{bmatrix} 6 & 2 & 2 & 2 \\ 1 & 1 & -1 & -3 \\ -3 & 4 & 1 & 1 \end{bmatrix}$$

Multiply row 1 by $1/6 = 0.16667$ and subtract from row 2 to eliminate a_{21} . Multiply row 1 by $-3/6 = -0.5$ and subtract from row 3 to eliminate a_{31} .

$$\begin{bmatrix} 6 & 2 & 2 & 2 \\ 0 & 0.66667 & -1.33333 & -3.33333 \\ 0 & 5 & 2 & 2 \end{bmatrix}$$

Pivot:

$$\begin{bmatrix} 6 & 2 & 2 & 2 \\ 0 & 5 & 2 & 2 \\ 0 & 0.66667 & -1.33333 & -3.33333 \end{bmatrix}$$

Multiply row 2 by $0.66667/5 = 0.133333$ and subtract from row 3 to eliminate a_{32} .

$$\begin{bmatrix} 6 & 2 & 2 & 2 \\ 0 & 5 & 2 & 2 \\ 0 & 0 & -1.6 & -3.6 \end{bmatrix}$$

Back substitution:

$$x_3 = \frac{-3.6}{1.6} = 2.25$$

$$x_2 = \frac{2 - 2(2.25)}{5} = -0.5$$

$$x_1 = \frac{2 - 2(2.25) - 2(-0.5)}{6} = -0.25$$

(c) The system is first expressed as an augmented matrix:

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 6 & 2 & 2 & 2 \\ -3 & 4 & 1 & 1 \end{bmatrix}$$

Normalize the first row, and then eliminate a_{21} and a_{31} ,

$$\begin{bmatrix} 1 & 1 & -1 & -3 \\ 0 & -4 & 8 & 20 \\ 0 & 7 & -2 & -8 \end{bmatrix}$$

Normalize the second row and eliminate a_{12} and a_{32} ,

$$\begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & -2 & -5 \\ 0 & 0 & 12 & 27 \end{bmatrix}$$

Normalize the third row and eliminate a_{13} and a_{23} ,

$$\begin{bmatrix} 1 & 0 & 0 & -0.25 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 2.25 \end{bmatrix}$$

9.14 In a fashion similar to Example 9.11, vertical force balances can be written to give the following system of equations,

$$\begin{aligned} m_1g - T_{12} - c_1v &= m_1a \\ m_2g + T_{12} - c_2v - T_{23} &= m_2a \\ m_3g - c_3v + T_{23} - T_{34} &= m_3a \\ m_4g - c_4v + T_{34} - T_{45} &= m_4a \\ m_5g - c_5v + T_{45} &= m_5a \end{aligned}$$

After substituting the known values, the equations can be expressed in matrix form ($g = 9.8$),

$$\begin{bmatrix} 55 & 1 & 0 & 0 & 0 \\ 75 & -1 & 1 & 0 & 0 \\ 60 & 0 & -1 & 1 & 0 \\ 75 & 0 & 0 & -1 & 1 \\ 90 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} a \\ T_{12} \\ T_{23} \\ T_{34} \\ T_{45} \end{Bmatrix} = \begin{Bmatrix} 449 \\ 627 \\ 453 \\ 591 \\ 792 \end{Bmatrix}$$

The system can be solved for

$$\begin{aligned} a &= 8.202817 & T_{12} &= -2.15493 & T_{23} &= 9.633803 \\ T_{34} &= -29.5352 & T_{12} &= -53.7465 \end{aligned}$$

9.15 Using the format of Eq. 9.27,

$$[A] = \begin{bmatrix} 3 & 4 \\ 0 & 1 \end{bmatrix} \quad [B] = \begin{bmatrix} 2 & 0 \\ -1 & 0 \end{bmatrix}$$

$$\{U\} = \begin{Bmatrix} 2 \\ 3 \end{Bmatrix} \quad \{V\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$$

The set of equations to be solved are

$$\begin{aligned} 3x_1 + 4x_2 - 2y_1 &= 2 \\ x_2 + y_1 &= 3 \\ 2x_1 + 3y_1 + 4y_2 &= 1 \\ -x_1 + y_2 &= 0 \end{aligned}$$

These can be solved for $x_1 = -0.53333$, $x_2 = 1.6$, $y_1 = 1.4$, and $y_2 = -0.53333$. Therefore, the solution is $z_1 = -0.53333 + 1.4i$ and $z_2 = 1.6 - 0.53333i$.

9.16 Here is a VBA program to implement matrix multiplication and solve Prob. 9.3 for the case of $[A] \times [B]$.

```

Option Explicit

Sub Mult()
    Dim i As Integer, j As Integer
    Dim l As Integer, m As Integer, n As Integer
    Dim a(10, 10) As Double, b(10, 10) As Double
    Dim c(10, 10) As Double
    l = 2
    m = 2
    n = 3
    a(1, 1) = 1: a(1, 2) = 6
    a(2, 1) = 3: a(2, 2) = 10
    a(3, 1) = 7: a(3, 2) = 4
    b(1, 1) = 1: b(1, 2) = 3
    b(2, 1) = 0.5: b(2, 2) = 2
    Call Mmult(a, b, c, m, n, l)
    For i = 1 To n
        For j = 1 To l
            MsgBox c(i, j)
        Next j
    Next i
    End Sub

    Sub Mmult(a, b, c, m, n, l)
        Dim i As Integer, j As Integer, k As Integer
        Dim sum As Double
        For i = 1 To n
            For j = 1 To l
                sum = 0
                For k = 1 To m
                    sum = sum + a(i, k) * b(k, j)
                Next k
                c(i, j) = sum
            Next j
        Next i
        End Sub

```

9.17 Here is a VBA program to implement the matrix transpose and solve Prob. 9.3 for the case of $[A]^T$.

```

Option Explicit

Sub TransTest()
    Dim i As Integer, j As Integer
    Dim m As Integer, n As Integer
    Dim a(10, 10) As Double, aT(10, 10) As Double
    n = 3
    m = 2
    a(1, 1) = 1: a(1, 2) = 6
    a(2, 1) = 3: a(2, 2) = 10
    a(3, 1) = 7: a(3, 2) = 4
    Call Transpose(a, aT, n, m)
    For i = 1 To m
        For j = 1 To n
            MsgBox aT(i, j)
        Next j
    Next i
End Sub

Sub Transpose(a, b, n, m)
    Dim i As Integer, j As Integer
    For i = 1 To m
        For j = 1 To n
            b(i, j) = a(j, i)
        Next j
    Next i
End Sub

```

9.18 Here is a VBA program to implement the Gauss elimination algorithm and solve the test case in Prob. 9.16.

```

Option Explicit

Sub GaussElim()
    Dim n As Integer, er As Integer, i As Integer
    Dim a(10, 10) As Double, b(10) As Double, x(10) As Double
    n = 3
    a(1, 1) = 1: a(1, 2) = 2: a(1, 3) = -1
    a(2, 1) = 5: a(2, 2) = 2: a(2, 3) = 2
    a(3, 1) = -3: a(3, 2) = 5: a(3, 3) = -1
    b(1) = 2: b(2) = 9: b(3) = 1
    Call Gauss(a, b, n, x, er)
    If er = 0 Then
        For i = 1 To n
            MsgBox "x(" & i & ") = " & x(i)
        Next i
    Else
        MsgBox "ill-conditioned system"
    End If
End Sub

Sub Gauss(a, b, n, x, er)
    Dim i As Integer, j As Integer
    Dim s(10) As Double
    Const tol As Double = 0.000001
    er = 0
    For i = 1 To n

```

```

s(i) = Abs(a(i, 1))
For j = 2 To n
    If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
Next j
Next i
Call Eliminate(a, s, n, b, tol, er)
If er <> -1 Then
    Call Substitute(a, n, b, x)
End If
End Sub

Sub Pivot(a, b, s, n, k)
Dim p As Integer, ii As Integer, jj As Integer
Dim factor As Double, big As Double, dummy As Double
p = k
big = Abs(a(k, k) / s(k))
For ii = k + 1 To n
    dummy = Abs(a(ii, k) / s(ii))
    If dummy > big Then
        big = dummy
        p = ii
    End If
Next ii
If p <> k Then
    For jj = k To n
        dummy = a(p, jj)
        a(p, jj) = a(k, jj)
        a(k, jj) = dummy
    Next jj
    dummy = b(p)
    b(p) = b(k)
    b(k) = dummy
    dummy = s(p)
    s(p) = s(k)
    s(k) = dummy
End If
End Sub

Sub Substitute(a, n, b, x)
Dim i As Integer, j As Integer
Dim sum As Double
x(n) = b(n) / a(n, n)
For i = n - 1 To 1 Step -1
    sum = 0
    For j = i + 1 To n
        sum = sum + a(i, j) * x(j)
    Next j
    x(i) = (b(i) - sum) / a(i, i)
Next i
End Sub

Sub Eliminate(a, s, n, b, tol, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For k = 1 To n - 1
    Call Pivot(a, b, s, n, k)
    If Abs(a(k, k) / s(k)) < tol Then
        er = -1
        Exit For
    End If
    For i = k + 1 To n
        factor = a(i, k) / a(k, k)
    Next i
End Sub

```

```
For j = k + 1 To n
    a(i, j) = a(i, j) - factor * a(k, j)
Next j
b(i) = b(i) - factor * b(k)
Next i
Next k
If Abs(a(k, k) / s(k)) < tol Then er = -1
End Sub
```

Its application yields a solution of (1, 1, 1).

CHAPTER 10

10.1 Matrix multiplication is distributive

$$[L]\{[U]\{X\} - \{D\}\} = [A]\{X\} - \{B\}$$

$$[L][U]\{X\} - [L]\{D\} = [A]\{X\} - \{B\}$$

Therefore, equating like terms,

$$[L][U]\{X\} = [A]\{X\}$$

$$[L]\{D\} = \{B\}$$

$$[L][U] = [A]$$

10.2 (a) The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -3/10 = -0.3$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 1/10 = 0.1$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31} .

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & 0.8 & 5.1 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = 0.8/(-5.4) = -0.14815$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32} .

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & -0.14815 & 5.351852 \end{bmatrix}$$

Therefore, the LU decomposition is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}$$

These two matrices can be multiplied to yield the original system. For example, using MATLAB to perform the multiplication gives

```
>> L=[1 0 0; -.3 1 0; 0.1 -.14815 1];
>> U=[10 2 -1; 0 -5.4 1.7; 0 0 5.351852];
>> L*U

ans =
    10.0000    2.0000   -1.0000
   -3.0000   -6.0000    2.0000
    1.0000    1.0000    5.0000
```

(b) Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 27 \\ 61.5 \\ -21.5 \end{Bmatrix}$$

Solving yields $d_1 = 27$, $d_2 = -53.4$, and $d_3 = -32.1111$.

Back substitution:

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 27 \\ -53.4 \\ -32.1111 \end{Bmatrix}$$

$$x_3 = \frac{-32.1111}{5.351852} = -6$$

$$x_2 = \frac{-53.4 - 1.7(-6)}{-5.4} = 8$$

$$x_1 = \frac{27 - (-1)(-6) - 2(8)}{10} = 0.5$$

(c) Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.14815 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 18 \\ -6 \end{Bmatrix}$$

Solving yields $d_1 = 12$, $d_2 = 21.6$, and $d_3 = -4$.

Back substitution:

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 21.6 \\ -4 \end{Bmatrix}$$

$$x_3 = \frac{-4}{5.351852} = -0.7474$$

$$x_2 = \frac{21.6 - 1.7(-0.7474)}{-5.4} = -4.23529$$

$$x_1 = \frac{12 - (-1)(-0.7474) - 2(-4.23529)}{10} = 1.972318$$

10.3 (a) The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -2/8 = -0.25$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 2/8 = 0.25$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31} .

$$\begin{bmatrix} 8 & 4 & -1 \\ -0.25 & 6 & 0.75 \\ 0.25 & -2 & 6.25 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = -2/6 = -0.33333$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32} .

$$\begin{bmatrix} 8 & 4 & -1 \\ -0.25 & 6 & 0.75 \\ 0.25 & -0.33333 & 6.5 \end{bmatrix}$$

Therefore, the LU decomposition is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix}$$

Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 11 \\ 4 \\ 7 \end{Bmatrix}$$

Solving yields $d_1 = 11$, $d_2 = 6.75$, and $d_3 = 6.5$.

Back substitution:

$$\begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 11 \\ 6.75 \\ 6.5 \end{Bmatrix}$$

$$x_3 = \frac{6.5}{6.5} = 1$$

$$x_2 = \frac{6.75 - 0.75(1)}{6} = 1$$

$$x_1 = \frac{11 - (-1)(1) - 4(1)}{8} = 1$$

(b) The first column of the inverse can be computed by using $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.25 & -0.33333 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

This can be solved for $d_1 = 1$, $d_2 = 0.25$, and $d_3 = -0.16667$. Then, we can implement back substitution

$$\begin{bmatrix} 8 & 4 & -1 \\ 0 & 6 & 0.75 \\ 0 & 0 & 6.5 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0.25 \\ -0.16667 \end{Bmatrix}$$

to yield the first column of the inverse

$$\{X\} = \begin{Bmatrix} 0.099359 \\ 0.0448718 \\ -0.025641 \end{Bmatrix}$$

For the second column use $\{B\}^T = \{0 1 0\}$ which gives $\{D\}^T = \{0 1 0.33333\}$. Back substitution then gives $\{X\}^T = \{-0.073718 0.160256 0.051282\}$.

For the third column use $\{B\}^T = \{0 0 1\}$ which gives $\{D\}^T = \{0 0 1\}$. Back substitution then gives $\{X\}^T = \{0.028846 -0.01923 0.153846\}$.

Therefore, the matrix inverse is

$$[A]^{-1} = \begin{bmatrix} 0.099359 & -0.073718 & 0.028846 \\ 0.044872 & 0.160256 & -0.019231 \\ -0.025641 & 0.051282 & 0.153846 \end{bmatrix}$$

We can verify that this is correct by multiplying $[A][A]^{-1}$ to yield the identity matrix. For example, using MATLAB,

```
>> A=[8 4 -1;-2 5 1;2 -1 6];
>> AI=[0.099359 -0.073718 0.028846;
0.044872 0.160256 -0.019231;
-0.025641 0.051282 0.153846]
>> A*AI

ans =
1.0000    -0.0000    -0.0000
0.0000    1.0000    -0.0000
0           0         1.0000
```

10.4 As the system is set up, we must first pivot by switching the first and third rows of $[A]$. Note that we must make the same switch for the right-hand-side vector $\{B\}$

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ -3 & -1 & 7 \\ 2 & -6 & -1 \end{bmatrix} \quad \{B\} = \begin{Bmatrix} -20 \\ -34 \\ -38 \end{Bmatrix}$$

The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -3/-8 = 0.375$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 2/(-8) = -0.25$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31} .

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ 0.375 & -1.375 & 7.75 \\ -0.25 & -5.75 & -1.5 \end{bmatrix}$$

Next, we pivot by switching rows 2 and 3. Again, we must also make the same switch for the right-hand-side vector $\{B\}$

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ -0.25 & -5.75 & -1.5 \\ 0.375 & -1.375 & 7.75 \end{bmatrix} \quad \{B\} = \begin{bmatrix} -20 \\ -38 \\ -34 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = -1.375/(-5.75) = 0.23913$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32} .

$$[A] = \begin{bmatrix} -8 & 1 & -2 \\ -0.25 & -5.75 & -1.5 \\ 0.375 & 0.23913 & 8.108696 \end{bmatrix}$$

Therefore, the LU decomposition is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix}$$

Forward substitution: $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ 0.375 & 0.23913 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} -20 \\ -38 \\ -34 \end{bmatrix}$$

Solving yields $d_1 = -20$, $d_2 = -43$, and $d_3 = -16.2174$.

Back substitution:

$$\begin{bmatrix} -8 & 1 & -2 \\ 0 & -5.75 & -1.5 \\ 0 & 0 & 8.108696 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -20 \\ -43 \\ -16.2174 \end{bmatrix}$$

$$x_3 = \frac{-16.2174}{8.108696} = -2$$

$$x_2 = \frac{-43 + 1.5(-2)}{-5.75} = 8$$

$$x_1 = \frac{-20 + 2(-2) - 8}{-8} = 4$$

10.5 The flop counts for *LU* decomposition can be determined in a similar fashion as was done for Gauss elimination. The major difference is that the elimination is only implemented for the left-hand side coefficients. Thus, for every iteration of the inner loop, there are n multiplications/divisions and $n - 1$ addition/subtractions. The computations can be summarized as

Outer Loop k	Inner Loop i	Addition/Subtraction flops	Multiplication/Division flops
1	2, n	$(n - 1)(n - 1)$	$(n - 1)n$
2	3, n	$(n - 2)(n - 2)$	$(n - 2)(n - 1)$
.	.	.	.
.	.	.	.
k	$k + 1, n$	$(n - k)(n - k)$	$(n - k)(n + 1 - k)$
.	.	.	.
.	.	.	.
$n - 1$	n, n	(1)(1)	(1)(2)

Therefore, the total addition/subtraction flops for elimination can be computed as

$$\sum_{k=1}^{n-1} (n - k)(n - k) = \sum_{k=1}^{n-1} [n^2 - 2nk + k^2]$$

Applying some of the relationships from Eq. (8.14) yields

$$\sum_{k=1}^{n-1} [n^2 - 2nk + k^2] = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$$

A similar analysis for the multiplication/division flops yields

$$\sum_{k=1}^{n-1} (n - k)(n + 1 - k) = \frac{n^3}{3} - \frac{n}{3}$$

Summing these results gives

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}$$

For forward substitution, the numbers of multiplications and subtractions are the same and equal to

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

Back substitution is the same as for Gauss elimination: $n^2/2 - n/2$ subtractions and $n^2/2 + n/2$ multiplications/divisions. The entire number of flops can be summarized as

	Mult/Div	Add/Subtr	Total
Forward elimination	$\frac{n^3}{3} - \frac{n}{3}$	$\frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$	$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}$
Forward substitution	$\frac{n^2}{2} - \frac{n}{2}$	$\frac{n^2}{2} - \frac{n}{2}$	$n^2 - n$
Back substitution	$\frac{n^2}{2} + \frac{n}{2}$	$\frac{n^2}{2} - \frac{n}{2}$	n^2
Total	$\frac{n^3}{3} + n^2 - \frac{n}{3}$	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$	$\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{7n}{6}$

Thus, the total number of flops is identical to that obtained with standard Gauss elimination.

10.6 First, we compute the *LU* decomposition. The coefficient a_{21} is eliminated by multiplying row 1 by $f_{21} = -3/10 = -0.3$ and subtracting the result from row 2. a_{31} is eliminated by multiplying row 1 by $f_{31} = 1/10 = 0.1$ and subtracting the result from row 3. The factors f_{21} and f_{31} can be stored in a_{21} and a_{31} .

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & 0.8 & 5.1 \end{bmatrix}$$

a_{32} is eliminated by multiplying row 2 by $f_{32} = 0.8/(-5.4) = -0.148148$ and subtracting the result from row 3. The factor f_{32} can be stored in a_{32} .

$$\begin{bmatrix} 10 & 2 & -1 \\ -0.3 & -5.4 & 1.7 \\ 0.1 & -0.148148 & 5.351852 \end{bmatrix}$$

Therefore, the *LU* decomposition is

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.148148 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix}$$

The first column of the inverse can be computed by using $[L]\{D\} = \{B\}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.1 & -0.148148 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

This can be solved for $d_1 = 1$, $d_2 = 0.3$, and $d_3 = -0.055556$. Then, we can implement back substitution

$$\begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.4 & 1.7 \\ 0 & 0 & 5.351852 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0.3 \\ -0.055556 \end{Bmatrix}$$

to yield the first column of the inverse

$$\{X\} = \begin{Bmatrix} 0.110727 \\ -0.058824 \\ -0.0103806 \end{Bmatrix}$$

For the second column use $\{B\}^T = \{0 1 0\}$ which gives $\{D\}^T = \{0 1 0.148148\}$. Back substitution then gives $\{X\}^T = \{0.038062 -0.176471 0.027682\}$.

For the third column use $\{B\}^T = \{0 0 1\}$ which gives $\{D\}^T = \{0 0 1\}$. Back substitution then gives $\{X\}^T = \{0.00692 0.058824 0.186851\}$.

Therefore, the matrix inverse is

$$[A]^{-1} = \begin{bmatrix} 0.110727 & 0.038062 & 0.006920 \\ -0.058824 & -0.176471 & 0.058824 \\ -0.010381 & 0.027682 & 0.186851 \end{bmatrix}$$

We can verify that this is correct by multiplying $[A][A]^{-1}$ to yield the identity matrix. For example, using MATLAB,

```
>> A=[10 2 -1;-3 -6 2;1 1 5];
>> AI=[0.110727 0.038062 0.006920;
-0.058824 -0.176471 0.058824;
-0.010381 0.027682 0.186851];
>> A*AI

ans =
    1.0000    -0.0000    -0.0000
    0.0000     1.0000    -0.0000
   -0.0000     0.0000     1.0000
```

10.7 Equation 10.17 yields

$$l_{11} = 2 \quad l_{21} = -1 \quad l_{31} = 1$$

Equation 10.18 gives

$$u_{12} = \frac{a_{12}}{l_{11}} = -3 \quad u_{13} = \frac{a_{13}}{l_{11}} = 0.5$$

Equation 10.19 gives

$$l_{22} = a_{22} - l_{21}u_{12} = 4 \quad l_{32} = a_{32} - l_{31}u_{12} = 0$$

Equation 10.20 gives

$$u_{23} = \frac{a_{23} - l_{21}u_{13}}{l_{22}} = -0.125$$

Equation 10.21 gives

$$l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} = 1.5$$

Therefore, the *LU* decomposition is

$$[L] = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 4 & 0 \\ 1 & 0 & 1.5 \end{bmatrix} \quad [U] = \begin{bmatrix} 1 & -3 & 0.5 \\ 0 & 1 & -0.125 \\ 0 & 0 & 1 \end{bmatrix}$$

These two matrices can be multiplied to yield the original system. For example, using MATLAB to perform the multiplication gives

```
>> L=[2 0 0;-1 4 0;1 0 1.5];
>> U=[1 -3 0.5;0 1 -0.125;0 0 1];
>> L*U

ans =
    2      -6       1
   -1       7      -1
    1      -3       2
```

10.8 (a) Using MATLAB, the matrix inverse can be computed as

```
>> A=[15 -3 -1;-3 18 -6;-4 -1 12];
>> AI=inv(A)

AI =
    0.0725    0.0128    0.0124
    0.0207    0.0608    0.0321
    0.0259    0.0093    0.0902
```

(b)

```
>> B=[3800;1200;2350];
>> C=AI*B
```

```
C =
```

320.2073
227.2021
321.5026

$$(c) \Delta W_3 = \frac{\Delta c_1}{a_{13}^{-1}} = \frac{10}{0.012435} = 804.1667$$

$$(d) \Delta c_3 = a_{31}^{-1} \Delta W_1 + a_{32}^{-1} \Delta W_2 = 0.025907(-500) + 0.009326(-250) = -15.285$$

10.9 First we can scale the matrix to yield

$$[A] = \begin{bmatrix} -0.8 & -0.2 & 1 \\ 1 & -0.11111 & -0.33333 \\ 1 & -0.06667 & 0.4 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.967901} = 1.991959$$

In order to compute the column-sum and row-sum norms, we can determine the sums of the absolute values of each of the columns and rows:

			row sums ↓
-0.8	-0.2	1	2
1	-0.11111	-0.33333	1.44444
1	-0.06667	0.4	1.46667
2.8	0.37778	1.73333	←column sums

Therefore, $\|A\|_1 = 2.8$ and $\|A\|_\infty = 2$.

10.10 For the system from Prob. 10.3, we can scale the matrix to yield

$$[A] = \begin{bmatrix} 1 & 0.5 & -0.125 \\ -0.4 & 1 & 0.2 \\ 0.33333 & -0.16667 & 1 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.604514} = 1.898556$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

			row sums ↓

1	0.5	-0.125	1.625
-0.4	1	0.2	1.6
0.333333	-0.166667	1	1.5

Therefore, $\|A\|_\infty = 1.625$.

For the system from Prob. 10.4, we can scale the matrix to yield

$$[A] = \begin{bmatrix} -0.3333 & 1 & 0.166667 \\ -0.42857 & -0.14286 & 1 \\ 1 & -0.125 & 0.25 \end{bmatrix}$$

Frobenius norm:

$$\|A\|_e = \sqrt{3.421096} = 1.84962$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

			row sums ↓
-0.33333	1	0.166667	1.5
-0.42857	-0.14286	1	1.571429
1	-0.125	0.25	1.375

Therefore, $\|A\|_\infty = 1.571429$.

10.11 In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

				row sums ↓
0.125	0.25	0.5	1	1.875
0.015625	0.625	0.25	1	1.890625
0.00463	0.02777	0.166667	1	1.19907
0.001953	0.015625	0.125	1	1.142578

Therefore, $\|A\|_\infty = 1.890625$. The matrix inverse can then be computed. For example, using MATLAB,

```
>> A=[0.125 0.25 0.5 1;
0.015625 0.625 0.25 1;
0.00463 0.02777 0.166667 1;
0.001953 0.015625 0.125 1]
>> AI=inv(A)

AI =
10.2329 -2.2339 -85.3872 77.3883
```

-0.1008	1.7674	-4.3949	2.7283
-0.6280	-0.3716	30.7645	-29.7649
0.0601	0.0232	-3.6101	4.5268

The row-sum norm can then be computed by determining the sum of the absolute values of each of the rows. The result is $\|A^{-1}\|_{\infty} = 175.2423$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 1.890625(175.2423) = 331.3174$$

This corresponds to $\log_{10}(331.3174) = 2.52$ suspect digits.

10.12 (a) In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

					row sums ↓
1	4	9	16	25	55
4	9	16	25	36	90
9	16	25	36	49	135
16	25	36	49	64	190
25	36	49	64	81	255

Therefore, $\|A\|_{\infty} = 255$. The matrix inverse can then be computed. For example, using MATLAB,

```
>> A=[1 4 9 16 25;
4 9 16 25 36;
9 16 25 36 49;
16 25 36 49 64;
25 36 49 64 81];
>> AI=inv(A)
Warning: Matrix is close to singular or badly scaled.
          Results may be inaccurate. RCOND = 9.944077e-019.
AI =
1.0e+015 *
-0.2800    0.6573   -0.2919   -0.2681    0.1827
 0.5211   -1.3275    0.8562    0.1859   -0.2357
 0.1168    0.0389   -0.8173    1.0508   -0.3892
-0.6767    1.2756    0.2335   -1.5870    0.7546
 0.3189   -0.6443    0.0195    0.6184   -0.3124
```

Notice that MATLAB alerts us that the matrix is ill-conditioned. This is also strongly suggested by the fact that the elements are so large.

The row-sum norm can then be computed by determining the sum of the absolute values of each of the rows. The result is $\|A^{-1}\|_{\infty} = 4.5274 \times 10^{15}$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 255(4.5274 \times 10^{15}) = 1.1545 \times 10^{18}$$

This corresponds to $\log_{10}(1.1545 \times 10^{18}) = 18.06$ suspect digits. Thus, the suspect digits are more than the number of significant digits for the double precision representation used in MATLAB (15-16 digits). Consequently, we can conclude that this matrix is highly ill-conditioned.

It should be noted that if you used Excel for this computation you would have arrived at a slightly different result of $\text{Cond}[A] = 1.263 \times 10^{18}$.

(b) First, the matrix is scaled. For example, using MATLAB,

```
>> A=[1/25 4/25 9/25 16/25 25/25;
    4/36 9/36 16/36 25/26 36/36;
    9/49 16/49 25/49 36/49 49/49;
    16/64 25/64 36/64 49/64 64/64;
    25/81 36/81 49/81 64/81 81/81]
```

```
A =
0.0400 0.1600 0.3600 0.6400 1.0000
0.1111 0.2500 0.4444 0.9615 1.0000
0.1837 0.3265 0.5102 0.7347 1.0000
0.2500 0.3906 0.5625 0.7656 1.0000
0.3086 0.4444 0.6049 0.7901 1.0000
```

The row-sum norm can be computed as 3.1481. Next, we can invert the matrix,

```
>> AI=inv(A)
Warning: Matrix is close to singular or badly scaled.
          Results may be inaccurate. RCOND = 2.230462e-018.
```

```
AI =
1.0e+016 *
-0.0730 0 0.8581 -1.4945 0.7093
0.1946 -0.0000 -2.2884 3.9852 -1.8914
-0.1459 0 1.7163 -2.9889 1.4186
-0.0000 0.0000 -0.0000 -0.0000 0.0000
0.0243 -0.0000 -0.2860 0.4982 -0.2364
```

The row-sum norm of the inverse can be computed as 8.3596×10^{16} . The condition number can then be computed as

$$\text{Cond}[A] = 3.1481(8.3596 \times 10^{16}) = 2.6317 \times 10^{17}$$

This corresponds to $\log_{10}(2.6317 \times 10^{17}) = 17.42$ suspect digits. Thus, as with (a), the suspect digits are more than the number of significant digits for the double precision representation used in MATLAB (15-16 digits). Consequently, we again can conclude that this matrix is highly ill-conditioned.

It should be noted that if you used Excel for this computation you would have arrived at a slightly different result of $\text{Cond}[A] = 1.3742 \times 10^{17}$.

10.13 In order to compute the row-sum norm of the normalized 5×5 Hilbert matrix, we can determine the sum of the absolute values of each of the rows:

					row sums ↓
1	0.500000	0.333333	0.250000	0.200000	2.283333
1	0.666667	0.500000	0.400000	0.333333	2.9
1	0.750000	0.600000	0.500000	0.428571	3.278571
1	0.800000	0.666667	0.571429	0.500000	3.538095
1	0.833333	0.714286	0.625000	0.555556	3.728175

Therefore, $\|A\|_{\infty} = 3.728175$. The matrix inverse can then be computed and the row sums calculated as

					row sums ↓
25	-150	350	-350	126	1001
-300	2400	-6300	6720	-2520	18240
1050	-9450	26460	-29400	11340	77700
-1400	13440	-39200	44800	-17640	116480
630	-6300	18900	-22050	8820	56700

The result is $\|A^{-1}\|_{\infty} = 116,480$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 3.728175(116,480) = 434,258$$

This corresponds to $\log_{10}(434,258) = 5.64$ suspect digits.

10.14 The matrix to be evaluated can be computed by substituting the x values into the Vandermonde matrix to give

$$[A] = \begin{bmatrix} 16 & 4 & 1 \\ 4 & 2 & 1 \\ 49 & 7 & 1 \end{bmatrix}$$

We can then scale the matrix by dividing each row by its maximum element,

$$[A] = \begin{bmatrix} 1 & 0.25 & 0.0625 \\ 1 & 0.5 & 0.25 \\ 1 & 0.142857 & 0.020408 \end{bmatrix}$$

In order to compute the row-sum norm, we can determine the sum of the absolute values of each of the rows:

			row sums ↓
1	0.25	0.0625	1.3125

1	0.5	0.25	1.75
1	0.142857	0.020408	1.163265

Therefore, $\|A\|_{\infty} = 1.75$. The matrix inverse can then be computed and the row sums calculated as

			row sums ↓
-2.666667	0.4	3.266667	6.333333
24	-4.4	-19.6	48
-37.3333	11.2	26.13333	74.66667

The result is $\|A^{-1}\|_{\infty} = 74.66667$. Therefore, the condition number can be computed as

$$\text{Cond}[A] = 1.75(74.66667) = 130.6667$$

This result can be checked with MATLAB,

```
>> A=[16/16 4/16 1/16;
4/4 2/4 1/4;
49/49 7/49 1/49]
>> cond(A,inf)

ans =
130.6667
```

(b) MATLAB can be used to compute the spectral and Frobenius condition numbers,

```
>> A=[16/16 4/16 1/16;
4/4 2/4 1/4;
49/49 7/49 1/49]
>> cond(A,2)

ans =
102.7443

>> cond(A,'fro')

ans =
104.2971
```

[Note: If you did not scale the original matrix, the results are: Cond[A] $_{\infty}$ = 323, Cond[A] $_2$ = 216.1294, and Cond[A] $_e$ = 217.4843]

10.15 Here is a VBA program that implements LU decomposition. It is set up to solve Example 10.1.

```
Option Explicit

Sub LUDTest()
Dim n As Integer, er As Integer, i As Integer, j As Integer
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

Dim a(3, 3) As Double, b(3) As Double, x(3) As Double
Dim tol As Double
n = 3
a(1, 1) = 3: a(1, 2) = -0.1: a(1, 3) = -0.2
a(2, 1) = 0.1: a(2, 2) = 7: a(2, 3) = -0.3
a(3, 1) = 0.3: a(3, 2) = -0.2: a(3, 3) = 10
b(1) = 7.85: b(2) = -19.3: b(3) = 71.4
tol = 0.000001
Call LUD(a, b, n, x, tol, er)
'output results to worksheet
Sheets("Sheet1").Select
Range("a3").Select
For i = 1 To n
    ActiveCell.Value = x(i)
    ActiveCell.Offset(1, 0).Select
Next i
Range("a3").Select

End Sub

Sub LUD(a, b, n, x, tol, er)
Dim i As Integer, j As Integer
Dim o(3) As Double, s(3) As Double
Call Decompose(a, n, tol, o, s, er)
If er = 0 Then
    Call Substitute(a, o, n, b, x)
Else
    MsgBox "ill-conditioned system"
    End
End If
End Sub

Sub Decompose(a, n, tol, o, s, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For i = 1 To n
    o(i) = i
    s(i) = Abs(a(i, 1))
    For j = 2 To n
        If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
    Next j
Next i
For k = 1 To n - 1
    Call Pivot(a, o, s, n, k)
    If Abs(a(o(k), k) / s(o(k))) < tol Then
        er = -1
        Exit For
    End If
    For i = k + 1 To n
        factor = a(o(i), k) / a(o(k), k)
        a(o(i), k) = factor
        For j = k + 1 To n
            a(o(i), j) = a(o(i), j) - factor * a(o(k), j)
        Next j
    Next i
Next k
If (Abs(a(o(k), k) / s(o(k))) < tol) Then er = -1
End Sub

Sub Pivot(a, o, s, n, k)
Dim ii As Integer, p As Integer
Dim big As Double, dummy As Double

```

```

p = k
big = Abs(a(o(k), k) / s(o(k)))
For ii = k + 1 To n
    dummy = Abs(a(o(ii), k) / s(o(ii)))
    If dummy > big Then
        big = dummy
        p = ii
    End If
Next ii
dummy = o(p)
o(p) = o(k)
o(k) = dummy
End Sub

Sub Substitute(a, o, n, b, x)
Dim k As Integer, i As Integer, j As Integer
Dim sum As Double, factor As Double
For k = 1 To n - 1
    For i = k + 1 To n
        factor = a(o(i), k)
        b(o(i)) = b(o(i)) - factor * b(o(k))
    Next i
Next k
x(n) = b(o(n)) / a(o(n), n)
For i = n - 1 To 1 Step -1
    sum = 0
    For j = i + 1 To n
        sum = sum + a(o(i), j) * x(j)
    Next j
    x(i) = (b(o(i)) - sum) / a(o(i), i)
Next i
End Sub

```

10.16 Here is a VBA program that uses *LU* decomposition to determine the matrix inverse. It is set up to solve Example 10.3.

```

Option Explicit

Sub LUInverseTest()
Dim n As Integer, er As Integer, i As Integer, j As Integer
Dim a(3, 3) As Double, b(3) As Double, x(3) As Double
Dim tol As Double, ai(3, 3) As Double
n = 3
a(1, 1) = 3: a(1, 2) = -0.1: a(1, 3) = -0.2
a(2, 1) = 0.1: a(2, 2) = 7: a(2, 3) = -0.3
a(3, 1) = 0.3: a(3, 2) = -0.2: a(3, 3) = 10
tol = 0.000001
Call LUDminv(a, b, n, x, tol, er, ai)
If er = 0 Then
    Range("a1").Select
    For i = 1 To n
        For j = 1 To n
            ActiveCell.Value = ai(i, j)
            ActiveCell.Offset(0, 1).Select
        Next j
        ActiveCell.Offset(1, -n).Select
    Next i
    Range("a1").Select
Else
    MsgBox "ill-conditioned system"
End If

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

End Sub

Sub LUDminv(a, b, n, x, tol, er, ai)
Dim i As Integer, j As Integer
Dim o(3) As Double, s(3) As Double
Call Decompose(a, n, tol, o, s, er)
If er = 0 Then
    For i = 1 To n
        For j = 1 To n
            If i = j Then
                b(j) = 1
            Else
                b(j) = 0
            End If
        Next j
        Call Substitute(a, o, n, b, x)
        For j = 1 To n
            ai(j, i) = x(j)
        Next j
    Next i
End If
End Sub

Sub Decompose(a, n, tol, o, s, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For i = 1 To n
    o(i) = i
    s(i) = Abs(a(i, 1))
    For j = 2 To n
        If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
    Next j
Next i
For k = 1 To n - 1
    Call Pivot(a, o, s, n, k)
    If Abs(a(o(k), k) / s(o(k))) < tol Then
        er = -1
        Exit For
    End If
    For i = k + 1 To n
        factor = a(o(i), k) / a(o(k), k)
        a(o(i), k) = factor
        For j = k + 1 To n
            a(o(i), j) = a(o(i), j) - factor * a(o(k), j)
        Next j
    Next i
Next k
If (Abs(a(o(k), k) / s(o(k))) < tol) Then er = -1
End Sub

Sub Pivot(a, o, s, n, k)
Dim ii As Integer, p As Integer
Dim big As Double, dummy As Double
p = k
big = Abs(a(o(k), k) / s(o(k)))
For ii = k + 1 To n
    dummy = Abs(a(o(ii), k) / s(o(ii)))
    If dummy > big Then
        big = dummy
        p = ii
    End If
Next ii

```

```

dummy = o(p)
o(p) = o(k)
o(k) = dummy
End Sub

Sub Substitute(a, o, n, b, x)
Dim k As Integer, i As Integer, j As Integer
Dim sum As Double, factor As Double
For k = 1 To n - 1
    For i = k + 1 To n
        factor = a(o(i), k)
        b(o(i)) = b(o(i)) - factor * b(o(k))
    Next i
Next k
x(n) = b(o(n)) / a(o(n), n)
For i = n - 1 To 1 Step -1
    sum = 0
    For j = i + 1 To n
        sum = sum + a(o(i), j) * x(j)
    Next j
    x(i) = (b(o(i)) - sum) / a(o(i), i)
Next i
End Sub

```

10.17 The problem can be set up as

$$2\Delta x_1 + 5\Delta x_2 + \Delta x_3 = -5 - (-3) = -2$$

$$6\Delta x_1 + 2\Delta x_2 + \Delta x_3 = 12 - 14 = -2$$

$$\Delta x_1 + 2\Delta x_2 + \Delta x_3 = 3 - 4 = -1$$

which can be solved for $\Delta x_1 = -0.2$, $\Delta x_2 = -0.26667$, and $\Delta x_3 = -0.26667$. These can be used to yield the corrected results

$$\begin{aligned} x_1 &= 2 - 0.2 = 1.8 \\ x_2 &= -3 - 0.26667 = -3.26667 \\ x_3 &= 8 - 0.26667 = 7.73333 \end{aligned}$$

These results are exact.

10.18

$$\begin{aligned} \vec{A} \cdot \vec{B} &= 0 \Rightarrow -4a + 2b = 3 \quad (1) \\ \vec{A} \cdot \vec{C} &= 0 \Rightarrow 2a - 3c = -6 \quad (2) \\ \vec{B} \cdot \vec{C} &= 2 \Rightarrow 3b + c = 10 \quad (3) \end{aligned}$$

Solve the three equations using Matlab:

```

>> A=[-4 2 0; 2 0 -3; 0 3 1]
b=[3; -6; 10]
x=inv(A)*b

x = 0.525
      2.550

```

2.350

Therefore, $a = 0.525$, $b = 2.550$, and $c = 2.350$.

10.19

$$(\vec{A} \times \vec{B}) = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ -2 & 1 & -4 \end{vmatrix} = (-4b - c)\vec{i} - (-4a + 2c)\vec{j} + (a + 2b)\vec{k}$$

$$(\vec{A} \times \vec{C}) = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ 1 & 3 & 2 \end{vmatrix} = (2b - 3c)\vec{i} - (2a - c)\vec{j} + (3a - b)\vec{k}$$

$$(\vec{A} \times \vec{B}) + (\vec{A} \times \vec{C}) = (-2b - 4c)\vec{i} - (-2a + c)\vec{j} + (4a + b)\vec{k}$$

Therefore,

$$(-2b - 4c)\vec{i} + (2a - c)\vec{j} + (4a + b)\vec{k} = (5a + 6)\vec{i} + (3b - 2)\vec{j} + (-4c + 1)\vec{k}$$

We get the following set of equations \Rightarrow

$$-2b - 4c = 5a + 6 \Rightarrow -5a - 2b - 4c = 6 \quad (1)$$

$$2a - c = 3b - 2 \Rightarrow 2a - 3b - c = -2 \quad (2)$$

$$4a + b = -4c + 1 \Rightarrow 4a + b + 4c = 1 \quad (3)$$

In Matlab:

```
>> A=[-5 -2 -4 ; 2 -3 -1 ; 4 1 4];
>> B=[6 ; -2 ; 1];
>> x = A\B

x =
-3.6522
-3.3478
4.7391
```

$$a = -3.6522, b = -3.3478, c = 4.7391$$

10.20

(I) $f(0) = 1 \Rightarrow a(0) + b = 1 \Rightarrow b = 1$
 $f(2) = 1 \Rightarrow c(2) + d = 1 \Rightarrow 2c + d = 1$

(II) If f is continuous, then at $x = 1$

$$ax + b = cx + d \Rightarrow a(1) + b = c(1) + d \Rightarrow a + b - c - d = 0$$

(III) $a + b = 4$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 4 \end{bmatrix}$$

These can be solved using MATLAB

```
>> A=[0 1 0 0;0 0 2 1;1 1 -1 -1;1 1 0 0];
>> B=[1;1;0;4];
>> A\B

ans =
    3
    1
   -3
    7
```

Thus, $a = 3$, $b = 1$, $c = -3$, and $d = 7$.

10.21 MATLAB provides a handy way to solve this problem.

(a)

```
>> a=hilb(3)

a =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000

>> x=[1 1 1]'

x =
    1
    1
    1

>> b=a*x

b =
    1.8333
    1.0833
    0.7833

>> format long e
>> x=a\b

x =
    9.99999999999992e-001
    1.00000000000006e+000
    9.99999999999939e-001

(b)
>> a=hilb(7);
>> x=ones(7,1);
>> b=a*x;
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```
>> x=a\b  
  
x =  
9.999999999927329e-001  
1.00000000289139e+000  
9.99999972198158e-001  
1.000000010794369e+000  
9.99999802287199e-001  
1.000000017073336e+000  
9.99999943967310e-001  
  
(c)  
>> a=hilb(10);  
>> x=ones(10,1);  
>> b=a*x;  
>> x=a\b  
  
x =  
9.99999993518614e-001  
1.000000053255573e+000  
9.99989124259656e-001  
1.000009539399602e+000  
9.99958816980565e-001  
1.000118062679701e+000  
9.998108238105067e-001  
1.000179021813331e+000  
9.999077593295230e-001  
1.000019946488826e+000
```

CHAPTER 11

11.1 First, the decomposition is implemented as

$$\begin{aligned} e_2 &= -0.4/0.8 = -0.5 \\ f_2 &= 0.8 - (-0.5)(-0.4) = 0.6 \\ e_3 &= -0.4/0.6 = -0.66667 \\ f_3 &= 0.8 - (-0.66667)(-0.4) = 0.53333 \end{aligned}$$

Transformed system is

$$\begin{bmatrix} 0.8 & -0.4 & 0 \\ -0.5 & 0.6 & -0.4 \\ 0 & -0.66667 & 0.53333 \end{bmatrix}$$

which is decomposed as

$$[L] = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0 & -0.66667 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} 0.8 & -0.4 & 0 \\ 0 & 0.6 & -0.4 \\ 0 & 0 & 0.53333 \end{bmatrix}$$

The right hand side becomes

$$\begin{aligned} r_1 &= 41 \\ r_2 &= 25 - (-0.5)(41) = 45.5 \\ r_3 &= 105 - (-0.66667)45.5 = 135.3333 \end{aligned}$$

which can be used in conjunction with the $[U]$ matrix to perform back substitution and obtain the solution

$$\begin{aligned} x_3 &= 135.3333/0.53333 = 253.75 \\ x_2 &= (45.5 - (-0.4)253.75)/0.6 = 245 \\ x_1 &= (41 - (-0.4)245)/0.8 = 173.75 \end{aligned}$$

11.2 As in Example 11.1, the LU decomposition is

$$[L] = \begin{bmatrix} 1 & & & \\ -0.49 & 1 & & \\ & -0.645 & 1 & \\ & & -0.717 & 1 \end{bmatrix} \quad [U] = \begin{bmatrix} 2.04 & -1 & & \\ & 1.550 & -1 & \\ & & 1.395 & -1 \\ & & & 1.323 \end{bmatrix}$$

To compute the first column of the inverse

$$[L]\{D\} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Solving this gives

$$\{D\} = \begin{Bmatrix} 1 \\ 0.490196 \\ 0.316296 \\ 0.226775 \end{Bmatrix}$$

Back substitution, $[U]\{X\} = \{D\}$, can then be implemented to give to first column of the inverse

$$\{X\} = \begin{Bmatrix} 0.755841 \\ 0.541916 \\ 0.349667 \\ 0.171406 \end{Bmatrix}$$

For the second column

$$[L]\{D\} = \begin{Bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{Bmatrix}$$

which leads to

$$\{X\} = \begin{Bmatrix} 0.541916 \\ 1.105509 \\ 0.713322 \\ 0.349667 \end{Bmatrix}$$

For the third column

$$[L]\{D\} = \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{Bmatrix}$$

which leads to

$$\{X\} = \begin{Bmatrix} 0.349667 \\ 0.713322 \\ 1.105509 \\ 0.541916 \end{Bmatrix}$$

For the fourth column

$$[L]\{D\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix}$$

which leads to

$$\{X\} = \begin{pmatrix} 0.171406 \\ 0.349667 \\ 0.541916 \\ 0.755841 \end{pmatrix}$$

Therefore, the matrix inverse is

$$[A]^{-1} = \begin{bmatrix} 0.755841 & 0.541916 & 0.349667 & 0.171406 \\ 0.541916 & 1.105509 & 0.713322 & 0.349667 \\ 0.349667 & 0.713322 & 1.105509 & 0.541916 \\ 0.171406 & 0.349667 & 0.541916 & 0.755841 \end{bmatrix}$$

11.3 First, the decomposition is implemented as

$$e_2 = -0.020875/2.01475 = -0.01036$$

$$f_2 = 2.014534$$

$$e_3 = -0.01036$$

$$f_3 = 2.014534$$

$$e_4 = -0.01036$$

$$f_4 = 2.014534$$

Transformed system is

$$\begin{bmatrix} 2.01475 & -0.02875 & & \\ -0.01036 & 2.014534 & -0.02875 & \\ & -0.01036 & 2.014534 & -0.02875 \\ & & -0.01036 & 2.014534 \end{bmatrix}$$

which is decomposed as

$$[L] = \begin{bmatrix} 1 & & & \\ -0.01036 & 1 & & \\ & -0.01036 & 1 & \\ & & -0.01036 & 1 \end{bmatrix}$$

$$[U] = \begin{bmatrix} 2.01475 & -0.02875 & & \\ & 2.014534 & -0.02875 & \\ & & 2.014534 & -0.02875 \\ & & & 2.014534 \end{bmatrix}$$

Forward substitution yields

$$r_1 = 4.175$$

$$r_2 = 0.043258$$

$$r_3 = 0.000448$$

$$r_4 = 2.087505$$

Back substitution

$$x_4 = 1.036222$$

$$x_3 = 0.01096$$

$$x_2 = 0.021586$$

$$x_1 = 2.072441$$

11.4 We can use MATLAB to verify the results of Example 11.2,

```
>> L=[2.4495 0 0; 6.1237 4.1833 0; 22.454 20.916 6.1106]
```

```
L =
2.4495 0 0
6.1237 4.1833 0
22.4540 20.9160 6.1106
```

```
>> L*L'
```

```
ans =
6.0001 15.0000 55.0011
15.0000 54.9997 224.9995
55.0011 224.9995 979.0006
```

11.5

$$l_{11} = \sqrt{8} = 2.828427$$

$$l_{21} = \frac{20}{2.828427} = 7.071068$$

$$l_{22} = \sqrt{80 - 7.071068^2} = 5.477226$$

$$l_{31} = \frac{15}{2.828427} = 5.303301$$

$$l_{32} = \frac{50 - 7.071068(5.303301)}{5.477226} = 2.282177$$

$$l_{33} = \sqrt{60 - 5.303301^2 - 2.282177^2} = 5.163978$$

Thus, the Cholesky decomposition is

$$[L] = \begin{bmatrix} 2.828427 \\ 7.071068 & 5.477226 \\ 5.303301 & 2.282177 & 5.163978 \end{bmatrix}$$

11.6

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$l_{11} = \sqrt{6} = 2.44949$$

$$l_{21} = \frac{15}{2.44949} = 6.123724$$

$$l_{22} = \sqrt{55 - 6.123724^2} = 4.1833$$

$$l_{31} = \frac{55}{2.44949} = 22.45366$$

$$l_{32} = \frac{225 - 6.123724(22.45366)}{4.1833} = 20.9165$$

$$l_{33} = \sqrt{979 - 22.45366^2 - 20.9165^2} = 6.110101$$

Thus, the Cholesky decomposition is

$$[L] = \begin{bmatrix} 2.44949 & & \\ 6.123724 & 4.1833 & \\ 22.45366 & 20.9165 & 6.110101 \end{bmatrix}$$

The solution can then be generated by first using forward substitution to modify the right-hand-side vector,

$$[L]\{D\} = \{B\}$$

which can be solved for

$$\{D\} = \begin{Bmatrix} 62.29869 \\ 48.78923 \\ 11.36915 \end{Bmatrix}$$

Then, we can use back substitution to determine the final solution,

$$[L]^T \{X\} = \{D\}$$

which can be solved for

$$\{D\} = \begin{Bmatrix} 2.478571 \\ 2.359286 \\ 1.860714 \end{Bmatrix}$$

11.7 (a) The first iteration can be implemented as

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

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

$$x_3 = \frac{105 + 0.4x_2}{0.8} = \frac{105 + 0.4(56.875)}{0.8} = 159.6875$$

Second iteration:

$$x_1 = \frac{41 + 0.4(56.875)}{0.8} = 79.6875$$

$$x_2 = \frac{25 + 0.4(79.6875) + 0.4(159.6875)}{0.8} = 150.9375$$

$$x_3 = \frac{105 + 0.4(150.9375)}{0.8} = 206.7188$$

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%	100.00%
	x_2	56.875	100.00%	
	x_3	159.6875	100.00%	
2	x_1	79.6875	35.69%	62.32%
	x_2	150.9375	62.32%	
	x_3	206.7188	22.75%	
3	x_1	126.7188	37.11%	37.11%
	x_2	197.9688	23.76%	
	x_3	230.2344	10.21%	
4	x_1	150.2344	15.65%	10.62%
	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 $x_3 = 253.622$.

11.8 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:

$$c_1 = \frac{3800 + 3c_2 + c_3}{15} = \frac{3800 + 3(108.8889) + 289.3519}{15} = 294.4012$$

$$c_2 = \frac{1200 + 3c_1 + 6c_3}{18} = \frac{1200 + 3(294.4012) + 6(289.3519)}{18} = 212.1842$$

$$c_3 = \frac{2350 + 4c_1 + c_2}{12} = \frac{2350 + 4(294.4012) + 212.1842}{12} = 311.6491$$

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

Thus, after 4 iterations, the maximum error is 1.43% and we arrive at the result: $c_1 = 319.3254$, $c_2 = 226.5402$ and $c_3 = 321.1535$.

11.9 The first iteration can be implemented as

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

$$c_2 = \frac{1200 + 3c_1 + 6c_3}{18} = \frac{1200 + 3(0) + 6(0)}{18} = 66.6667$$

$$c_3 = \frac{2350 + 4c_1 + c_2}{12} = \frac{2350 + 4(0) + 66.6667}{12} = 195.8333$$

Second iteration:

$$c_1 = \frac{3800 + 3c_2 + c_3}{15} = \frac{3800 + 3(66.6667) + 195.8333}{15} = 279.7222$$

$$c_2 = \frac{1200 + 3c_1 + 6c_3}{18} = \frac{1200 + 3(253.3333) + 6(195.8333)}{18} = 174.1667$$

$$c_3 = \frac{2350 + 4c_1 + c_2}{12} = \frac{2350 + 4(253.3333) + 66.6667}{12} = 285.8333$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left| \frac{279.7222 - 253.3333}{279.7222} \right| \times 100\% = 9.43\%$$

$$\varepsilon_{a,2} = \left| \frac{174.1667 - 66.6667}{174.1667} \right| \times 100\% = 61.72\%$$

$$\varepsilon_{a,3} = \left| \frac{285.8333 - 195.8333}{285.8333} \right| \times 100\% = 31.49\%$$

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	c_1	253.3333	100.00%	100.00%
	c_2	66.66667	100.00%	
	c_3	195.8333	100.00%	
2	c_1	279.7222	9.43%	61.72%
	c_2	174.1667	61.72%	
	c_3	285.8333	31.49%	
3	c_1	307.2222	8.95%	16.49%
	c_2	208.5648	16.49%	
	c_3	303.588	5.85%	
4	c_1	315.2855	2.56%	4.79%
	c_2	219.0664	4.79%	
	c_3	315.6211	3.81%	

Thus, after 4 iterations, the maximum error is 4.79% and we arrive at the result: $c_1 = 315.5402$, $c_2 = 219.0664$ and $c_3 = 315.6211$.

11.10 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%	100%
	x_2	8.9	100.00%	
	x_3	-6.62	100.00%	
2	x_1	0.258	946.51%	946%
	x_2	7.914333	12.45%	
	x_3	-5.934467	11.55%	

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 $x_3 = -6.00007$.

11.11 The equations should first be rearranged so that they are diagonally dominant,

$$6x_1 - x_2 - x_3 = 3$$

$$6x_1 + 9x_2 + x_3 = 40$$

$$-3x_1 + x_2 + 12x_3 = 50$$

Each can be solved for the unknown on the diagonal as

$$x_1 = \frac{3 + x_2 + x_3}{6}$$

$$x_2 = \frac{40 - 6x_1 - x_3}{9}$$

$$x_3 = \frac{50 + 3x_1 - x_2}{12}$$

(a) The first iteration can be implemented as

$$x_1 = \frac{3 + 0 + 0}{6} = 0.5$$

$$x_2 = \frac{40 - 6(0.5) - 0}{9} = 4.11111$$

$$x_3 = \frac{50 + 3(0.5) - 4.11111}{12} = 3.949074$$

Second iteration:

$$x_1 = \frac{3 + 4.11111 + 3.949074}{6} = 1.843364$$

$$x_2 = \frac{40 - 6(1.843364) - 3.949074}{9} = 2.776749$$

$$x_3 = \frac{50 + 3(1.843364) - 2.776749}{12} = 4.396112$$

The error estimates can be computed as

$$\varepsilon_{a,1} = \left| \frac{1.843364 - 0.5}{1.843364} \right| \times 100\% = 72.88\%$$

$$\varepsilon_{a,2} = \left| \frac{2.776749 - 4.11111}{2.776749} \right| \times 100\% = 48.05\%$$

$$\varepsilon_{a,3} = \left| \frac{4.396112 - 3.949074}{4.396112} \right| \times 100\% = 10.17\%$$

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	0.5	100.00%	100.00%
	x_2	4.111111	100.00%	
	x_3	3.949074	100.00%	
2	x_1	1.843364	72.88%	72.88%
	x_2	2.776749	48.05%	
	x_3	4.396112	10.17%	
3	x_1	1.695477	8.72%	8.72%
	x_2	2.82567	1.73%	
	x_3	4.355063	0.94%	
4	x_1	1.696789	0.08%	0.13%
	x_2	2.829356	0.13%	
	x_3	4.355084	0.00%	

Thus, after 4 iterations, the maximum error is 0.13% and we arrive at the result: $x_1 = 1.696789$, $x_2 = 2.829356$ and $x_3 = 4.355084$.

(b) First iteration: To start, assume $x_1 = x_2 = x_3 = 0$

$$x_1^{new} = \frac{3 + 0 + 0}{6} = 0.5$$

Apply relaxation

$$x_1 = 0.95(0.5) + (1 - 0.95)0 = 0.475$$

$$x_2^{new} = \frac{40 - 6(0.475) - 0}{9} = 4.12778$$

$$x_2 = 0.95(4.12778) + (1 - 0.95)0 = 3.92139$$

$$x_3^{new} = \frac{50 + 3(0.475) - 3.92139}{12} = 3.95863$$

$$x_3 = 0.95(3.95863) + (1 - 0.95)0 = 3.76070$$

Note that error estimates are not made on the first iteration, because all errors will be 100%.

Second iteration:

$$x_1^{new} = \frac{3 + 3.92139 + 3.76070}{6} = 1.78035$$

$$x_1 = 0.95(1.78035) + (1 - 0.95)(0.475) = 1.71508$$

At this point, an error estimate can be made

$$\varepsilon_{a,1} = \left| \frac{1.71508 - 0.475}{1.71508} \right| 100\% = 72.3\%$$

Because this error exceeds the stopping criterion, it will not be necessary to compute error estimates for the remainder of this iteration.

$$x_2^{new} = \frac{40 - 6(1.71508) - 3.76070}{9} = 2.88320$$

$$x_2 = 0.95(2.88320) + (1 - 0.95)3.92139 = 2.93511$$

$$x_3^{new} = \frac{50 + 3(1.71508) - 2.93511}{12} = 4.35084$$

$$x_3 = 0.95(4.35084) + (1 - 0.95)3.76070 = 4.32134$$

The computations can be continued for one more iteration. The entire calculation is summarized in the following table.

iteration	x_1	x_{1r}	ε_{a1}	x_2	x_{2r}	ε_{a2}	x_3	x_{3r}	ε_{a3}
1	0.50000	0.47500	100.0%	4.12778	3.92139	100.0%	3.95863	3.76070	100.0%
2	1.78035	1.71508	72.3%	2.88320	2.93511	33.6%	4.35084	4.32134	13.0%
3	1.70941	1.70969	0.3%	2.82450	2.83003	3.7%	4.35825	4.35641	0.8%

After 3 iterations, the approximate errors fall below the stopping criterion with the final result: $x_1 = 1.70969$, $x_2 = 2.82450$ and $x_3 = 4.35641$. Note that the exact solution is $x_1 = 1.69737$, $x_2 = 2.82895$ and $x_3 = 4.35526$

11.12 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

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

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

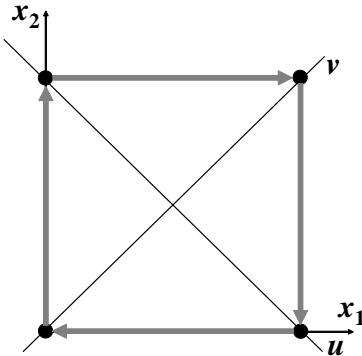
$$\varepsilon_{a,3} = \left| \frac{-1.5229518 - (-2.7771429)}{-1.5229518} \right| \times 100\% = 82.35\%$$

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 $x_3 = -1.979007$.

11.13 As shown below, for slopes of 1 and -1 the Gauss-Seidel technique will neither converge nor diverge but will oscillate interminably.



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

$$\begin{array}{ll} \text{Set 1:} & 9x + 3y + z = 13 \\ & 2x + 5y - z = 6 \\ & -6x + 8z = 2 \end{array}$$

$$\begin{array}{ll} \text{Set 2:} & 4x + 2y - 2z = 4 \\ & x + 5y - z = 5 \\ & x + y + 6z = 8 \end{array}$$

At face value, because it is not strictly diagonally dominant, Set 2 would seem to be divergent. However, since it is very close to being diagonally dominant, a solution can be obtained.

The third set is not diagonally dominant and will diverge for most orderings. However, the following arrangement will converge albeit at a very slow rate:

$$\begin{array}{ll} \text{Set 3:} & -3x + 4y + 5z = 6 \\ & 2y - z = 1 \\ & -2x + 2y - 3z = -3 \end{array}$$

11.15 Using MATLAB:

(a) The results for the first system will come out as expected.

```
>> A=[1 4 9;4 9 16;9 16 25]
>> B=[14 29 50]'
```

x =

```
1.0000
1.0000
1.0000
```

```

>> inv(A)

ans =
    3.8750    -5.5000     2.1250
   -5.5000     7.0000    -2.5000
    2.1250    -2.5000     0.8750

>> cond(A,inf)

ans =
    750.0000

```

(b) However, for the 4×4 system, the ill-conditioned nature of the matrix yields poor results:

```

>> A=[1 4 9 16;4 9 16 25;9 16 25 36;16 25 36 49];
>> B=[30 54 86 126]';
>> x=A\B
Warning: Matrix is close to singular or badly scaled.
          Results may be inaccurate. RCOND = 3.037487e-019.

x =
    0.5496
    2.3513
   -0.3513
    1.4504

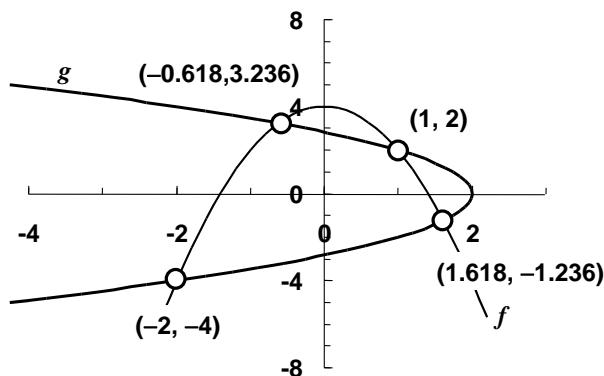
>> cond(A,inf)
Warning: Matrix is close to singular or badly scaled.
          Results may be inaccurate. RCOND = 3.037487e-019.
> In cond at 48

ans =
    3.2922e+018

```

Note that using other software such as Excel yields similar results. For example, the condition number computed with Excel is 5×10^{17} .

11.16 (a) As shown, there are 4 roots, one in each quadrant.



(b) It might be expected that if an initial guess was within a quadrant, the result would be the root in the quadrant. However a sample of initial guesses spanning the range yield the following roots:

6	(-2, -4)	(-0.618, 3.236)	(-0.618, 3.236)	(1, 2)	(-0.618, 3.236)
3	(-0.618, 3.236)	(-0.618, 3.236)	(-0.618, 3.236)	(1, 2)	(-0.618, 3.236)
0	(1, 2)	(1.618, -1.236)	(1.618, -1.236)	(1.618, -1.236)	(1.618, -1.236)
-3	(-2, -4)	(-2, -4)	(1.618, -1.236)	(1.618, -1.236)	(1.618, -1.236)
-6	(-2, -4)	(-2, -4)	(-2, -4)	(1.618, -1.236)	(-2, -4)
	-6	-3	0	3	6

We have highlighted the guesses that converge to the roots in their quadrants. Although some follow the pattern, others jump to roots that are far away. For example, the guess of (-6, 0) jumps to the root in the first quadrant.

This underscores the notion that root location techniques are highly sensitive to initial guesses and that open methods like the Solver can locate roots that are not in the vicinity of the initial guesses.

11.17 Define the quantity of transistors, resistors, and computer chips as x_1 , x_2 and x_3 . The system equations can then be defined as

$$4x_1 + 3x_2 + 2x_3 = 960$$

$$x_1 + 3x_2 + x_3 = 510$$

$$2x_1 + x_2 + 3x_3 = 610$$

The solution can be implemented in Excel as shown below:

	A	B	C	D
1	A:			B:
2	4	3	2	960
3	1	3	1	510
4	2	1	3	610
5				
6	A ₁₁ :			X:
7	0.421053	-0.36842	-0.15789	120
8	-0.05263	0.421053	-0.10526	100
9	-0.26316	0.105263	0.473684	90

The following view shows the formulas that are employed to determine the inverse in cells A7:C9 and the solution in cells D7:D9.

	A	B	C	D
1	A:			B:
2	4	3	2	960
3	1	3	1	510
4	2	1	3	610
5				
6	A ₁₁ :			X:
7	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MMULT(A7:C9,D2:D4)
8	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MMULT(A7:C9,D2:D4)
9	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MINVERSE(A2:C4)	=MMULT(A7:C9,D2:D4)

Here is the same solution generated in MATLAB:

```
>> A=[4 3 2;1 3 1;2 1 3];
>> B=[960 510 610]';
>> x=A\B

x =
    120
    100
    90
```

In both cases, the answer is $x_1 = 120$, $x_2 = 100$, and $x_3 = 90$

11.18 The spectral condition number can be evaluated as

```
>> A = hilb(10);
>> N = cond(A)

N =
1.6025e+013
```

The digits of precision that could be lost due to ill-conditioning can be calculated as

```
>> c = log10(N)

c =
13.2048
```

Thus, about 13 digits could be suspect. A right-hand side vector can be developed corresponding to a solution of ones:

```
>> b=[sum(A(1,:)); sum(A(2,:)); sum(A(3,:)); sum(A(4,:)); sum(A(5,:));
      sum(A(6,:)); sum(A(7,:)); sum(A(8,:)); sum(A(9,:)); sum(A(10,:))]

b =
    2.9290
    2.0199
    1.6032
    1.3468
    1.1682
    1.0349
    0.9307
    0.8467
    0.7773
    0.7188
```

The solution can then be generated by left division

```
>> x = A\b

x =
    1.0000
    1.0000
```

```

1.0000
1.0000
0.9999
1.0003
0.9995
1.0005
0.9997
1.0001

```

The maximum and mean errors can be computed as

```

>> e=max(abs(x-1))

e =
5.3822e-004

>> e=mean(abs(x-1))

e =
1.8662e-004

```

Thus, some of the results are accurate to only about 3 to 4 significant digits. Because MATLAB represents numbers to 15 significant digits, this means that about 11 to 12 digits are suspect.

11.19 First, the Vandermonde matrix can be set up

```

>> x1 = 4;x2=2;x3=7;x4=10;x5=3;x6=5;
>> A = [x1^5 x1^4 x1^3 x1^2 x1 1;x2^5 x2^4 x2^3 x2^2 x2 1;x3^5 x3^4
x3^3 x3^2 x3 1;x4^5 x4^4 x4^3 x4^2 x4 1;x5^5 x5^4 x5^3 x5^2 x5 1;x6^5
x6^4 x6^3 x6^2 x6 1]

```

A =	1024	256	64	16	4	1
	32	16	8	4	2	1
	16807	2401	343	49	7	1
	100000	10000	1000	100	10	1
	243	81	27	9	3	1
	3125	625	125	25	5	1

The spectral condition number can be evaluated as

```

>> N = cond(A)

N =
1.4492e+007

```

The digits of precision that could be lost due to ill-conditioning can be calculated as

```

>> c = log10(N)

c =
7.1611

```

Thus, about 7 digits might be suspect. A right-hand side vector can be developed corresponding to a solution of ones:

```
>> b=[sum(A(1,:));sum(A(2,:));sum(A(3,:));sum(A(4,:));sum(A(5,:));
      sum(A(6,:))]

b =
    1365
     63
   19608
  111111
    364
   3906
```

The solution can then be generated by left division

```
>> format long
>> x=A\b

x =
  1.00000000000000
  0.999999999991
  1.00000000000075
  0.99999999999703
  1.000000000000542
  0.99999999999630
```

The maximum and mean errors can be computed as

```
>> e = max(abs(x-1))

e =
  5.420774940034789e-012

>> e = mean(abs(x-1))

e =
  2.154110223528960e-012
```

Some of the results are accurate to about 12 significant digits. Because MATLAB represents numbers to about 15 significant digits, this means that about 3 digits are suspect. Thus, for this case, the condition number tends to exaggerate the impact of ill-conditioning.

11.20 The flop counts for the tridiagonal algorithm in Fig. 11.2 can be determined as

mult/div	add/subt
Sub DecomP(e, f, g, n)	
Dim k As Integer	
For k = 2 To n	
e(k) = e(k) / f(k - 1)	' (n - 1)
f(k) = f(k) - e(k) * g(k - 1)	' (n - 1) (n - 1)
Next k	
End Sub	
Sub Substitute(e, f, g, r, n, x)	

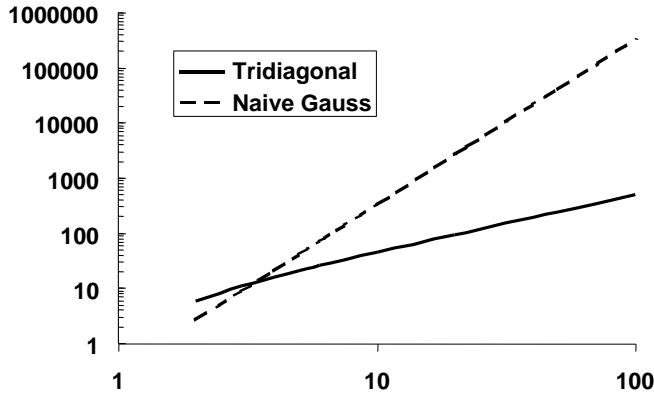
```

Dim k As Integer
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)      ' (n - 1)      (n - 1)
Next k
x(n) = r(n) / f(n)                      '   1
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)  '2(n - 1)      (n - 1)
Next k
End Sub

Sum =                                     5(n-1) + 1      (3n - 3)

```

The multiply/divides and add/subtracts can be summed to yield $8n - 7$ as opposed to $n^3/3$ for naive Gauss elimination. Therefore, a tridiagonal solver is well worth using.



11.21 Here is a VBA macro to obtain a solution for a tridiagonal system using the Thomas algorithm. It is set up to duplicate the results of Example 11.1.

```

Option Explicit

Sub TriDiag()
    Dim i As Integer, n As Integer
    Dim e(10) As Double, f(10) As Double, g(10) As Double
    Dim r(10) As Double, x(10) As Double
    n = 4
    e(2) = -1: e(3) = -1: e(4) = -1
    f(1) = 2.04: f(2) = 2.04: f(3) = 2.04: f(4) = 2.04
    g(1) = -1: g(2) = -1: g(3) = -1
    r(1) = 40.8: r(2) = 0.8: r(3) = 0.8: r(4) = 200.8
    Call Thomas(e, f, g, r, n, x)
    For i = 1 To n
        MsgBox x(i)
    Next i
    End Sub

    Sub Thomas(e, f, g, r, n, x)
        Call Decomp(e, f, g, n)
        Call Substitute(e, f, g, r, n, x)
    End Sub

    Sub Decomp(e, f, g, n)
        Dim k As Integer
        For k = 2 To n
            e(k) = e(k) / f(k - 1)
        Next k
    End Sub

```

```

f(k) = f(k) - e(k) * g(k - 1)
Next k
End Sub

Sub Substitute(e, f, g, r, n, x)
Dim k As Integer
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

11.22 Here is a VBA macro to obtain a solution of a symmetric system with Cholesky decomposition. It is set up to duplicate the results of Example 11.2.

```

Option Explicit

Sub TestChol()
Dim i As Integer, j As Integer
Dim n As Integer
Dim a(10, 10) As Double
n = 3
a(1, 1) = 6: a(1, 2) = 15: a(1, 3) = 55
a(2, 1) = 15: a(2, 2) = 55: a(2, 3) = 225
a(3, 1) = 55: a(3, 2) = 225: a(3, 3) = 979
Call Cholesky(a, n)
'output results to worksheet
Sheets("Sheet1").Select
Range("a3").Select
For i = 1 To n
    For j = 1 To n
        ActiveCell.Value = a(i, j)
        ActiveCell.Offset(0, 1).Select
    Next j
    ActiveCell.Offset(1, -n).Select
Next i
Range("a3").Select
End Sub

Sub Cholesky(a, n)
Dim i As Integer, j As Integer, k As Integer
Dim sum As Double
For k = 1 To n
    For i = 1 To k - 1
        sum = 0
        For j = 1 To i - 1
            sum = sum + a(i, j) * a(k, j)
        Next j
        a(k, i) = (a(k, i) - sum) / a(i, i)
    Next i
    sum = 0
    For j = 1 To k - 1
        sum = sum + a(k, j) ^ 2
    Next j
    a(k, k) = Sqr(a(k, k) - sum)
Next k
End Sub

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

11.23 Here is a VBA macro to obtain a solution of a linear diagonally-dominant system with the Gauss-Seidel method. It is set up to duplicate the results of Example 11.3.

Option Explicit

```

Sub Gausseid()
    Dim n As Integer, imax As Integer, i As Integer
    Dim a(3, 3) As Double, b(3) As Double, x(3) As Double
    Dim es As Double, lambda As Double
    n = 3
    a(1, 1) = 3: a(1, 2) = -0.1: a(1, 3) = -0.2
    a(2, 1) = 0.1: a(2, 2) = 7: a(2, 3) = -0.3
    a(3, 1) = 0.3: a(3, 2) = -0.2: a(3, 3) = 10
    b(1) = 7.85: b(2) = -19.3: b(3) = 71.4
    es = 0.1
    imax = 20
    lambda = 1#
    Call Gseid(a, b, n, x, imax, es, lambda)
    For i = 1 To n
        MsgBox x(i)
    Next i
    End Sub

Sub Gseid(a, b, n, x, imax, es, lambda)
    Dim i As Integer, j As Integer, iter As Integer, sentinel As Integer
    Dim dummy As Double, sum As Double, ea As Double, old As Double
    For i = 1 To n
        dummy = a(i, i)
        For j = 1 To n
            a(i, j) = a(i, j) / dummy
        Next j
        b(i) = b(i) / dummy
    Next i
    For i = 1 To n
        sum = b(i)
        For j = 1 To n
            If i <> j Then sum = sum - a(i, j) * x(j)
        Next j
        x(i) = sum
    Next i
    iter = 1
    Do
        sentinel = 1
        For i = 1 To n
            old = x(i)
            sum = b(i)
            For j = 1 To n
                If i <> j Then sum = sum - a(i, j) * x(j)
            Next j
            x(i) = lambda * sum + (1# - lambda) * old
            If sentinel = 1 And x(i) <> 0 Then
                ea = Abs((x(i) - old) / x(i)) * 100
                If ea > es Then sentinel = 0
            End If
        Next i
        iter = iter + 1
        If sentinel = 1 Or iter >= imax Then Exit Do
    Loop
End Sub

```

CHAPTER 12

12.1 Flow balances can be used to determine

$$\begin{array}{lllll} Q_{01} = 6 & Q_{15} = 3 & Q_{12} = 4 & Q_{31} = 1 & Q_{03} = 8 \\ Q_{25} = 1 & Q_{23} = 1 & Q_{54} = 2 & Q_{55} = 2 & Q_{24} = 2 \\ Q_{34} = 8 & Q_{44} = 12 & & & \end{array}$$

Mass balances can be used to determine the following simultaneous equations,

$$\left[\begin{array}{ccccc} 7 & 0 & -1 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 0 & -1 & 9 & 0 & 0 \\ 0 & -2 & -8 & 12 & -2 \\ -3 & -1 & 0 & 0 & 4 \end{array} \right] \left\{ \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \right\} = \left\{ \begin{array}{c} 240 \\ 0 \\ 80 \\ 0 \\ 0 \end{array} \right\}$$

The solution and the matrix inverse can then be developed. For example, using MATLAB,

```
>> A=[7 0 -1 0 0;
   -4 4 0 0 0;
   0 -1 9 0 0;
   0 -2 -8 12 -2;
   -3 -1 0 0 4];
>> B=[240;0;80;0;0];
>> C=A\B

C =
    36.1290
    36.1290
    12.9032
    20.6452
    36.1290

>> inv(A)

ans =

    0.1452    0.0040    0.0161      0      0
    0.1452    0.2540    0.0161      0      0
    0.0161    0.0282    0.1129      0      0
    0.0591    0.0722    0.0806    0.0833    0.0417
    0.1452    0.0665    0.0161      0    0.2500
```

12.2 The relevant coefficients of the matrix inverse are $a_{13}^{-1} = 0.018868$ and $a_{43}^{-1} = 0.087479$.

Therefore a 25% change in the input to reactor 3 will lead to the following concentration changes to reactors 1 and 4:

$$\Delta c_1 = 0.018868 (0.25 \times 160) = 0.754717$$

$$\Delta c_4 = 0.087479 (0.25 \times 160) = 3.499142$$

These can be expressed as percent changes,

$$\frac{\Delta c_1}{c_1} \times 100\% = \frac{0.754717}{11.50943} \times 100\% = 6.56\%$$

$$\frac{\Delta c_4}{c_4} \times 100\% = \frac{3.499142}{16.99828} \times 100\% = 20.59\%$$

12.3 Because of conservation of flow:

$$Q_{01} + Q_{03} = Q_{44} + Q_{55}$$

12.4 Mass balances can be used to determine the following simultaneous equations,

$$\begin{bmatrix} 8 & 0 & -3 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 0 & -2 & 10 & 0 & 0 \\ 0 & 0 & -7 & 10 & -3 \\ -4 & -2 & 0 & 0 & 6 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{Bmatrix} = \begin{Bmatrix} 50 \\ 0 \\ 160 \\ 0 \\ 0 \end{Bmatrix}$$

The solution can then be developed. For example, using MATLAB,

```
>> A=[8 0 -3 0 0;
   -4 4 0 0 0;
   0 -2 10 0 0;
   0 0 -7 10 -3;
   -4 -2 0 0 6];
>> B=[50;0;160;0;0];
>> C=A\B

C =
    13.2432
    13.2432
    18.6486
    17.0270
    13.2432
```

12.5 Flow balances can be used to determine

$$\begin{array}{lllll} Q_{01}=5 & Q_{15}=3 & Q_{12}=0 & Q_{31}=-2 & Q_{03}=8 \\ Q_{25}=0 & Q_{23}=-7 & Q_{54}=0 & Q_{55}=3 & Q_{24}=7 \\ Q_{34}=3 & Q_{44}=10 & & & \end{array}$$

Mass balances can be used to determine the following simultaneous equations,

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 7 & -7 & 0 & 0 \\ -2 & 0 & 10 & 0 & 0 \\ 0 & -7 & -3 & 10 & 0 \\ -3 & 0 & 0 & 0 & 3 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{Bmatrix} = \begin{Bmatrix} 50 \\ 0 \\ 160 \\ 0 \\ 0 \end{Bmatrix}$$

The solution can then be developed. For example, using MATLAB,

```
>> A=[5 0 0 0 0;
0 7 -7 0 -1;
-2 0 10 0 0;
0 -7 -3 10 0;
-3 0 0 0 3];
>> B=[50;0;160;0;0];
>> C=A\B

C =
    10.0000
    18.0000
    18.0000
    18.0000
    10.0000
```

12.6 Mass balances can be written for each of the reactors as

$$\begin{aligned} 500 - Q_{13}c_1 - Q_{12}c_1 + Q_{21}c_2 &= 0 \\ Q_{12}c_1 - Q_{21}c_2 - Q_{23}c_2 &= 0 \\ 200 + Q_{13}c_1 + Q_{23}c_2 - Q_{33}c_3 &= 0 \end{aligned}$$

Values for the flows can be substituted and the system of equations can be written in matrix form as

$$\begin{bmatrix} 130 & -30 & 0 \\ -90 & 90 & 0 \\ -40 & -60 & 120 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} 500 \\ 0 \\ 200 \end{Bmatrix}$$

The solution can then be developed. For example, using MATLAB,

```
>> A=[130 -30 0;-90 90 0;-40 -60 120];
>> B=[500;0;200];
>> C=A\B

C =
    5.0000
    5.0000
    5.8333
```

12.7 Mass balances can be written for each of the lakes as

Superior, c_1 :

$$180 = 67c_1$$

Michigan, c_2 :

$$710 = 36c_2$$

Huron, c_3 :

$$740 + 67c_1 + 36c_2 = 161c_3$$

Erie, c_4 :

$$3850 + 161c_3 = 182c_4$$

Ontario, c_5 :

$$4720 + 182c_4 = 212c_5$$

The system of equations can be written in matrix form as

$$\begin{bmatrix} 67 & 0 & 0 & 0 & 0 \\ 0 & 36 & 0 & 0 & 0 \\ -67 & -36 & 161 & 0 & 0 \\ 0 & 0 & -161 & 182 & 0 \\ 0 & 0 & 0 & -182 & 212 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 180 \\ 710 \\ 740 \\ 3850 \\ 4720 \end{bmatrix}$$

The solution can then be developed. For example, using MATLAB,

```
>> A=[67 0 0 0 0;
0 36 0 0 0;
-67 -36 161 0 0;
0 0 -161 182 0;
0 0 0 -182 212];
>> B=[180 710 740 3850 4720]';
>> C=A\B

C =
2.6866
19.7222
10.1242
30.1099
48.1132
```

12.8 (a) The solution can be developed using your own software or a package. For example, using MATLAB,

```
>> A=[13.422 0 0 0;
-13.422 12.252 0 0;
0 -12.252 12.377 0;
0 0 -12.377 11.797];
>> W=[750.5 300 102 30]';
>> AI=inv(A)

AI =
0.0745 0 0 0
0.0816 0.0816 0 0
0.0808 0.0808 0.0808 0
```

```

0.0848      0.0848      0.0848      0.0848
>> C=AI*w

C =
55.9157
85.7411
93.1163
100.2373

```

(b) The element of the matrix that relates the concentration of Havasu (lake 4) to the loading of Powell (lake 1) is $a_{41}^{-1} = 0.084767$. This value can be used to compute how much the loading to Lake Powell must be reduced in order for the chloride concentration of Lake Havasu to be 75 as

$$\Delta W_1 = \frac{\Delta c_4}{a_{41}^{-1}} = \frac{100.2373 - 75}{0.084767} = 297.725$$

(c) First, normalize the matrix to give

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -0.91283 & 0 & 0 \\ 0 & -0.9899 & 1 & 0 \\ 0 & 0 & 1 & -0.95314 \end{bmatrix}$$

The column-sum norm for this matrix is 2. The inverse of the matrix can be computed as

$$[A]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1.095495 & -1.09549 & 0 & 0 \\ 1.084431 & -1.08443 & 1 & 0 \\ 1.137747 & -1.13775 & 1.049165 & -1.04917 \end{bmatrix}$$

The column-sum norm for the inverse can be computed as 4.317672. The condition number is, therefore, $2(4.317672) = 8.635345$. This means that less than 1 digit is suspect [$\log_{10}(8.635345) = 0.93628$]. Interestingly, if the original matrix is unscaled, the same condition number results.

12.9 For the first stage, the mass balance can be written as

$$F_1 y_{\text{in}} + F_2 x_2 = F_2 x_1 + F_1 x_1$$

Substituting $x = Ky$ and rearranging gives

$$-\left(1 + \frac{F_2}{F_1} K\right) y_1 + \frac{F_2}{F_1} K y_2 = -y_{\text{in}}$$

Using a similar approach, the equation for the last stage is

$$y_4 - \left(1 + \frac{F_2}{F_1} K\right) y_5 = -\frac{F_2}{F_1} x_{in}$$

For interior stages,

$$y_{i-1} - \left(1 + \frac{F_2}{F_1} K\right) y_i + \frac{F_2}{F_1} K y_{i+1} = 0$$

These equations can be used to develop the following system,

$$\begin{bmatrix} 9 & -8 & 0 & 0 & 0 \\ -1 & 9 & -8 & 0 & 0 \\ 0 & -1 & 9 & -8 & 0 \\ 0 & 0 & -1 & 9 & -8 \\ 0 & 0 & 0 & -1 & 9 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The solution can be developed in a number of ways. For example, using MATLAB,

```
>> format long
>> A=[ 9 -8 0 0 0 ;
-1 9 -8 0 0 ;
0 -1 9 -8 0 ;
0 0 -1 9 -8 ;
0 0 0 -1 9 ];
>> B=[ 0.1;0;0;0;0 ];
>> Y=A\B

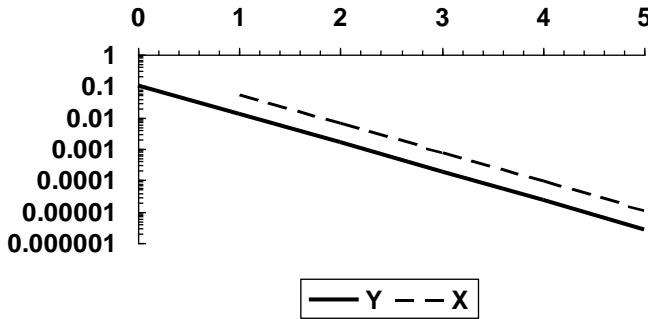
Y =
0.01249966621272
0.00156212448931
0.00019493177388
0.00002403268445
0.00000267029827
```

Note that the corresponding values of X can be computed as

```
>> X=4*Y

X =
0.04999866485086
0.00624849795722
0.00077972709552
0.00009613073780
0.00001068119309
```

Therefore, $y_{out} = 0.0000026703$ and $x_{out} = 0.05$. In addition, here is a logarithmic plot of the simulation results versus stage,



12.10 Steady-state mass balances for A in each reactor can be written as

$$\begin{aligned} Q_{\text{in}}c_{A,\text{in}} - Q_{\text{in}}c_{A,1} - k_1V_1c_{A,1} &= 0 \\ Q_{\text{in}}c_{A,1} + Q_{32}c_{A,3} - (Q_{\text{in}} + Q_{32})c_{A,2} - k_2V_2c_{A,2} &= 0 \\ (Q_{\text{in}} + Q_{32})c_{A,2} + Q_{43}c_{A,4} - (Q_{\text{in}} + Q_{43})c_{A,3} - k_3V_3c_{A,3} &= 0 \\ (Q_{\text{in}} + Q_{43})c_{A,3} - (Q_{\text{in}} + Q_{43})c_{A,4} - k_4V_4c_{A,4} &= 0 \end{aligned}$$

Steady-state mass balances for B in each reactor can be written as

$$\begin{aligned} -Q_{\text{in}}c_{B,1} + k_1V_1c_{A,1} &= 0 \\ Q_{\text{in}}c_{B,1} + Q_{32}c_{B,3} - (Q_{\text{in}} + Q_{32})c_{B,2} + k_2V_2c_{A,2} &= 0 \\ (Q_{\text{in}} + Q_{32})c_{B,2} + Q_{43}c_{B,4} - (Q_{\text{in}} + Q_{43})c_{B,3} + k_3V_3c_{A,3} &= 0 \\ (Q_{\text{in}} + Q_{43})c_{B,3} - (Q_{\text{in}} + Q_{43})c_{B,4} + k_4V_4c_{A,4} &= 0 \end{aligned}$$

Values for the parameters can be substituted and the system of equations can be written in matrix form as

$$\left[\begin{array}{ccccccccc} 11.875 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.875 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 26.25 & 0 & -5 & 0 & 0 & 0 \\ 0 & -10 & -11.25 & 15 & 0 & -5 & 0 & 0 \\ 0 & 0 & -15 & 0 & 53 & 0 & -3 & 0 \\ 0 & 0 & 0 & -15 & -40 & 13 & 0 & -3 \\ 0 & 0 & 0 & 0 & -13 & 0 & 15.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & -13 & -2.5 & 13 \end{array} \right] \begin{Bmatrix} c_{A,1} \\ c_{B,1} \\ c_{A,2} \\ c_{B,2} \\ c_{A,3} \\ c_{B,3} \\ c_{A,4} \\ c_{B,4} \end{Bmatrix} = \begin{Bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

The solution can then be developed. For example, using MATLAB,

```
>> A=[11.875 0 0 0 0 0 0 0;
-1.875 10 0 0 0 0 0 0;
-10 0 26.25 0 -5 0 0 0;
0 -10 -11.25 15 0 -5 0 0;
0 0 -15 0 53 0 -3 0;
0 0 0 -15 40 13 0 -3;
0 0 0 0 -13 0 15.5 0;
0 0 0 0 0 -13 -2.5 13];
```

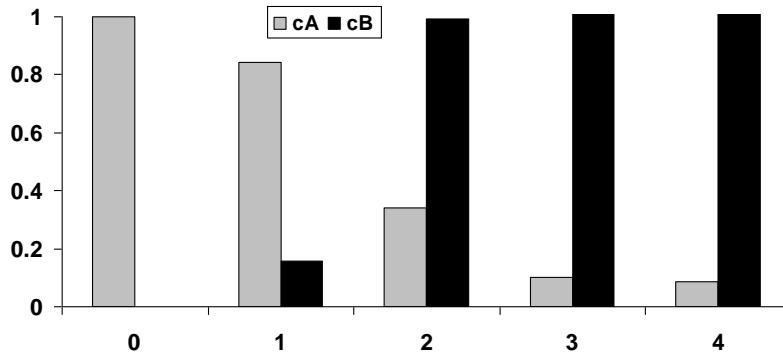
```
>> B=[1 0 0 0 0 0 0 0]';
>> C=A\B

C =
0.8421
0.1579
0.3400
0.9933
0.1010
1.8990
0.0847
1.9153
```

Therefore, to summarize the results

reactor	A	B
inflow	1	0
1	0.842105	0.157895
2	0.340047	0.993286
3	0.101036	1.898964
4	0.084740	1.915260

Here is a plot of the results:



12.11 Assuming a unit flow for Q_1 , the simultaneous equations can be written in matrix form as

$$\begin{bmatrix} -2 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & -2 & 3 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \begin{Bmatrix} Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{Bmatrix}$$

These equations can then be solved. For example, using MATLAB,

```
>> A=[-2 1 2 0 0 0;
0 0 -2 1 2 0;
0 0 0 0 -2 3;
```

```

1 1 0 0 0 0;
0 1 -1 -1 0 0;
0 0 0 1 -1 -1];
>> B=[0 0 0 1 0 0 ]';
>> Q=A\B

Q =
0.5059
0.4941
0.2588
0.2353
0.1412
0.0941

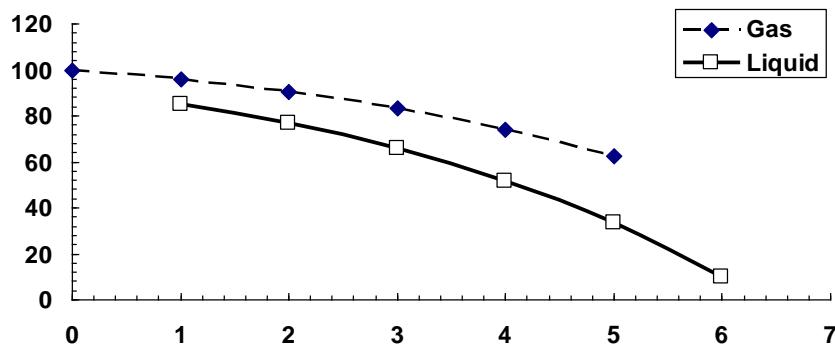
```

12.12 The mass balances can be expressed in matrix form as

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

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

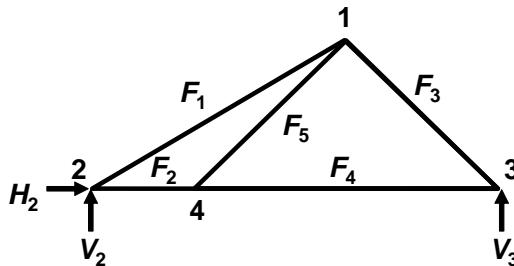


12.13 Let x_i = the volume taken from pit i . Therefore, the following system of equations must hold

$$\begin{aligned} 0.55x_1 + 0.25x_2 + 0.25x_3 &= 4800 \\ 0.30x_1 + 0.45x_2 + 0.20x_3 &= 5800 \\ 0.15x_1 + 0.30x_2 + 0.55x_3 &= 5700 \end{aligned}$$

These can then be solved for $x_1 = 2416.667$, $x_2 = 9193.333$, and $x_3 = 4690$.

12.14 We can number the nodes as



Node 1:

$$\Sigma F_H = 0 = -F_1 \cos 30^\circ - F_5 \cos 45^\circ + F_3 \cos 45^\circ + 1200$$

$$\Sigma F_V = 0 = -F_1 \sin 30^\circ - F_5 \sin 45^\circ - F_3 \sin 45^\circ - 600$$

Node 2:

$$\Sigma F_H = 0 = H_2 + F_2 + F_1 \cos 30^\circ$$

$$\Sigma F_V = 0 = F_1 \sin 30^\circ + V_2$$

Node 3:

$$\Sigma F_H = 0 = -F_4 - F_3 \cos 45^\circ$$

$$\Sigma F_V = 0 = V_3 + F_3 \sin 45^\circ$$

Node 4:

$$\Sigma F_H = 0 = -F_2 + F_4 + F_5 \cos 45^\circ$$

$$\Sigma F_V = 0 = F_5 \sin 45^\circ - 500$$

These balances can then be expressed in matrix form as

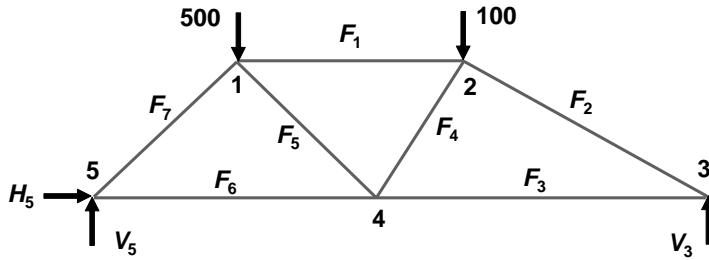
$$\left[\begin{array}{ccccccc} 0.866 & 0 & -0.707 & 0 & 0.707 & 0 & 0 \\ 0.5 & 0 & -0.707 & 0 & 0.707 & 0 & 0 \\ -0.866 & -1 & 0 & 0 & 0 & -1 & 0 \\ -0.5 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0.707 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.707 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & -0.707 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.707 & 0 & 0 \end{array} \right] \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ H_2 \\ V_2 \\ V_3 \end{Bmatrix} = \begin{Bmatrix} 1200 \\ -600 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -500 \end{Bmatrix}$$

This system can be solved for

$$\begin{array}{llll} F_1 = -292.82 & F_2 = 1453.59 & F_3 = -1348.58 & F_4 = 953.5898 \\ F_5 = 707.1068 & F_6 = -1200 & F_7 = 146.4102 & V_2 = 146.4102 \\ & & & V_3 = 953.5898 \end{array}$$

Note that the horizontal reactions ($H_2 = -1200$) and the vertical reactions ($V_2 + V_3 = 146.4102 + 953.5898 = 1100$) are equal to the negative of the imposed loads. This is a good check that the computation is correct.

12.15 We can number the nodes as



Node 1:

$$\Sigma F_H = 0 = F_1 + F_5 \cos 45^\circ - F_7 \cos 45^\circ$$

$$\Sigma F_V = 0 = -F_5 \sin 45^\circ - F_7 \sin 45^\circ - 500$$

Node 2:

$$\Sigma F_H = 0 = -F_1 + F_2 \cos 30^\circ - F_4 \cos 60^\circ$$

$$\Sigma F_V = 0 = -F_2 \sin 30^\circ - F_4 \sin 60^\circ - 100$$

Node 3:

$$\Sigma F_H = 0 = -F_2 \cos 30^\circ - F_3$$

$$\Sigma F_V = 0 = V_3 + F_2 \sin 30^\circ$$

Node 4:

$$\Sigma F_H = 0 = F_3 + F_4 \cos 60^\circ - F_5 \cos 45^\circ - F_6$$

$$\Sigma F_V = 0 = F_4 \sin 60^\circ + F_5 \sin 45^\circ$$

Node 5:

$$\Sigma F_H = 0 = F_6 + F_7 \cos 45^\circ + H_5$$

$$\Sigma F_V = 0 = F_7 \sin 45^\circ + V_5$$

These balances can then be expressed in matrix form as

$$\begin{bmatrix} -1 & 0 & 0 & 0 & -0.707 & 0 & 0.707 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.707 & 0 & 0.707 & 0 & 0 & 0 \\ 1 & -0.866 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.866 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.866 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -0.5 & 0.707 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.866 & -0.707 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -0.707 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.707 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ V_3 \\ H_5 \\ V_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -500 \\ 0 \\ -100 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

This system can be solved for

$$\begin{aligned} F_1 &= -348.334 & F_2 &= -351.666 & F_3 &= 304.5517 & F_4 &= 87.56443 & F_5 &= -107.244 \\ F_6 &= 424.167 & F_7 &= -599.863 & V_3 &= 175.833 & H_5 &= 0 & V_5 &= 424.167 \end{aligned}$$

12.16 The first two columns of the inverse provide the information to solve this problem

	F_{1H}	F_{1V}
F_1	0.866025	0.500000
F_2	0.250000	-0.433013
F_3	-0.500000	0.866025
H_2	-1.000000	0.000000
V_2	-0.433013	-0.250000
V_3	0.433013	-0.750000

$$\begin{aligned} F_1 &= 2000(0.866025) - 2500(0.5) = 482.0508 \\ F_2 &= 2000(0.25) - 2500(-0.433013) = 1582.532 \\ F_3 &= 2000(-0.5) - 2500(0.866025) = -3165.06 \\ H_2 &= 2000(-1) - 2500(0) = -2000 \\ V_2 &= 2000(-0.433013) - 2500(-0.25) = -241.025 \\ V_3 &= 2000(0.433013) - 2500(-0.75) = 2741.025 \end{aligned}$$

12.17

$$\begin{aligned} \Sigma F_y &= 0 & V_2 + V_3 &= 1000 \\ \Sigma M &= 0 & 1000(\cos 30^\circ)L_1 - V_3 L_2 & \\ \text{Geometry} & & \cos 30^\circ L_1 + \cos 60^\circ L_3 &= L_2 \end{aligned}$$

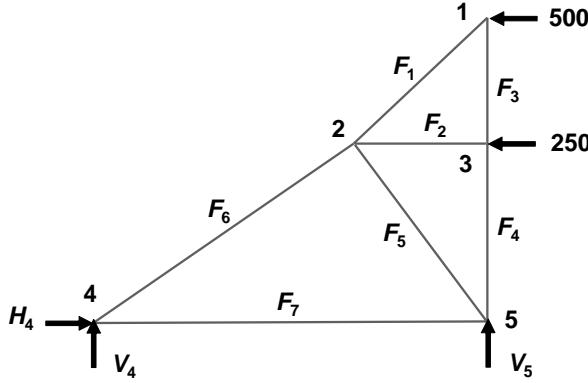
Since $V_2 = 250$ and $V_3 = 750$,

$$\begin{aligned} 866L_1 - 750L_2 &= 0 \\ 0.866L_1 + 0.5L_3 &= L_2 \end{aligned}$$

Therefore,

$$L_3 = \frac{L_2 - 0.866L_1}{0.5}$$

12.18 We can number the nodes as



Node 1:

$$\Sigma F_H = 0 = -F_1 \cos 45^\circ - 500$$

$$\Sigma F_V = 0 = -F_1 \sin 45^\circ - F_3$$

Node 2:

$$\Sigma F_H = 0 = F_1 \cos 45^\circ + F_2 + F_5 \cos 60^\circ - F_6 \cos 30^\circ$$

$$\Sigma F_V = 0 = F_1 \sin 45^\circ - F_5 \sin 60^\circ - F_6 \sin 30^\circ$$

Node 3:

$$\Sigma F_H = 0 = -F_2 - 250$$

$$\Sigma F_V = 0 = F_3 - F_4$$

Node 4:

$$\Sigma F_H = 0 = F_6 \cos 30^\circ + F_7 + H_4$$

$$\Sigma F_V = 0 = F_6 \sin 30^\circ + V_4$$

Node 5:

$$\Sigma F_H = 0 = -F_7 - F_5 \cos 60^\circ$$

$$\Sigma F_V = 0 = F_4 + F_5 \sin 60^\circ + V_5$$

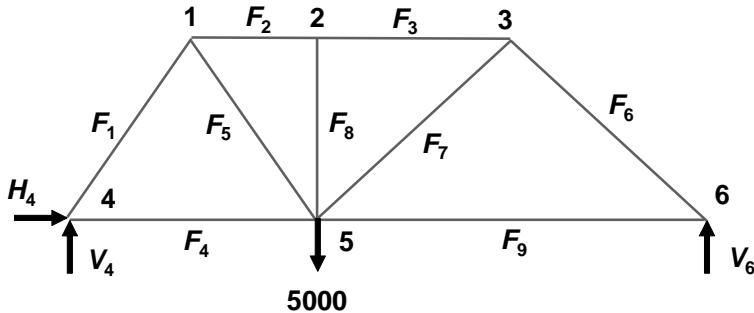
These balances can then be expressed in matrix form as

$$\begin{bmatrix} 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.707 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.707 & -1 & 0 & 0 & -0.5 & 0.866 & 0 & 0 & 0 & 0 \\ -0.707 & 0 & 0 & 0 & 0.866 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.866 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -0.866 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ H_4 \\ V_4 \\ V_5 \end{Bmatrix} = \begin{Bmatrix} -500 \\ 0 \\ 0 \\ 0 \\ -250 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

This system can be solved for

$$\begin{aligned} F_1 &= -707.107 & F_2 &= -250 & F_3 &= 500 & F_4 &= 500 & F_5 &= -58.0127 \\ F_6 &= -899.519 & F_7 &= 29.00635 & H_4 &= 750 & V_4 &= 449.7595 & V_5 &= -449.76 \end{aligned}$$

12.19 We can number the nodes as



Node 1:

$$\Sigma F_H = 0 = -F_1 \cos 60^\circ + F_2 + F_5 \cos 60^\circ$$

$$\Sigma F_V = 0 = -F_1 \sin 60^\circ - F_5 \sin 60^\circ$$

Node 2:

$$\Sigma F_H = 0 = -F_2 + F_3$$

$$\Sigma F_V = 0 = -F_8$$

Node 3:

$$\Sigma F_H = 0 = -F_3 + F_6 \cos 45^\circ - F_7 \cos 45^\circ$$

$$\Sigma F_V = 0 = -F_6 \sin 45^\circ - F_7 \sin 45^\circ$$

Node 4:

$$\Sigma F_H = 0 = F_1 \cos 30^\circ + F_4 + H_4$$

$$\Sigma F_V = 0 = F_1 \sin 60^\circ + V_4$$

Node 5:

$$\Sigma F_H = 0 = -F_4 - F_5 \cos 60^\circ + F_7 \cos 45^\circ + F_9$$

$$\Sigma F_V = 0 = F_5 \sin 60^\circ + F_8 + F_7 \sin 45^\circ - 5000$$

Node 6:

$$\Sigma F_H = 0 = -F_6 \cos 45^\circ - F_9$$

$$\Sigma F_V = 0 = F_6 \sin 45^\circ + V_6$$

Note that $F_8 = 0$. Thus, the middle member is unnecessary unless there is a load with a nonzero vertical component at node 2. These balances can then be expressed in matrix form as

$$\begin{bmatrix} 0.5 & -1 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.866 & 0 & 0 & 0 & 0.866 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -0.707 & 0.707 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.707 & 0.707 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -0.866 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.5 & 0 & -0.707 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.866 & 0 & -0.707 & 0 & 0 & 0 & 0 & H_4 \\ 0 & 0 & 0 & 0 & 0 & 0.707 & 0 & 1 & 0 & 0 & 0 & V_4 \\ 0 & 0 & 0 & 0 & 0 & -0.707 & 0 & 0 & 0 & 0 & 1 & V_6 \end{bmatrix} \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_9 \\ H_4 \\ V_4 \\ V_6 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -5000 \\ 0 \\ 0 \end{Bmatrix}$$

This system can be solved for

$$\begin{aligned} F_1 &= -3660.25 & F_2 &= -3660.25 & F_3 &= -3660.25 & F_4 &= 1830.127 & F_5 &= -3660.25 \\ F_6 &= -2588.19 & F_7 &= 2588.19 & F_9 &= 1830.13 & H_4 &= 0 & V_4 &= 3169.87 \\ V_6 &= 1830.13 \end{aligned}$$

12.20 (a)

Room 1:

$$0 = W_{\text{smoker}} + Q_a c_a - Q_a c_1 + E_{13}(c_3 - c_1)$$

Room 2:

$$0 = Q_b c_b + (Q_a - Q_d) c_4 - Q_c c_2 + E_{24}(c_4 - c_2)$$

Room 3:

$$0 = W_{\text{grill}} + Q_a c_1 + E_{13}(c_1 - c_3) + E_{34}(c_4 - c_3) - Q_a c_3$$

Room 4:

$$0 = Q_a c_3 + E_{34}(c_3 - c_4) + E_{24}(c_2 - c_4) - Q_a c_4$$

Substituting the parameters yields

$$\begin{bmatrix} 225 & 0 & -25 & 0 \\ 0 & 175 & 0 & -125 \\ -225 & 0 & 275 & -50 \\ 0 & -25 & -250 & 275 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 1400 \\ 100 \\ 2000 \\ 0 \end{Bmatrix}$$

These can be solved for

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 8.0996 \\ 12.3448 \\ 16.8966 \\ 16.4828 \end{bmatrix}$$

(b) The matrix inverse can be determined as

$$[A]^{-1} = \begin{bmatrix} 0.004996 & 0.0000153 & 0.000552 & 0.000107 \\ 0.003448 & 0.006207 & 0.003448 & 0.003448 \\ 0.004966 & 0.000138 & 0.004966 & 0.000966 \\ 0.004828 & 0.00069 & 0.004828 & 0.004828 \end{bmatrix}$$

The percent of the carbon monoxide in the kids' section due to each source can be computed as

(i) the smokers

$$c_{2,\text{smokers}} = a_{21}^{-1} W_{\text{smokers}} = 0.003448(1000) = 3.448$$

$$\%_{\text{smokers}} = \frac{3.448}{12.3448} \times 100\% = 27.93\%$$

(ii) the grill

$$c_{2,\text{grill}} = a_{31}^{-1} W_{\text{grill}} = 0.003448(2000) = 6.897$$

$$\%_{\text{grill}} = \frac{6.897}{12.3448} \times 100\% = 55.87\%$$

(iii) the intakes

$$c_{2,\text{intakes}} = a_{21}^{-1} Q_a c_a + a_{22}^{-1} Q_b c_b = 0.003448(200)2 + 0.006207(50)2 = 1.37931 + 0.62069 = 2$$

$$\%_{\text{grill}} = \frac{2}{12.3448} \times 100\% = 16.20\%$$

(c) If the smoker and grill loads are increased by 1000 and 3000 mg/hr, respectively, the concentration in the kids' section will be increased by

$$\begin{aligned} \Delta c_2 &= a_{21}^{-1} \Delta W_{\text{smoker}} + a_{23}^{-1} \Delta W_{\text{grill}} = 0.003448(2000 - 1000) + 0.003448(5000 - 2000) \\ &= 3.448 + 10.3448 = 13.7931 \end{aligned}$$

(d) If the mixing between the kids' area and zone 4 is decreased to 5, the system of equations is changed to

$$\begin{bmatrix} 225 & 0 & -25 & 0 \\ 0 & 155 & 0 & -105 \\ -225 & 0 & 275 & -50 \\ 0 & -5 & -250 & 255 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 1400 \\ 100 \\ 2000 \\ 0 \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 8.1084 \\ 12.0800 \\ 16.9760 \\ 16.8800 \end{Bmatrix}$$

Therefore, the concentration in the kids' area would be decreased 0.26483 mg/m^3 or 2.145%.

12.21 The coordinates of the connection points are

$$D: (0, 0, 2.4)$$

$$A: (0.8, -0.6, 0)$$

$$B: (-0.8, -0.6, 0)$$

$$C: (0, 1, 0)$$

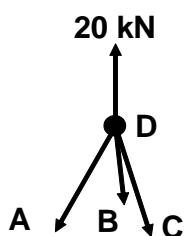
The lengths of the legs can be computed as

$$DA = \sqrt{0.8^2 + 0.6^2 + 2.4^2} = 2.6$$

$$DB = \sqrt{0.8^2 + 0.6^2 + 2.4^2} = 2.6$$

$$DC = \sqrt{0^2 + 1^2 + 2.4^2} = 2.6$$

Assume that each leg is in tension, which mean that each pulls on point D .



The direction cosines of the vectors A, B and C can be determined as

$$A: \left(\frac{4}{13}, -\frac{3}{13}, -\frac{12}{13} \right)$$

$$\mathbf{B}: \left(-\frac{4}{13}, -\frac{3}{13}, -\frac{12}{13} \right)$$

$$\mathbf{C}: \left(0, \frac{5}{13}, -\frac{12}{13} \right)$$

Force balances for point D can then be written as

$$\sum F_x = \frac{4}{13}A - \frac{4}{13}B = 0$$

$$\sum F_y = -\frac{3}{13}A - \frac{3}{13}B + \frac{5}{13}C = 0$$

$$\sum F_z = -\frac{12}{13}A - \frac{12}{13}B - \frac{12}{13}C + 20 = 0$$

Thus, the solution amounts to solving the following system of linear algebraic equations

$$\begin{aligned} 0.30769A - 0.30769B &= 0 \\ -0.23077A - 0.23077B + 0.38462C &= 0 \\ -0.92308A - 0.92308B - 0.92308C &= -20 \end{aligned}$$

These equations can be solved with Gauss elimination for $A = 6.7708$, $B = 6.7708$, and $C = 8.125$.

12.22 The solution can be generated in a number of ways. For example, using MATLAB,

```
>> A=[1 0 0 0 0 0 0 0 1 0;
       0 0 1 0 0 0 0 1 0 0;
       0 1 0 3/5 0 0 0 0 0 0;
       -1 0 0 -4/5 0 0 0 0 0 0;
       0 -1 0 0 0 0 3/5 0 0 0;
       0 0 0 0 -1 0 -4/5 0 0 0;
       0 0 -1 -3/5 0 1 0 0 0 0;
       0 0 0 4/5 1 0 0 0 0 0;
       0 0 0 0 0 -1 -3/5 0 0 0;
       0 0 0 0 0 0 4/5 0 0 1];
>> B=[0 0 -74 0 0 24 0 0 0 0]';
>> x=A\B

x =
    37.3333
   -46.0000
    74.0000
   -46.6667
    37.3333
   46.0000
   -76.6667
   -74.0000
   -37.3333
```

61.3333

Therefore, in kN

$$\begin{array}{lllll} AB = 37.3333 & BC = -46 & AD = 74 & BD = -46.6667 & CD = 37.3333 \\ DE = 46 & CE = -76.6667 & A_x = -74 & A_y = -37.33333 & E_y = 61.3333 \end{array}$$

12.23 The simultaneous equations are

$$\left[\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 5 & -15 & 0 & -5 & -2 \\ 10 & -5 & 0 & -25 & 0 & 0 \end{array} \right] \begin{Bmatrix} i_{12} \\ i_{52} \\ i_{32} \\ i_{65} \\ i_{54} \\ i_{43} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 200 \end{Bmatrix}$$

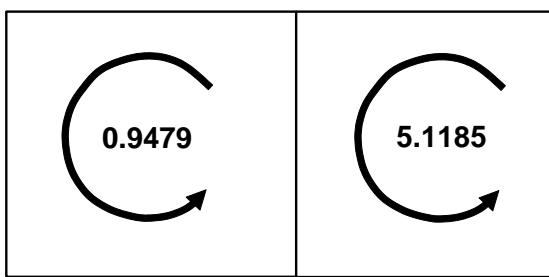
This system can be solved in a number of ways. For example, using MATLAB,

```
>> A=[1 1 1 0 0 0;
0 -1 0 1 -1 0;
0 0 -1 0 0 1;
0 0 0 0 1 -1;
0 5 -15 0 -5 -2;
10 -5 0 -25 0 0];
>> B=[0 0 0 0 0 200]';
>> I=A\B

I =
5.1185
-4.1706
-0.9479
-5.1185
-0.9479
-0.9479
```

$$i_{21} = 5.1185 \quad i_{52} = -4.1706 \quad i_{32} = -0.9479 \quad i_{65} = -5.1185 \quad i_{54} = -0.9479 \quad i_{43} = -0.9479$$

Here are the resulting currents superimposed on the circuit:



12.24 The current equations can be written as

$$\begin{aligned} -i_{21} - i_{23} + i_{52} &= 0 \\ i_{23} - i_{35} + i_{43} &= 0 \\ -i_{43} + i_{54} &= 0 \\ i_{35} - i_{52} + i_{65} - i_{54} &= 0 \end{aligned}$$

Voltage equations:

$$i_{21} = \frac{V_2 - 10}{35} \quad i_{54} = \frac{V_5 - V_4}{15}$$

$$i_{23} = \frac{V_2 - V_3}{30} \quad i_{35} = \frac{V_3 - V_5}{7}$$

$$i_{43} = \frac{V_4 - V_3}{8} \quad i_{52} = \frac{V_5 - V_2}{10}$$

$$i_{65} = \frac{150 - V_5}{5}$$

$$\left[\begin{array}{cccccccccc} -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 35 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 1 \end{array} \right] \begin{Bmatrix} i_{21} \\ i_{23} \\ i_{52} \\ i_{35} \\ i_{43} \\ i_{54} \\ i_{65} \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 150 \end{Bmatrix}$$

This system can be solved for

$$\begin{aligned} i_{21} &= 2.9291 & i_{23} &= -0.6457 & i_{52} &= 2.2835 & i_{35} &= -0.4950 & i_{43} &= 0.1507 \\ i_{54} &= 0.1507 & i_{65} &= 2.9291 & V_2 &= 112.5196 & V_3 &= 131.8893 & V_4 &= 133.0945 \\ V_5 &= 135.3543 \end{aligned}$$

12.25 The current equations can be written as

$$\begin{aligned} i_{32} - i_{25} + i_{12} &= 0 \\ -i_{32} - i_{34} + i_{63} &= 0 \\ i_{34} - i_{47} &= 0 \\ i_{25} + i_{65} - i_{58} &= 0 \end{aligned}$$

$$\begin{aligned} i_{76} - i_{63} - i_{65} &= 0 \\ i_{47} - i_{76} + i_{97} &= 0 \\ i_{58} - i_{89} - i_{80} &= 0 \\ i_{89} - i_{97} &= 0 \end{aligned}$$

Voltage equations:

$$\begin{aligned} -20i_{25} + 10i_{65} - 5i_{63} - 5i_{32} &= 0 \\ 5i_{63} + 20i_{76} + 5i_{47} + 20i_{34} &= 0 \\ -50i_{58} - 15i_{89} - 0i_{97} - 20i_{76} - 10i_{65} &= 0 \\ 120 - 20i_{25} - 50i_{58} &= 40 \end{aligned}$$

$$\left[\begin{array}{cccccccccc|c} 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 20 & 0 & 0 & 5 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -20 & -5 & -5 & 0 & 0 & -20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 50 & 20 & 0 & 15 \\ 0 & 20 & 0 & 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 \end{array} \right] \begin{Bmatrix} i_{32} \\ i_{25} \\ i_{12} \\ i_{34} \\ i_{63} \\ i_{47} \\ i_{65} \\ i_{58} \\ i_{76} \\ i_{97} \\ i_{89} \\ i_{80} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 80 \end{Bmatrix}$$

This system can be solved for

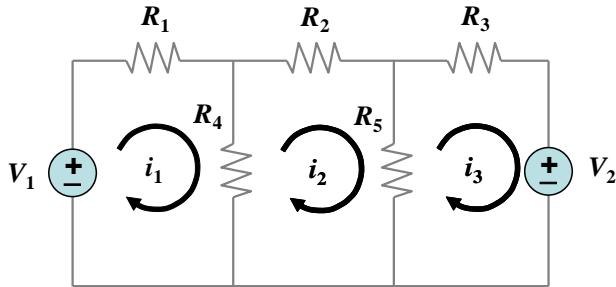
$$\begin{aligned} i_{32} &= -2.5670 & i_{25} &= 1.0449 & i_{12} &= 3.6119 & i_{34} &= 1.2287 & i_{63} &= -1.3384 \\ i_{47} &= 1.2287 & i_{65} &= 0.1371 & i_{58} &= 1.1820 & i_{76} &= -1.2012 & i_{97} &= -2.4299 \\ i_{89} &= -2.4299 & i_{80} &= 3.6119 & & & & & \end{aligned}$$

12.26 Let c_i = component i . Therefore, the following system of equations must hold

$$\begin{aligned} 15c_1 + 17c_2 + 19c_3 &= 3890 \\ 0.30c_1 + 0.40c_2 + 0.55c_3 &= 95 \\ 1.0c_1 + 1.2c_2 + 1.5c_3 &= 282 \end{aligned}$$

These can then be solved for $c_1 = 90$, $c_2 = 60$, and $c_3 = 80$.

12.27 First, we can number the loops and assume that the currents are clockwise.



Kirchhoff's voltage law can be applied to each loop.

$$\begin{aligned} -V_1 + R_1 i_1 + R_4(i_1 - i_2) &= 0 \\ R_4(i_2 - i_1) + R_2 i_2 + R_5(i_2 - i_3) &= 0 \\ R_5(i_3 - i_2) + R_3 i_3 + V_2 &= 0 \end{aligned}$$

Collecting terms, the system can be written in matrix form as

$$\begin{bmatrix} 20 & -15 & 0 \\ -15 & 50 & -25 \\ 0 & -25 & 45 \end{bmatrix} \begin{Bmatrix} i_1 \\ i_2 \\ i_3 \end{Bmatrix} = \begin{Bmatrix} 80 \\ 0 \\ -50 \end{Bmatrix}$$

This can be solved with a tool like MATLAB,

```
> A=[20 -15 0;-15 50 -25;0 -25 45];
>> B=[80;0;-50];
```

```
>> I=A\B
```

I =

4.9721
1.2961
-0.3911

Therefore, $I_1 = 4.9721$, $I_2 = 1.2961$, and $I_3 = -0.3911$.

12.28 This problem can be solved by applying Kirchhoff's voltage law to each loop.

$$\begin{aligned} -20 + 4(i_1 - i_2) + 2(i_1 - i_3) &= 0 \\ 4(i_2 - i_1) + 6i_2 + 8(i_2 - i_3) &= 0 \\ 8(i_3 - i_2) + 5i_3 + 2(i_3 - i_1) &= 0 \end{aligned}$$

Collecting terms, the system can be written in matrix form as

$$\begin{bmatrix} 6 & -4 & -2 \\ -4 & 18 & -8 \\ -2 & -8 & 15 \end{bmatrix} \begin{Bmatrix} i_1 \\ i_2 \\ i_3 \end{Bmatrix} = \begin{Bmatrix} 20 \\ 0 \\ 0 \end{Bmatrix}$$

This can be solved with a tool like MATLAB,

```

> A=[6 -4 -2;-4 18 -8;-2 -8 15];
>> B=[20;0;0];
>> I=A\B

I =
5.1759
1.9095
1.7085

```

Therefore, $I_1 = 5.1759$, $I_2 = 1.9095$, and $I_3 = 1.7085$.

12.29 This problem can be solved directly on a calculator capable of doing matrix operations or on MATLAB.

```

>> b=[-200;-250;100];
>> a=[55 0 -25;0 -37 -4;-25 -4 29];
>> b=[-200;-250;100];
>> x=a\b

x =
-2.7278
6.5407
1.9989

```

Therefore, $I_1 = -2.7278$ A, $I_2 = 6.5407$ A, and $I_3 = 1.9989$ A

12.30 This problem can be solved directly on a calculator capable of doing matrix operations or on MATLAB.

```

>> a=[60 -40 0
      -40 150 -100
      0 -100 130];
>> b=[200
      0
      230];
>> x=a\b

x =
7.7901
6.6851
6.9116

```

Therefore, $I_1 = 7.79$ A, $I_2 = 6.69$ A, and $I_3 = 6.91$ A.

12.31 At steady state, the force balances can be written as

$$\begin{aligned} 4kx_1 - 3kx_2 &= m_1 g \\ -3kx_1 + 4kx_2 - kx_3 &= m_2 g \\ -kx_2 + kx_3 &= m_3 g \end{aligned}$$

Substituting the parameter values

$$\begin{bmatrix} 120 & -90 & 0 \\ -90 & 120 & -30 \\ 0 & -30 & 30 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 19.6 \\ 29.4 \\ 24.5 \end{Bmatrix}$$

The solution is $x_1 = 2.45$, $x_2 = 3.049$, and $x_3 = 3.866$.

12.32 At steady state, the force balances can be written as

$$\begin{bmatrix} 30 & -20 & 0 \\ -20 & 30 & -10 \\ 0 & -10 & -10 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 98 \\ 34.3 \\ 19.6 \end{Bmatrix}$$

The solution is $x_1 = 15.19$, $x_2 = 17.885$, and $x_3 = 19.845$.

12.33 The force balances can be written as

$$\begin{bmatrix} k_1 + k_2 & -k_2 & 0 & 0 \\ -k_2 & k_2 + k_3 & -k_3 & 0 \\ 0 & -k_3 & k_3 + k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ F \end{Bmatrix}$$

Substituting the parameter values

$$\begin{bmatrix} 150 & -50 & 0 & 0 \\ -50 & 130 & -80 & 0 \\ 0 & -80 & 280 & -200 \\ 0 & 0 & -200 & 200 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1500(9.8) \end{Bmatrix}$$

The solution is $x_1 = 147$, $x_2 = 441$, $x_3 = 624.75$, and $x_4 = 698.25$.

12.34 The equations can be solved in a number of ways. For example, using MATLAB,

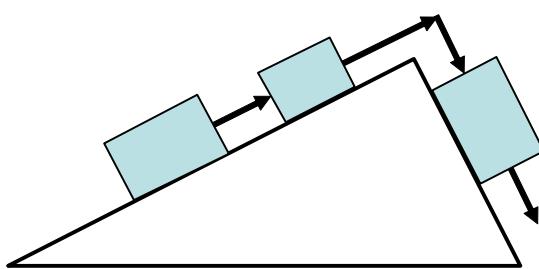
```
>> A=[100 1 0;50 -1 1;25 0 -1];
>> B=[519.72;216.55;108.27];
>> x=A\B

x =
    4.8259
    37.1257
    12.3786
```

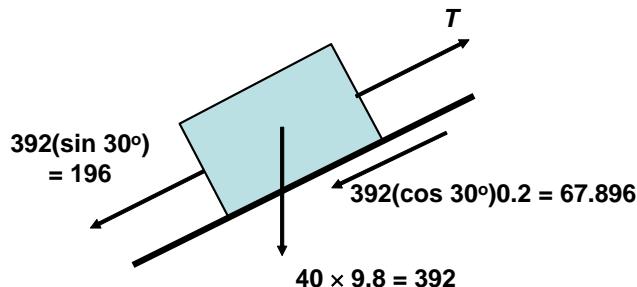
Therefore, $a = 4.8259$, $T = 37.1257$, and $R = 12.3786$.

12.35 In order to solve this problem, we must assume the direction that the blocks are moving.

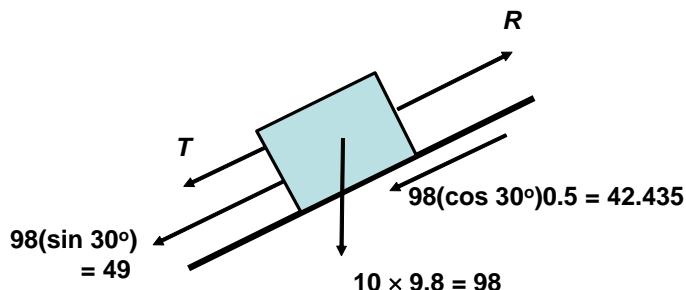
For example, we can assume that the blocks are moving from left to right as shown



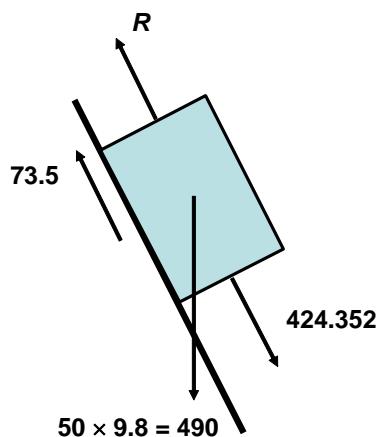
Force balances can be written for each block:



$$-196 - 67.896 + T = 40a$$



$$-49 - T + R - 42.435 = 10a$$



$$424.352 - 73.5 - R = 50a$$

Therefore, the system of equations to be solved can be written in matrix form as

$$\begin{bmatrix} 40 & -1 & 0 \\ 10 & 1 & -1 \\ 50 & 0 & 1 \end{bmatrix} \begin{Bmatrix} a \\ T \\ R \end{Bmatrix} = \begin{Bmatrix} -263.8964 \\ -91.43524 \\ 350.852 \end{Bmatrix}$$

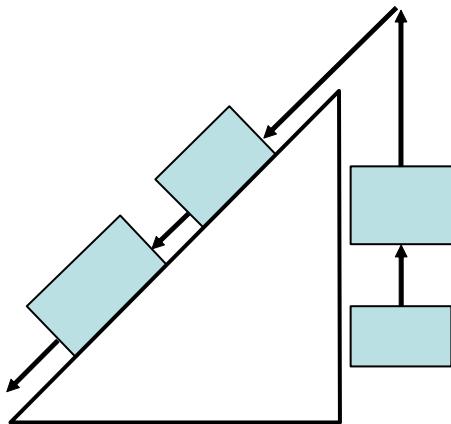
The solution is $a = -0.044792$, $T = 262.1047$, and $R = 353.092$.

Note that if we had assumed that the blocks were moving from right to left, the system of equations would have been

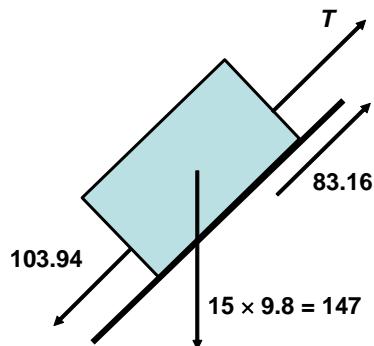
$$\begin{bmatrix} 40 & 1 & 0 \\ 10 & -1 & 1 \\ 50 & 0 & -1 \end{bmatrix} \begin{Bmatrix} a \\ T \\ R \end{Bmatrix} = \begin{Bmatrix} 128.1036 \\ 6.564755 \\ -497.852 \end{Bmatrix}$$

The solution for this case is $a = -3.63184$, $T = 273.3772$, and $R = 316.2604$.

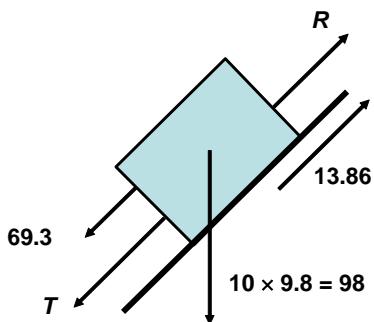
- 12.36** In order to solve this problem, we must assume the direction that the blocks are moving. For example, we can assume that the blocks are moving from right to left as shown



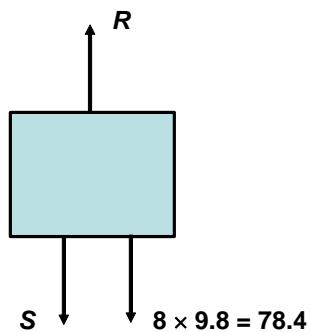
Force balances can be written for each block:



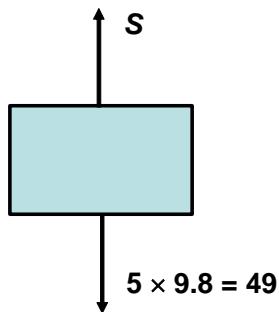
$$103.94 - T - 83.16 = 15a$$



$$T + 69.3 - R - 13.86 = 10a$$



$$R - 78.4 - S = 8a$$



$$S - 49 = 5a$$

Therefore, the system of equations to be solved can be written in matrix form as

$$\begin{bmatrix} 15 & 1 & 0 & 0 \\ 10 & -1 & 1 & 0 \\ 8 & 0 & -1 & 1 \\ 5 & 0 & 0 & -1 \end{bmatrix} \begin{Bmatrix} a \\ T \\ R \\ S \end{Bmatrix} = \begin{Bmatrix} 20.789 \\ 55.437 \\ -78.4 \\ -49 \end{Bmatrix}$$

The solution is $a = -1.347$, $T = 40.989$, $R = 109.893$, and $S = 42.267$.

Note that if we had assumed that the blocks were moving from left to right, the system of equations would have been

$$\begin{bmatrix} 15 & -1 & 0 & 0 \\ 10 & 1 & -1 & 0 \\ 8 & 0 & 1 & -1 \\ 5 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} a \\ T \\ S \\ R \end{Bmatrix} = \begin{Bmatrix} -187.1005 \\ -83.15576 \\ 78.4 \\ 49 \end{Bmatrix}$$

The solution for this case is $a = -3.759374$, $T = 130.7098$, $R = 176.27186$, and $S = 67.79687$.

12.37 This problem can be solved in a number of ways. For example, using MATLAB,

```
%prob1237.m

k1=10;k2=30;k3=30;k4=10;
m1=2;m2=2;m3=2;
km=[(1/m1)*(k2+k1),-(k2/m1),0;
     -(k2/m2),(1/m2)*(k2+k3),-(k3/m2);
     0,-(k3/m3),(1/m3)*(k3+k4)];
x=[0.05;0.04;0.03]
kmx=km*x

>> prob1237

km =
    20    -15      0
   -15     30    -15
     0    -15     20

x =
    0.0500
    0.0400
    0.0300

kmx =
    0.4000
        0
        0
        0
```

Therefore, $\ddot{x}_1 = -0.4$, $\ddot{x}_2 = 0$, and $\ddot{x}_3 = 0 \text{ m/s}^2$.

12.38 (a) Substituting the parameters gives

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

An analytical solution can be derived in a number of ways. One way is to assume a solution of the form

$$T = A + Be^{\lambda x} + Ce^{\lambda x}$$

Differentiating twice gives

$$T'' = \lambda^2 Be^{\lambda x} + \lambda^2 Ce^{\lambda x}$$

Substituting these into the original differential equation yields

$$\lambda^2 Be^{\lambda x} + \lambda^2 Ce^{\lambda x} + h'(T_a - A - Be^{\lambda x} - Ce^{\lambda x}) = 0$$

Equating like terms yields

$$\lambda^2 Be^{\lambda x} = h' Be^{\lambda T}$$

$$\lambda^2 Ce^{\lambda x} = h' Ce^{\lambda T}$$

$$h'T_a = h'A$$

The first two equations give $\lambda = \pm\sqrt{h'}$. The third equation gives $A = T_a$. Therefore, the solution is

$$T = T_a + Be^{\sqrt{h'}x} + Ce^{-\sqrt{h'}x}$$

The unknown constants can be evaluated from the boundary conditions

$$40 = 20 + B + C$$

$$200 = 20 + Be^{\sqrt{0.02}(10)} + Ce^{-\sqrt{0.02}(10)}$$

These simultaneous equations can be solved for $B = 45.25365$ and $C = -25.25365$. Therefore, the analytical solution is

$$T = 20 + 45.25365 e^{0.141421x} - 25.25365 e^{-0.141421x}$$

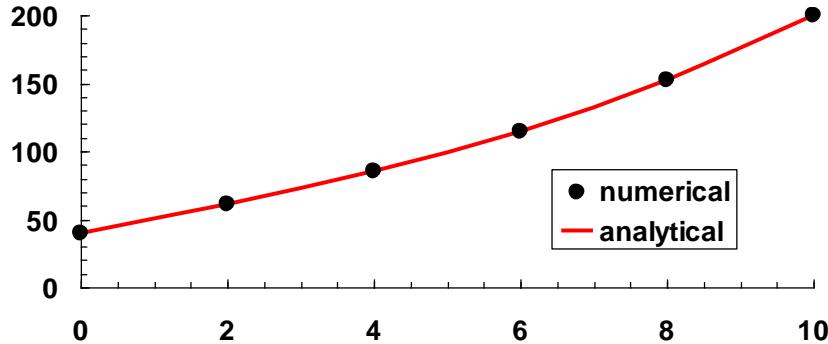
(b) Substituting the parameters into the finite-difference equation gives

$$-T_{i-1} + 2.08T_i - T_{i+1} = 1.6$$

An analytical solution can be derived in a number of ways. A nice approach is to employ L. This equation can be written for each node to yield the following system of equations,

$$\begin{bmatrix} 2.08 & -1 & 0 & 0 \\ -1 & 2.08 & -1 & 0 \\ 0 & -1 & 2.08 & -1 \\ 0 & 0 & -1 & 2.08 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 41.6 \\ 1.6 \\ 1.6 \\ 201.6 \end{Bmatrix}$$

These can be solved for $T_1 = 61.0739$, $T_2 = 85.4338$, $T_3 = 115.0283$, and $T_4 = 152.2252$. A plot of the results is shown below (circles). In addition, the plot also shows the analytical solution (line) that was developed in (a):



12.39 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: $T_{11} = 68.64$, $T_{12} = 56.195$, $T_{21} = 43.695$, and $T_{22} = 31.22$.

12.40 Find the unit vectors:

$$A\left(\frac{1\hat{i} - 2\hat{j} - 4\hat{k}}{\sqrt{1^2 + 2^2 + 4^2}}\right) = 0.218\hat{i} - 0.436\hat{j} - 0.873\hat{k}$$

$$B\left(\frac{2\hat{i} + 1\hat{j} - 4\hat{k}}{\sqrt{1^2 + 2^2 + 4^2}}\right) = 0.436\hat{i} + 0.218\hat{j} - 0.873\hat{k}$$

Sum moments about the origin:

$$\sum M_{ox} = 50(2) - 0.436B(4) - 0.218A(4) = 0$$

$$\sum M_{oy} = 0.436A(4) - 0.218B(4) = 0$$

Solve for A and B using equations 9.10 and 9.11:

$$\text{In the form } \begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$$

$$\begin{aligned} -0.872A - 1.744B &= -100 \\ 1.744A - 0.872B &= 0 \end{aligned}$$

Plug into equations 9.10 and 9.11:

$$A = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}} = \frac{87.2}{3.80192} = 22.94 \text{ N}$$

$$B = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}a_{21}} = \frac{174.4}{3.80192} = 45.87 \text{ N}$$

CHAPTER 13

13.1 (a) The function can be differentiated to give

$$f'(x) = -2x + 8$$

This function can be set equal to zero and solved for $x = 8/2 = 4$. The derivative can be differentiated to give the second derivative

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

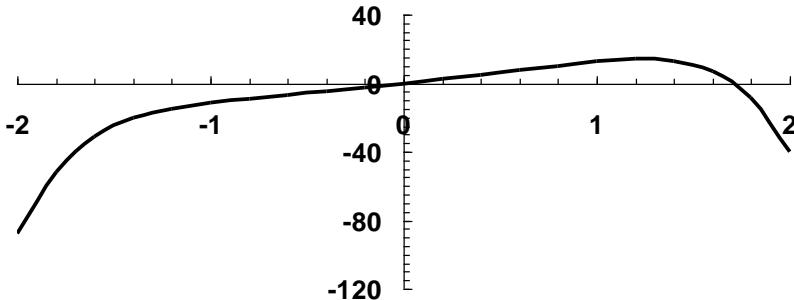
Because this is negative, it indicates that the function has a maximum at $x = 4$.

(b) Using Eq. 13.7

$$\begin{array}{ll} x_0 = 0 & f(x_0) = -12 \\ x_1 = 2 & f(x_1) = 0 \\ x_2 = 6 & f(x_2) = 0 \end{array}$$

$$x_3 = \frac{-12(4 - 36) + 0(36 - 0) + 0(0 - 4)}{2(-12)(2 - 6) + 2(0)(6 - 0) + 2(0)(0 - 2)} = 4$$

13.2 (a) The function can be plotted



(b) The function can be differentiated twice to give

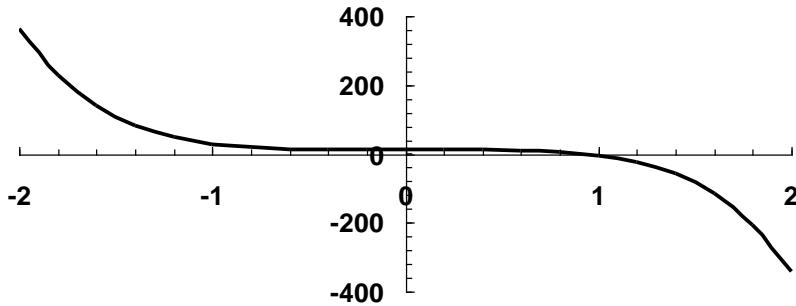
$$f''(x) = -45x^4 - 24x^2$$

Thus, the second derivative will always be negative and hence the function is concave for all values of x .

(c) Differentiating the function and setting the result equal to zero results in the following roots problem to locate the maximum

$$f'(x) = 0 = -9x^5 - 8x^3 + 12$$

A plot of this function can be developed



A technique such as bisection can be employed to determine the root. Here are the first few iterations:

iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_u)$	$f(x_l) \times f(x_r)$	ε_a
1	0.00000	2.00000	1.00000	12	-5	-60.0000	
2	0.00000	1.00000	0.50000	12	10.71875	128.6250	100.00%
3	0.50000	1.00000	0.75000	10.71875	6.489258	69.5567	33.33%
4	0.75000	1.00000	0.87500	6.489258	2.024445	13.1371	14.29%
5	0.87500	1.00000	0.93750	2.024445	-1.10956	-2.2463	6.67%

The approach can be continued to yield a result of $x = 0.91692$.

13.3 First, the golden ratio can be used to create the interior points,

$$d = \frac{\sqrt{5} - 1}{2}(2 - 0) = 1.2361$$

$$x_1 = 0 + 1.2361 = 1.2361$$

$$x_2 = 2 - 1.2361 = 0.7639$$

The function can be evaluated at the interior points

$$f(x_2) = f(0.7639) = 8.1879$$

$$f(x_1) = f(1.2361) = 4.8142$$

Because $f(x_2) > f(x_1)$, the maximum is in the interval defined by x_l , x_2 , and x_1 . where x_2 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{2 - 0}{0.7639} \right| \times 100\% = 100\%$$

For the second iteration, $x_l = 0$ and $x_u = 1.2361$. The former x_2 value becomes the new x_1 , that is, $x_1 = 0.7639$ and $f(x_1) = 8.1879$. The new values of d and x_2 can be computed as

$$d = \frac{\sqrt{5} - 1}{2}(1.2361 - 0) = 0.7639$$

$$x_2 = 1.2361 - 0.7639 = 0.4721$$

The function evaluation at $f(x_2) = 5.5496$. Since this value is less than the function value at x_1 , the maximum is in the interval prescribed by x_2 , x_1 and x_u . The process can be repeated and all three iterations summarized as

<i>i</i>	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	<i>d</i>	x_{opt}	ϵ_a
1	0.0000	0.0000	0.7639	8.1879	1.2361	4.8142	2.0000	-104.0000	1.2361	0.7639	100.00%
2	0.0000	0.0000	0.4721	5.5496	0.7639	8.1879	1.2361	4.8142	0.7639	0.7639	61.80%
3	0.4721	5.5496	0.7639	8.1879	0.9443	8.6778	1.2361	4.8142	0.4721	0.9443	30.90%

13.4 First, the function values at the initial values can be evaluated

$$f(x_0) = f(0) = 0$$

$$f(x_1) = f(1) = 8.5$$

$$f(x_2) = f(2) = -104$$

and substituted into Eq. (13.7) to give,

$$x_3 = \frac{0(1^2 - 2^2) + 8.5(2^2 - 0^2) + (-104)(0^2 - 1^2)}{2(0)(1-2) + 2(8.5)(2-0) + 2(-104)(0-1)} = 0.570248$$

which has a function value of $f(0.570248) = 6.5799$. Because the function value for the new point is lower than for the intermediate point (x_1) and the new x value is to the left of the intermediate point, the lower guess (x_0) is discarded. Therefore, for the next iteration,

$$f(x_0) = f(0.570248) = 6.6799$$

$$f(x_1) = f(1) = 8.5$$

$$f(x_2) = f(2) = -104$$

which can be substituted into Eq. (13.7) to give $x_3 = 0.812431$, which has a function value of $f(0.812431) = 8.446523$. At this point, an approximate error can be computed as

$$\epsilon_a = \left| \frac{0.81243 - 0.570248}{0.81243} \right| \times 100\% = 29.81\%$$

The process can be repeated, with the results tabulated below:

<i>i</i>	x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$	ϵ_a
1	0.00000	0.00000	1.00000	8.50000	2.0000	-104	0.57025	6.57991	
2	0.57025	6.57991	1.00000	8.50000	2.0000	-104	0.81243	8.44652	29.81%
3	0.81243	8.44652	1.00000	8.50000	2.0000	-104	0.90772	8.69575	10.50%

Thus, after 3 iterations, the result is converging on the true value of $f(x) = 8.69793$ at $x = 0.91692$.

13.5 The first and second derivatives of the function can be evaluated as

$$f'(x) = -9x^5 - 8x^3 + 12$$

$$f''(x) = -45x^4 - 24x^2$$

which can be substituted into Eq. (13.8) to give

$$x_{i+1} = x_i - \frac{-9x_i^5 - 8x_i^3 + 12}{-45x_i^4 - 24x_i^2}$$

Substituting the initial guess yields

$$x_{i+1} = 2 - \frac{-9(2^5) - 8(2^3) + 12}{-45(2^4) - 24(2^2)} = 2 - \frac{-340}{-816} = 1.583333$$

which has a function value of -17.2029. The second iteration gives

$$x_{i+1} = 1.583333 - \frac{-9(1.583333^5) - 8(1.583333^3) + 12}{-45(1.583333^4) - 24(1.583333^2)} = 1.583333 - \frac{-109.313}{-342.981} = 1.26462$$

which has a function value of 3.924617. At this point, an approximate error can be computed as

$$\varepsilon_a = \left| \frac{1.26462 - 1.583333}{1.26462} \right| \times 100\% = 26.316\%$$

The process can be repeated, with the results tabulated below:

i	x	f(x)	f'(x)	f''(x)	ea
0	2	-104	-340	-816	
1	1.583333	-17.2029	-109.313	-342.981	26.316%
2	1.26462	3.924617	-33.2898	-153.476	25.202%
3	1.047716	8.178616	-8.56281	-80.5683	20.703%

Thus, within five iterations, the result is converging on the true value of $f(x) = 8.69793$ at $x = 0.91692$.

13.6 Golden section search is inefficient, but always converges if x_l and x_u bracket the maximum or minimum of a unimodal function.

Quadratic interpolation can be programmed as either a bracketing or as an open method. For the former, convergence is guaranteed if the initial guesses bracket the maximum or minimum of a unimodal function. However, as mentioned at the top of p. 351, it may

sometimes converge slowly. If it is programmed as an open method, it may converge rapidly for well-behaved functions and good initial values. Otherwise, it may diverge. It also has the disadvantage that three initial guesses are required.

Newton's method may converge rapidly for well-behaved functions and good initial values. Otherwise, it may diverge. It also has the disadvantage that both the first and second derivatives must be determined.

13.7 (a) First, the golden ratio can be used to create the interior points,

$$d = \frac{\sqrt{5} - 1}{2} (4 - (-2)) = 3.7082$$

$$x_1 = -2 + 3.7082 = 1.7082$$

$$x_2 = 4 - 3.7082 = 0.2918$$

The function can be evaluated at the interior points

$$f(x_2) = f(0.2918) = 1.04156$$

$$f(x_1) = f(1.7082) = 5.00750$$

Because $f(x_1) > f(x_2)$, the maximum is in the interval defined by x_2 , x_1 and x_u where x_1 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{4 - (-2)}{1.7082} \right| \times 100\% = 134.16\%$$

The process can be repeated and all the iterations summarized as

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d	x_{opt}	ε_a
1	-2.0000	-29.6000	0.2918	1.0416	1.7082	5.0075	4.0000	-12.8000	3.7082	1.7082	134.16%
2	0.2918	1.0416	1.7082	5.0075	2.5836	5.6474	4.0000	-12.8000	2.2918	2.5836	54.82%
3	1.7082	5.0075	2.5836	5.6474	3.1246	2.9361	4.0000	-12.8000	1.4164	2.5836	33.88%
4	1.7082	5.0075	2.2492	5.8672	2.5836	5.6474	3.1246	2.9361	0.8754	2.2492	24.05%
5	1.7082	5.0075	2.0426	5.6648	2.2492	5.8672	2.5836	5.6474	0.5410	2.2492	14.87%
6	2.0426	5.6648	2.2492	5.8672	2.3769	5.8770	2.5836	5.6474	0.3344	2.3769	8.69%
7	2.2492	5.8672	2.3769	5.8770	2.4559	5.8287	2.5836	5.6474	0.2067	2.3769	5.37%
8	2.2492	5.8672	2.3282	5.8853	2.3769	5.8770	2.4559	5.8287	0.1277	2.3282	3.39%
9	2.2492	5.8672	2.2980	5.8828	2.3282	5.8853	2.3769	5.8770	0.0789	2.3282	2.10%
10	2.2980	5.8828	2.3282	5.8853	2.3468	5.8840	2.3769	5.8770	0.0488	2.3282	1.30%
11	2.2980	5.8828	2.3166	5.8850	2.3282	5.8853	2.3468	5.8840	0.0301	2.3282	0.80%

(b) First, the function values at the initial values can be evaluated

$$f(x_0) = f(1.75) = 5.1051$$

$$f(x_1) = f(2) = 5.6$$

$$f(x_2) = f(2.5) = 5.7813$$

and substituted into Eq. (13.7) to give,

$$x_3 = \frac{5.1051(2^2 - 2.5^2) + 5.6(2.5^2 - 1.75^2) + 5.7813(1.75^2 - 2^2)}{2(5.1051)(2 - 2.5) + 2(5.6)(2.5 - 1.75) + 2(5.7813)(1.75 - 2)} = 2.3341$$

which has a function value of $f(2.3341) = 5.8852$. Because the function value for the new point is higher than for the intermediate point (x_1) and the new x value is to the right of the intermediate point, the lower guess (x_0) is discarded. Therefore, for the next iteration,

$$\begin{aligned} f(x_0) &= f(2) = 5.6 \\ f(x_1) &= f(2.3341) = 5.8852 \\ f(x_2) &= f(2.5) = 5.7813 \end{aligned}$$

which can be substituted into Eq. (13.7) to give $x_3 = 2.3112$, which has a function value of $f(2.3112) = 5.8846$. At this point, an approximate error can be computed as

$$\varepsilon_a = \left| \frac{2.3112 - 2.3341}{2.3112} \right| \times 100\% = 0.99\%$$

The process can be repeated, with the results tabulated below:

<i>i</i>	x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$	ε_a
1	1.7500	5.1051	2.0000	5.6000	2.5000	5.7813	2.3341	5.8852	
2	2.0000	5.6000	2.3341	5.8852	2.5000	5.7813	2.3112	5.8846	0.99%
3	2.3112	5.8846	2.3341	5.8852	2.5000	5.7813	2.3260	5.8853	0.64%
4	2.3112	5.8846	2.3260	5.8853	2.3341	5.8852	2.3263	5.8853	0.01%

Thus, after 4 iterations, the result is converging rapidly on the true value of $f(x) = 5.8853$ at $x = 2.3263$.

(c) The first and second derivatives of the function can be evaluated as

$$\begin{aligned} f'(x) &= 4 - 3.6x + 3.6x^2 - 1.2x^3 \\ f''(x) &= -3.6 + 7.2x - 3.6x^2 \end{aligned}$$

which can be substituted into Eq. (13.8) to give

$$x_{i+1} = x_i - \frac{4 - 3.6x_i + 3.6x_i^2 - 1.2x_i^3}{-3.6 + 7.2x_i - 3.6x_i^2} = 3 - \frac{-6.8}{-14.4} = 2.5278$$

which has a function value of 5.7434. The second iteration gives 2.3517, which has a function value of 5.8833. At this point, an approximate error can be computed as $\varepsilon_a = 18.681\%$. The process can be repeated, with the results tabulated below:

<i>i</i>	<i>x</i>	<i>f(x)</i>	<i>f'(x)</i>	<i>f''(x)</i>	ε_a
0	3.0000	3.9000	-6.8000	-14.4000	
1	2.5278	5.7434	-1.4792	-8.4028	18.681%
2	2.3517	5.8833	-0.1639	-6.5779	7.485%
3	2.3268	5.8853	-0.0030	-6.3377	1.071%
4	2.3264	5.8853	0.0000	-6.3332	0.020%

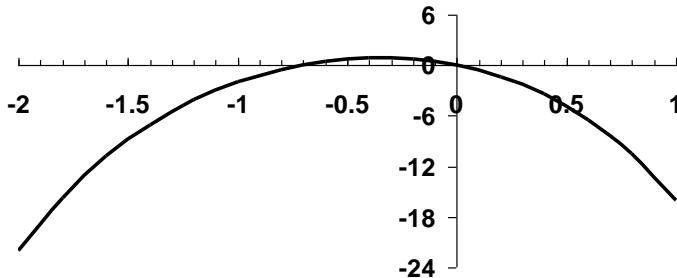
Thus, within four iterations, the result is converging on the true value of $f(x) = 5.8853$ at $x = 2.3264$.

13.8 The function can be differentiated twice to give

$$f'(x) = -4x^3 - 6x^2 - 16x - 5$$

$$f''(x) = -12x^2 - 12x - 16$$

which is negative for $-2 \leq x \leq 1$. This suggests that an optimum in the interval would be a maximum. A graph of the original function shows a maximum at about $x = -0.35$.



13.9 (a) First, the golden ratio can be used to create the interior points,

$$d = \frac{\sqrt{5} - 1}{2}(1 - (-2)) = 1.8541$$

$$x_1 = -2 + 1.8541 = -0.1459$$

$$x_2 = 1 - 1.8541 = -0.8541$$

The function can be evaluated at the interior points

$$f(x_2) = f(-0.8541) = -0.8514$$

$$f(x_1) = f(-0.1459) = 0.5650$$

Because $f(x_1) > f(x_2)$, the maximum is in the interval defined by x_2 , x_1 and x_u where x_1 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{1 - (-2)}{-0.1459} \right| \times 100\% = 785.41\%$$

The process can be repeated and all the iterations summarized as

<i>i</i>	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	<i>d</i>	x_{opt}	ε_a
1	-2	-22	-0.8541	-0.851	-0.1459	0.565	1	-16.000	1.8541	-0.1459	785.41%
2	-0.8541	-0.851	-0.1459	0.565	0.2918	-2.197	1	-16.000	1.1459	-0.1459	485.41%
3	-0.8541	-0.851	-0.4164	0.809	-0.1459	0.565	0.2918	-2.197	0.7082	-0.4164	105.11%
4	-0.8541	-0.851	-0.5836	0.475	-0.4164	0.809	-0.1459	0.565	0.4377	-0.4164	64.96%
5	-0.5836	0.475	-0.4164	0.809	-0.3131	0.833	-0.1459	0.565	0.2705	-0.3131	53.40%
6	-0.4164	0.809	-0.3131	0.833	-0.2492	0.776	-0.1459	0.565	0.1672	-0.3131	33.00%
7	-0.4164	0.809	-0.3525	0.841	-0.3131	0.833	-0.2492	0.776	0.1033	-0.3525	18.11%
8	-0.4164	0.809	-0.3769	0.835	-0.3525	0.841	-0.3131	0.833	0.0639	-0.3525	11.19%
9	-0.3769	0.835	-0.3525	0.841	-0.3375	0.840	-0.3131	0.833	0.0395	-0.3525	6.92%
10	-0.3769	0.835	-0.3619	0.839	-0.3525	0.841	-0.3375	0.840	0.0244	-0.3525	4.28%
11	-0.3619	0.839	-0.3525	0.841	-0.3468	0.841	-0.3375	0.840	0.0151	-0.3468	2.69%
12	-0.3525	0.841	-0.3468	0.841	-0.3432	0.841	-0.3375	0.840	0.0093	-0.3468	1.66%
13	-0.3525	0.841	-0.3490	0.841	-0.3468	0.841	-0.3432	0.841	0.0058	-0.3468	1.03%
14	-0.3490	0.841	-0.3468	0.841	-0.3454	0.841	-0.3432	0.841	0.0036	-0.3468	0.63%

(b) First, the function values at the initial values can be evaluated

$$f(x_0) = f(-2) = -22$$

$$f(x_1) = f(-1) = -2$$

$$f(x_2) = f(1) = -16$$

and substituted into Eq. (13.7) to give,

$$x_3 = \frac{-22((-1)^2 - 1^2) + (-2)(1^2 - (-2)^2) + (-16)((-2)^2 - (-1)^2)}{2(-22)(-1 - 1) + 2(-2)(1 - (-2)) + 2(-16)(-2 - (-1))} = -0.38889$$

which has a function value of $f(-0.38889) = 0.829323$. Because the function value for the new point is higher than for the intermediate point (x_1) and the new x value is to the right of the intermediate point, the lower guess (x_0) is discarded. Therefore, for the next iteration,

$$f(x_0) = f(-1) = -2$$

$$f(x_1) = f(-0.38889) = 0.829323$$

$$f(x_2) = f(1) = -16$$

which can be substituted into Eq. (13.7) to give $x_3 = -0.41799$, which has a function value of $f(-0.41799) = 0.80776$. At this point, an approximate error can be computed as

$$\varepsilon_a = \left| \frac{-0.41799 - (-0.38889)}{-0.41799} \right| \times 100\% = 6.96\%$$

The process can be repeated, with the results tabulated below:

<i>i</i>	<i>x₀</i>	<i>f(x₀)</i>	<i>x₁</i>	<i>f(x₁)</i>	<i>x₂</i>	<i>f(x₂)</i>	<i>x₃</i>	<i>f(x₃)</i>	<i>ε_a</i>
1	-2	-22	-1.0000	-2	1.0000	-16	-0.3889	0.8293	
2	-1.0000	-2	-0.3889	0.8293	1.0000	-16	-0.4180	0.8078	6.96%
3	-0.4180	0.8078	-0.3889	0.8293	1.0000	-16	-0.3626	0.8392	15.28%
4	-0.3889	0.8293	-0.3626	0.8392	1.0000	-16	-0.3552	0.8404	2.08%

After 4 iterations, the result is converging on the true value of $f(x) = 0.8408$ at $x = -0.34725$.

(c) The first and second derivatives of the function can be evaluated as

$$f'(x) = -4x^3 - 6x^2 - 16x - 5$$

$$f''(x) = -12x^2 - 12x - 16$$

which can be substituted into Eq. (13.8) to give

$$x_{i+1} = x_i - \frac{-4x_i^3 - 6x_i^2 - 16x_i - 5}{-12x_i^2 - 12x_i - 16} = -1 - \frac{9}{-16} = -0.4375$$

which has a function value of 0.787094. The second iteration gives -0.34656, which has a function value of 0.840791. At this point, an approximate error can be computed as $\epsilon_a = 128.571\%$. The process can be repeated, with the results tabulated below:

<i>i</i>	<i>x</i>	<i>f(x)</i>	<i>f'(x)</i>	<i>f''(x)</i>	<i>ε_a</i>
0	-1	-2	9	-16	
1	-0.4375	0.787094	1.186523	-13.0469	128.571%
2	-0.34656	0.840791	-0.00921	-13.2825	26.242%
3	-0.34725	0.840794	-8.8E-07	-13.28	0.200%

Thus, within three iterations, the result is converging on the true value of $f(x) = 0.840794$ at $x = -0.34725$.

13.10 First, the function values at the initial values can be evaluated

$$f(x_0) = f(0.1) = 30.2$$

$$f(x_1) = f(0.5) = 7$$

$$f(x_2) = f(5) = 10.6$$

and substituted into Eq. (13.7) to give,

$$x_3 = \frac{30.2(0.5^2 - 5^2) + 7(5^2 - 0.1^2) + 10.6(0.1^2 - 0.5^2)}{2(30.2)(0.5 - 5) + 2(7)(5 - 0.1) + 2(10.6)(0.1 - 0.5)} = 2.7167$$

which has a function value of $f(2.7167) = 6.5376$. Because the function value for the new point is lower than for the intermediate point (x_1) and the new x value is to the right of the intermediate point, the lower guess (x_0) is discarded. Therefore, for the next iteration,

$$f(x_0) = f(0.5) = 7$$

$$f(x_1) = f(2.7167) = 6.5376$$

$$f(x_2) = f(5) = 10.6$$

which can be substituted into Eq. (13.7) to give $x_3 = 1.8444$, which has a function value of $f(1.8444) = 5.3154$. At this point, an approximate error can be computed as

$$\varepsilon_a = \left| \frac{1.8444 - 2.7167}{1.8444} \right| \times 100\% = 47.29\%$$

The process can be repeated, with the results tabulated below:

<i>i</i>	x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$	ε_a
1	0.1	30.2	0.5000	7.0000	5.0000	10.6000	2.7167	6.5376	
2	0.5000	7	2.7167	6.5376	5.0000	10.6000	1.8444	5.3154	47.29%
3	0.5000	7	1.8444	5.3154	2.7167	6.5376	1.6954	5.1603	8.79%
4	0.5000	7	1.6954	5.1603	1.8444	5.3154	1.4987	4.9992	13.12%
5	0.5000	7	1.4987	4.9992	1.6954	5.1603	1.4236	4.9545	5.28%
6	0.5000	7	1.4236	4.9545	1.4987	4.9992	1.3556	4.9242	5.02%
7	0.5000	7	1.3556	4.9242	1.4236	4.9545	1.3179	4.9122	2.85%
8	0.5000	7	1.3179	4.9122	1.3556	4.9242	1.2890	4.9054	2.25%
9	0.5000	7	1.2890	4.9054	1.3179	4.9122	1.2703	4.9023	1.47%
10	0.5000	7	1.2703	4.9023	1.2890	4.9054	1.2568	4.9006	1.08%

Thus, after 10 iterations, the result is converging very slowly on the true value of $f(x) = 4.8990$ at $x = 1.2247$.

13.11 Differentiating the function and setting the result equal to zero results in the following roots problem to locate the minimum

$$f'(x) = 6 + 10x + 9x^2 + 16x^3$$

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

iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l) \times f(x_r)$	ε_a
1	-2.0000	1.0000	-0.5000	-106.000	1.2500	-132.500	300.00%
2	-2.0000	-0.5000	-1.2500	-106.000	-23.6875	2510.875	60.00%
3	-1.2500	-0.5000	-0.8750	-23.6875	-6.5781	155.819	42.86%
4	-0.8750	-0.5000	-0.6875	-6.5781	-1.8203	11.974	27.27%
5	-0.6875	-0.5000	-0.5938	-1.8203	-0.1138	0.207	15.79%
6	-0.5938	-0.5000	-0.5469	-0.1138	0.6060	-0.0689	8.57%
7	-0.5938	-0.5469	-0.5703	-0.1138	0.2562	-0.0291	4.11%
8	-0.5938	-0.5703	-0.5820	-0.1138	0.0738	-0.0084	2.01%
9	-0.5938	-0.5820	-0.5879	-0.1138	-0.0193	0.0022	1.00%
10	-0.5879	-0.5820	-0.5850	-0.0193	0.0274	-0.0005	0.50%

The approach can be continued to yield a result of $x = -0.5867$.

13.12 (a) The first and second derivatives of the function can be evaluated as

$$f'(x) = 6 + 10x + 9x^2 + 16x^3$$

$$f''(x) = 10 + 18x + 48x^2$$

which can be substituted into Eq. (13.8) to give

$$x_{i+1} = x_i - \frac{6 + 10x_i + 9x_i^2 + 16x_i^3}{10 + 18x_i + 48x_i^2} = -1 - \frac{-11}{40} = -0.725$$

which has a function value of 1.24. The second iteration gives -0.60703, which has a function value of 1.07233. At this point, an approximate error can be computed as $\varepsilon_a = 37.931\%$. The process can be repeated, with the results tabulated below:

i	x	f(x)	f'(x)	f''(x)	ε_a
0	-1	3	-11	40	
1	-0.725	1.24002	-2.61663	22.180	37.931%
2	-0.60703	1.07233	-0.33280	16.76067	19.434%
3	-0.58717	1.06897	-0.00781	15.97990	3.382%
4	-0.58668	1.06897	-4.6E-06	15.96115	0.083%

Thus, within four iterations, the stopping criterion is met and the result is converging on the true value of $f(x) = 1.06897$ at $x = -0.58668$.

(b) The finite difference approximations of the derivatives can be computed as

$$f'(x) = \frac{3.1120 - 2.8920}{-0.01} = -11.001$$

$$f''(x) = \frac{3.1120 - 2(3) + 2.8920}{(-0.01)^2} = 40.001$$

which can be substituted into Eq. (13.8) to give

$$x_{i+1} = x_i - \frac{6 + 10x_i + 9x_i^2 + 16x_i^3}{10 + 18x_i + 48x_i^2} = -1 - \frac{-11.001}{40.001} = -0.725$$

which has a function value of 1.2399. The second iteration gives -0.6070, which has a function value of 1.0723. At this point, an approximate error can be computed as $\varepsilon_a = 37.936\%$. The process can be repeated, with the results tabulated below:

i	x_i	$f(x_i)$	δx_i	$x_i - \delta x_i$	$f(x_i - \delta x_i)$	$x_i + \delta x_i$	$f(x_i + \delta x_i)$	$f'(x_i)$	$f''(x_i)$	ε_a
0	-1	3	-0.01	-0.99	2.8920	-1.0100	3.1120	-11.001	40.001	
1	-0.7250	1.2399	-0.00725	-0.7177	1.2216	-0.7322	1.2595	-2.616	22.179	37.94%
2	-0.6070	1.0723	-0.00607	-0.6009	1.0706	-0.6131	1.0746	-0.333	16.760	19.43%

3	-0.5872	1.0690	-0.00587	-0.5813	1.0692	-0.5930	1.0693	-0.008	15.980	3.38%
4	-0.5867	1.0690	-0.00587	-0.5808	1.0692	-0.5925	1.0692	-4.1E-06	15.961	0.081%

Thus, within four iterations, the stopping criterion is met and the result is converging on the true value of $f(x) = 1.06897$ at $x = -0.58668$.

13.13 Because of multiple local minima and maxima, there is no really simple means to test whether a single maximum occurs within an interval without actually performing a search. However, if we assume that the function has one maximum and no minima within the interval, a check can be included. Here is a VBA program to implement the Golden section search algorithm for maximization and solve Example 13.1.

```
Option Explicit

Sub GoldMax()
    Dim ier As Integer
    Dim xlow As Double, xhigh As Double
    Dim xoxt As Double, foxt As Double
    xlow = 0
    xhigh = 4
    Call GoldMx(xlow, xhigh, xoxt, foxt, ier)
    If ier = 0 Then
        MsgBox "xoxt = " & xoxt
        MsgBox "f(xoxt) = " & foxt
    Else
        MsgBox "Does not appear to be maximum in [xl, xu]"
    End If
End Sub

Sub GoldMx(xlow, xhigh, xoxt, foxt, ier)
    Dim iter As Integer, maxit As Integer, ea As Double, es As Double
    Dim xl As Double, xu As Double, d As Double, x1 As Double
    Dim x2 As Double, f1 As Double, f2 As Double
    Const R As Double = (5 ^ 0.5 - 1) / 2
    ier = 0
    maxit = 50
    es = 0.001
    xl = xlow
    xu = xhigh
    iter = 1
    d = R * (xu - xl)
    x1 = xl + d
    x2 = xu - d
    f1 = f(x1)
    f2 = f(x2)
    If f1 > f2 Then
        xoxt = x1
        foxt = f1
    Else
        xoxt = x2
        foxt = f2
    End If
    If foxt > f(xl) And foxt > f(xu) Then
        Do
            d = R * d
            If f1 > f2 Then
                xl = x2
                x2 = x1
            Else
                xu = x1
                x1 = x2
            End If
            f1 = f(x1)
            f2 = f(x2)
        Loop Until Abs(xu - xl) <= es
    End If
    ier = 1
End Sub
```

```

x1 = xL + d
f2 = f1
f1 = f(x1)
Else
    xU = x1
    x1 = x2
    x2 = xU - d
    f1 = f2
    f2 = f(x2)
End If
iter = iter + 1
If f1 > f2 Then
    xopt = x1
    fopt = f1
Else
    xopt = x2
    fopt = f2
End If
If xopt <> 0 Then ea = (1 - R) * Abs((xU - xL) / xopt) * 100
If ea <= es Or iter >= maxit Then Exit Do
Loop
Else
    ier = 1
End If
End Sub

Function f(x)
f = 2 * Sin(x) - x ^ 2 / 10
End Function

```

13.14 The easiest way to set up a maximization algorithm so that it can do minimization is to realize that minimizing a function is the same as maximizing its negative. Therefore, the following algorithm written in VBA minimizes or maximizes depending on the value of a user input variable, ind, where ind = -1 and 1 correspond to minimization and maximization, respectively. It is set up to solve the minimization described in Prob. 13.10.

```

Option Explicit

Sub GoldMinMax()
Dim ind As Integer      'Minimization (ind = -1); Maximization (ind = 1)
Dim xlow As Double, xhigh As Double
Dim xopt As Double, fopt As Double
xlow = 0.1
xhigh = 5
Call GoldMnMx(xlow, xhigh, -1, xopt, fopt)
MsgBox "xopt = " & xopt
MsgBox "f(xopt) = " & fopt
End Sub

Sub GoldMnMx(xlow, xhigh, ind, xopt, fopt)
Dim iter As Integer, maxit As Integer, ea As Double, es As Double
Dim xL As Double, xU As Double, d As Double, x1 As Double
Dim x2 As Double, f1 As Double, f2 As Double
Const R As Double = (5 ^ 0.5 - 1) / 2
maxit = 50
es = 0.001
xL = xlow
xU = xhigh
iter = 1
d = R * (xU - xL)

```

```

x1 = xL + d
x2 = xU - d
f1 = f(ind, x1)
f2 = f(ind, x2)
If f1 > f2 Then
    xopt = x1
    fopt = f1
Else
    xopt = x2
    fopt = f2
End If
Do
    d = R * d
    If f1 > f2 Then
        xL = x2
        x2 = x1
        x1 = xL + d
        f2 = f1
        f1 = f(ind, x1)
    Else
        xU = x1
        x1 = x2
        x2 = xU - d
        f1 = f2
        f2 = f(ind, x2)
    End If
    iter = iter + 1
    If f1 > f2 Then
        xopt = x1
        fopt = f1
    Else
        xopt = x2
        fopt = f2
    End If
    If xopt <> 0 Then ea = (1 - R) * Abs((xU - xL) / xopt) * 100
    If ea <= es Or iter >= maxit Then Exit Do
Loop
fopt = ind * fopt
End Sub

Function f(ind, x)
f = 2 * x + 3 / x 'place function to be evaluated here
f = ind * f
End Function

```

13.15 Because of multiple local minima and maxima, there is no really simple means to test whether a single maximum occurs within an interval without actually performing a search. However, if we assume that the function has one maximum and no minima within the interval, a check can be included. Here is a VBA program to implement the Quadratic Interpolation algorithm for maximization and solve Example 13.2.

```

Option Explicit

Sub QuadMax()
Dim ier As Integer
Dim xlow As Double, xhigh As Double
Dim xopt As Double, fopt As Double
xlow = 0
xhigh = 4
Call QuadMx(xlow, xhigh, xopt, fopt, ier)

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

If ier = 0 Then
    MsgBox "xopt = " & xopt
    MsgBox "f(xopt) = " & fopt
Else
    MsgBox "Does not appear to be maximum in [xl, xu]"
End If
End Sub

Sub QuadMx(xlow, xhigh, xopt, fopt, ier)
Dim iter As Integer, maxit As Integer, ea As Double, es As Double
Dim x0 As Double, x1 As Double, x2 As Double
Dim f0 As Double, f1 As Double, f2 As Double
Dim xoptOld As Double
ier = 0
maxit = 50
es = 0.0001
x0 = xlow
x2 = xhigh
x1 = (x0 + x2) / 2
f0 = f(x0)
f1 = f(x1)
f2 = f(x2)
If f1 > f0 Or f1 > f2 Then
    xoptOld = x1
    Do
        xopt = f0 * (x1 ^ 2 - x2 ^ 2) + f1 * (x2 ^ 2 - x0 ^ 2) + f2 * (x0 ^ 2 - x1 ^ 2)
        xopt = xopt / (2 * f0 * (x1 - x2) + 2 * f1 * (x2 - x0) + 2 * f2 * (x0 - x1))
        fopt = f(xopt)
        iter = iter + 1
        If xopt > x1 Then
            x0 = x1
            f0 = f1
            x1 = xopt
            f1 = fopt
        Else
            x2 = x1
            f2 = f1
            x1 = xopt
            f1 = fopt
        End If
        If xopt <> 0 Then ea = Abs((xopt - xoptOld) / xopt) * 100
        xoptOld = xopt
        If ea <= es Or iter >= maxit Then Exit Do
    Loop
Else
    ier = 1
End If
End Sub

Function f(x)
f = 2 * Sin(x) - x ^ 2 / 10
End Function

```

13.16 Here is a VBA program to implement the Newton-Raphson method for maximization. It is set up to duplicate the computation from Example 13.3.

```

Option Explicit

Sub NRMax()
Dim xguess As Double
Dim xopt As Double, fopt As Double

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

xguess = 2.5
Call NRMx(xguess, xopt, fopt)
MsgBox "xopt = " & xopt
MsgBox "f(xopt) = " & fopt
End Sub

Sub NRMx(xguess, xopt, fopt)
Dim iter As Integer, maxit As Integer, ea As Double, es As Double
Dim x0 As Double, x1 As Double, x2 As Double
Dim f0 As Double, f1 As Double, f2 As Double
Dim xoptOld As Double
maxit = 50
es = 0.01
Do
    xopt = xguess - df(xguess) / d2f(xguess)
    fopt = f(xopt)
    If xopt <> 0 Then ea = Abs((xopt - xguess) / xopt) * 100
    xguess = xopt
    If ea <= es Or iter >= maxit Then Exit Do
Loop
End Sub

Function f(x)
f = 2 * Sin(x) - x ^ 2 / 10
End Function

Function df(x)
df = 2 * Cos(x) - x / 5
End Function

Function d2f(x)
d2f = -2 * Sin(x) - 1 / 5
End Function

```

13.17 The first iteration of the golden-section search can be implemented as

$$d = \frac{\sqrt{5} - 1}{2}(4 - 2) = 1.2361$$

$$x_1 = 2 + 1.2361 = 3.2361$$

$$x_2 = 4 - 1.2361 = 2.7639$$

$$f(x_2) = f(2.7639) = -6.1303$$

$$f(x_1) = f(3.2361) = -5.8317$$

Because $f(x_2) < f(x_1)$, the minimum is in the interval defined by x_l , x_2 , and x_1 where x_2 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{4 - 2}{2.7639} \right| \times 100\% = 27.64\%$$

The process can be repeated and all the iterations summarized as

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d	x_{opt}	ε_a
1	2	-3.8608	2.7639	-6.1303	3.2361	-5.8317	4	-2.7867	1.2361	2.7639	27.64%
2	2	-3.8608	2.4721	-5.6358	2.7639	-6.1303	3.2361	-5.8317	0.7639	2.7639	17.08%

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

3	2.4721	-5.6358	2.7639	-6.1303	2.9443	-6.1776	3.2361	-5.8317	0.4721	2.9443	9.91%
4	2.7639	-6.1303	2.9443	-6.1776	3.0557	-6.1065	3.2361	-5.8317	0.2918	2.9443	6.13%

After four iterations, the process is converging on the true minimum at $x = 2.8966$ where the function has a value of $f(x) = -6.1847$.

13.18 The first iteration of the golden-section search can be implemented as

$$d = \frac{\sqrt{5} - 1}{2}(60 - 0) = 37.0820$$

$$x_1 = 0 + 37.0820 = 37.0820$$

$$x_2 = 60 - 37.0820 = 22.9180$$

$$f(x_2) = f(22.9180) = 18.336$$

$$f(x_1) = f(37.0820) = 19.074$$

Because $f(x_1) > f(x_2)$, the maximum is in the interval defined by x_2 , x_1 , and x_u where x_1 is the optimum. The error at this point can be computed as

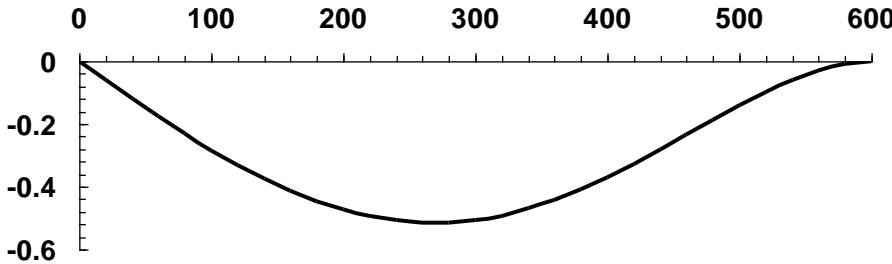
$$\varepsilon_a = (1 - 0.61803) \left| \frac{60 - 0}{37.0820} \right| \times 100\% = 61.80\%$$

The process can be repeated and all the iterations summarized as

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d	x_{opt}	ε_a
1	0	1	22.9180	18.336	37.0820	19.074	60	4.126	37.0820	37.0820	61.80%
2	22.9180	18.336	37.0820	19.074	45.8359	15.719	60	4.126	22.9180	37.0820	38.20%
3	22.9180	18.336	31.6718	19.692	37.0820	19.074	45.8359	15.719	14.1641	31.6718	27.64%
4	22.9180	18.336	28.3282	19.518	31.6718	19.692	37.0820	19.074	8.7539	31.6718	17.08%
5	28.3282	19.518	31.6718	19.692	33.7384	19.587	37.0820	19.074	5.4102	31.6718	10.56%
6	28.3282	19.518	30.3947	19.675	31.6718	19.692	33.7384	19.587	3.3437	31.6718	6.52%
7	30.3947	19.675	31.6718	19.692	32.4612	19.671	33.7384	19.587	2.0665	31.6718	4.03%
8	30.3947	19.675	31.1840	19.693	31.6718	19.692	32.4612	19.671	1.2772	31.1840	2.53%
9	30.3947	19.675	30.8825	19.689	31.1840	19.693	31.6718	19.692	0.7893	31.1840	1.56%
10	30.8825	19.689	31.1840	19.693	31.3703	19.693	31.6718	19.692	0.4878	31.3703	0.96%

After ten iterations, the process falls below the stopping criterion and the result is converging on the true maximum at $x = 31.3713$ where the function has a value of $y = 19.6934$.

13.19 (a) A graph indicates the minimum at about $x = 270$.



(b) Golden section search,

$$d = \frac{\sqrt{5} - 1}{2}(600 - 0) = 370.820$$

$$x_1 = 0 + 370.820 = 370.820$$

$$x_2 = 600 - 370.820 = 229.180$$

$$f(x_2) = f(229.180) = -0.5016$$

$$f(x_1) = f(370.820) = -0.4249$$

Because $f(x_2) < f(x_1)$, the minimum is in the interval defined by x_l , x_2 , and x_1 where x_2 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{600 - 0}{370.820} \right| \times 100\% = 100\%$$

The process can be repeated and all the iterations summarized as

<i>i</i>	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	<i>d</i>	x_{opt}	ε_a
1	0	0	229.180	-0.5016	370.820	-0.4249	600	0	370.820	229.1796	100.00%
2	0	0	141.641	-0.3789	229.180	-0.5016	370.820	-0.4249	229.180	229.1796	61.80%
3	141.641	-0.3789	229.180	-0.5016	283.282	-0.5132	370.820	-0.4249	141.641	283.2816	30.90%
4	229.180	-0.5016	283.282	-0.5132	316.718	-0.4944	370.820	-0.4249	87.539	283.2816	19.10%
5	229.180	-0.5016	262.616	-0.5149	283.282	-0.5132	316.718	-0.4944	54.102	262.6165	12.73%
6	229.180	-0.5016	249.845	-0.5121	262.616	-0.5149	283.282	-0.5132	33.437	262.6165	7.87%
7	249.845	-0.5121	262.616	-0.5149	270.510	-0.5151	283.282	-0.5132	20.665	270.5098	4.72%
8	262.616	-0.5149	270.510	-0.5151	275.388	-0.5147	283.282	-0.5132	12.772	270.5098	2.92%
9	262.616	-0.5149	267.495	-0.5152	270.510	-0.5151	275.388	-0.5147	7.893	267.4948	1.82%
10	262.616	-0.5149	265.631	-0.5151	267.495	-0.5152	270.510	-0.5151	4.878	267.4948	1.13%
11	265.631	-0.5151	267.495	-0.5152	268.646	-0.5152	270.510	-0.5151	3.015	268.6465	0.69%

After eleven iterations, the process falls below the stopping criterion and the result is converging on the true minimum at $x = 268.3281$ where the function has a value of $y = -0.51519$.

13.20 The velocity of a falling object with an initial velocity and first-order drag can be computed as

$$v = v_0 e^{-(c/m)t} + \frac{mg}{c} \left(1 - e^{-(c/m)t}\right)$$

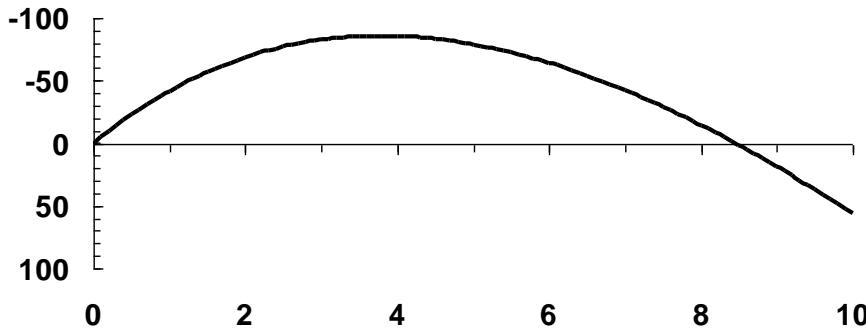
The vertical distance traveled can be determined by integration

$$z = z_0 + \int_0^t v_0 e^{-(c/m)t} + \frac{mg}{c} \left(1 - e^{-(c/m)t}\right) dt$$

where negative z is distance upwards. Assuming that $z_0 = 0$, evaluating the integral yields

$$z = \frac{mg}{c} t + \frac{m}{c} \left(v_0 - \frac{mg}{c} \right) \left(1 - e^{-(c/m)t} \right)$$

Therefore the solution to this problem amounts to determining the minimum of this function (since the most negative value of z corresponds to the maximum height). This function can be plotted using the given parameter values. As in the following graph (note that the ordinate values are plotted in reverse), the maximum height occurs after about 3.5 s and appears to be about 75 m.



Here is the result of using the golden-section search to determine the maximum height

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d	x_{opt}	ϵ_a
1	0	0	1.9098	-66.733	3.0902	-83.278	5	-78.926	3.0902	3.0902	61.80%
2	1.9098	-66.733	3.0902	-83.278	3.8197	-85.731	5	-78.926	1.9098	3.8197	30.90%
3	3.0902	-83.278	3.8197	-85.731	4.2705	-84.612	5.0000	-78.926	1.1803	3.8197	19.10%
4	3.0902	-83.278	3.5410	-85.439	3.8197	-85.731	4.2705	-84.612	0.7295	3.8197	11.80%
5	3.5410	-85.439	3.8197	-85.731	3.9919	-85.531	4.2705	-84.612	0.4508	3.8197	7.29%
6	3.5410	-85.439	3.7132	-85.711	3.8197	-85.731	3.9919	-85.531	0.2786	3.8197	4.51%
7	3.7132	-85.711	3.8197	-85.731	3.8854	-85.688	3.9919	-85.531	0.1722	3.8197	2.79%
8	3.7132	-85.711	3.7790	-85.737	3.8197	-85.731	3.8854	-85.688	0.1064	3.7790	1.74%
9	3.7132	-85.711	3.7539	-85.732	3.7790	-85.737	3.8197	-85.731	0.0658	3.7790	1.08%
10	3.7539	-85.732	3.7790	-85.737	3.7945	-85.736	3.8197	-85.731	0.0407	3.7790	0.66%

After ten iterations, the process falls below a stopping of 1% criterion and the result is converging on the true minimum at $x = 3.78588$ where the function has a value of $y = -85.7367$. Thus, the maximum height is 74.811.

13.21 The inflection point corresponds to the point at which the derivative of the normal distribution is a minimum. The derivative can be evaluated as

$$\frac{dy}{dx} = -2xe^{-x^2}$$

Starting with initial guesses of $x_l = 0$ and $x_u = 2$, the golden-section search method can be implemented as

$$d = \frac{\sqrt{5}-1}{2}(2-0) = 1.2361$$

$$x_1 = 0 + 1.2361 = 1.2361$$

$$x_2 = 2 - 1.2361 = 0.7639$$

$$f(x_2) = f(0.7639) = -2(0.7639)e^{-(0.7639)^2} = -0.8524$$

$$f(x_1) = f(1.2361) = -2(1.2361)e^{-(1.2361)^2} = -0.5365$$

Because $f(x_2) < f(x_1)$, the minimum is in the interval defined by x_l , x_2 , and x_1 where x_2 is the optimum. The error at this point can be computed as

$$\varepsilon_a = (1 - 0.61803) \left| \frac{2-0}{0.7639} \right| \times 100\% = 100\%$$

The process can be repeated and all the iterations summarized as

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d	x_{opt}	ε_a
1	0	0.0000	0.7639	-0.8524	1.2361	-0.5365	2	-0.0733	1.2361	0.7639	100.00%
2	0	0.0000	0.4721	-0.7556	0.7639	-0.8524	1.2361	-0.5365	0.7639	0.7639	61.80%
3	0.4721	-0.7556	0.7639	-0.8524	0.9443	-0.7743	1.2361	-0.5365	0.4721	0.7639	38.20%
4	0.4721	-0.7556	0.6525	-0.8525	0.7639	-0.8524	0.9443	-0.7743	0.2918	0.6525	27.64%
5	0.4721	-0.7556	0.5836	-0.8303	0.6525	-0.8525	0.7639	-0.8524	0.1803	0.6525	17.08%
6	0.5836	-0.8303	0.6525	-0.8525	0.6950	-0.8575	0.7639	-0.8524	0.1115	0.6950	9.91%
7	0.6525	-0.8525	0.6950	-0.8575	0.7214	-0.8574	0.7639	-0.8524	0.0689	0.6950	6.13%
8	0.6525	-0.8525	0.6788	-0.8564	0.6950	-0.8575	0.7214	-0.8574	0.0426	0.6950	3.79%
9	0.6788	-0.8564	0.6950	-0.8575	0.7051	-0.8578	0.7214	-0.8574	0.0263	0.7051	2.31%
10	0.6950	-0.8575	0.7051	-0.8578	0.7113	-0.8577	0.7214	-0.8574	0.0163	0.7051	1.43%
11	0.6950	-0.8575	0.7013	-0.8577	0.7051	-0.8578	0.7113	-0.8577	0.0100	0.7051	0.88%

After eleven iterations, the process falls below a stopping of 1% criterion and the result is converging on the true minimum at $x = 0.707107$ where the function has a value of $y = -0.85776$.

CHAPTER 14

14.1 The elevation can be determined as

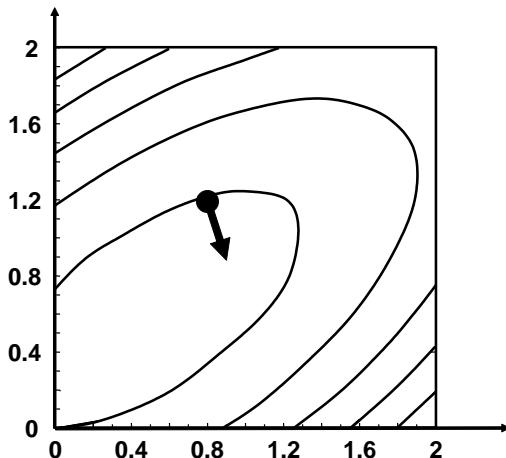
$$f(0.8, 1.2) = 2(0.8)1.2 + 1.5(1.2) - 1.25(0.8)^2 - 2(1.2)^2 + 5 = 5.04$$

The partial derivatives can be evaluated,

$$\frac{\partial f}{\partial x} = 2y - 2.5x = 2(1.2) - 2.5(0.8) = 0.4$$

$$\frac{\partial f}{\partial y} = 2x + 1.5 - 4y = 2(0.8) + 1.5 - 4(1.2) = -1.7$$

which can be used to determine the gradient as $\nabla f = 0.4\mathbf{i} - 1.7\mathbf{j}$. This corresponds to the direction $\theta = \tan^{-1}(-1.7/0.4) = -1.3397$ radians ($= -76.76^\circ$). This vector can be sketched on a topographical map of the function as shown below:



The slope in this direction can be computed as

$$\sqrt{0.4^2 + (-1.7)^2} = 1.746$$

14.2 The partial derivatives can be evaluated,

$$\frac{\partial f}{\partial x} = 2x = 2(2) = 4$$

$$\frac{\partial f}{\partial y} = 4y = 4(2) = 8$$

The angle in the direction of \mathbf{h} is

$$\theta = \tan^{-1}\left(\frac{3}{2}\right) = 0.9828 \text{ radians } (= 56.31^\circ)$$

The directional derivative can be computed as

$$g'(0) = 4\cos(0.9828) + 8\sin(0.9828) = 8.875$$

14.3 (a)

$$\nabla f = \begin{Bmatrix} 3y^2 + 2ye^{xy} \\ 6xy + 2xe^{xy} \end{Bmatrix} \quad H = \begin{bmatrix} 2y^2 e^{xy} & 6y + 2xye^{xy} + 2e^{xy} \\ 6y + 2xye^{xy} + 2e^{xy} & 6x + 2x^2 e^{xy} \end{bmatrix}$$

(b)

$$\nabla f = \begin{Bmatrix} 4x \\ 2y \\ 2z \end{Bmatrix} \quad H = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

(c)

$$\nabla f = \begin{Bmatrix} 2x + 3y \\ x^2 + 3xy + 2y^2 \\ 3x + 4y \\ x^2 + 3xy + 2y^2 \end{Bmatrix} \quad H = \frac{\begin{bmatrix} -2x^2 - 6xy - 5y^2 & -3x^2 - 8xy - 6y^2 \\ -3x^2 - 8xy - 6y^2 & -5x^2 - 12xy - 8y^2 \end{bmatrix}}{(x^2 + 3xy + 2y^2)^2}$$

14.4

The partial derivatives can be evaluated,

$$\frac{\partial f}{\partial x} = -3x + 2.25y$$

$$\frac{\partial f}{\partial y} = 2.25x - 4y + 1.75$$

These can be set to zero to generate the following simultaneous equations

$$3x - 2.25y = 0$$

$$-2.25x + 4y = 1.75$$

which can be solved for $x = 0.567568$ and $y = 0.756757$, which is the optimal solution.

14.5

The partial derivatives can be evaluated at the initial guesses, $x = 1$ and $y = 1$,

$$\frac{\partial f}{\partial x} = -3x + 2.25y = -3(1) + 2.25(1) = -0.75$$

$$\frac{\partial f}{\partial y} = 2.25x - 4y + 1.75 = 2.25(1) - 4(1) + 1.75 = 0$$

Therefore, the search direction is $-0.75\mathbf{i}$.

$$f(1 - 0.75h, 1) = 0.5 + 0.5625h - 0.84375h^2$$

This can be differentiated and set equal to zero and solved for $h^* = 0.33333$. Therefore, the result for the first iteration is $x = 1 - 0.75(0.3333) = 0.75$ and $y = 1 + 0(0.3333) = 1$.

For the second iteration, the partial derivatives can be evaluated as,

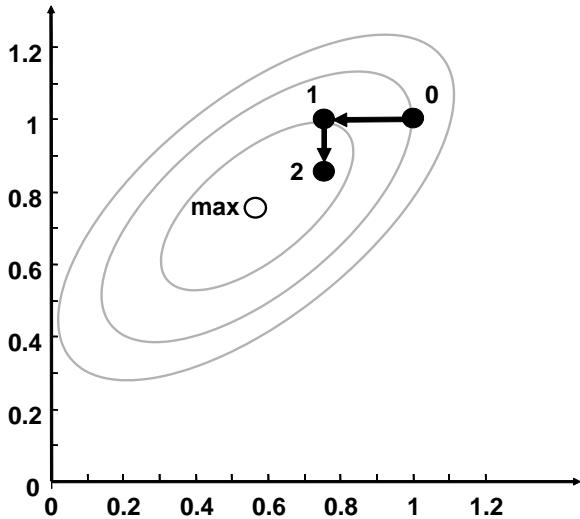
$$\frac{\partial f}{\partial x} = -3(0.75) + 2.25(1) = 0$$

$$\frac{\partial f}{\partial y} = 2.25(0.75) - 4(1) + 1.75 = -0.5625$$

Therefore, the search direction is $-0.5625\mathbf{j}$.

$$f(0.75, 1 - 0.5625h) = 0.59375 + 0.316406h - 0.63281h^2$$

This can be differentiated and set equal to zero and solved for $h^* = 0.25$. Therefore, the result for the second iteration is $x = 0.75 + 0(0.25) = 0.75$ and $y = 1 + (-0.5625)0.25 = 0.859375$.



14.6 The partial derivatives can be evaluated at the initial guesses, $x = 1$ and $y = 1$,

$$\frac{\partial f}{\partial x} = 2(x - 3) = 2(1 - 3) = -4$$

$$\frac{\partial f}{\partial y} = 2(y - 2) = 2(1 - 2) = -2$$

$$f(1 - 4h, 1 - 2h) = (1 - 4h - 3)^2 + (1 - 2h - 2)^2$$

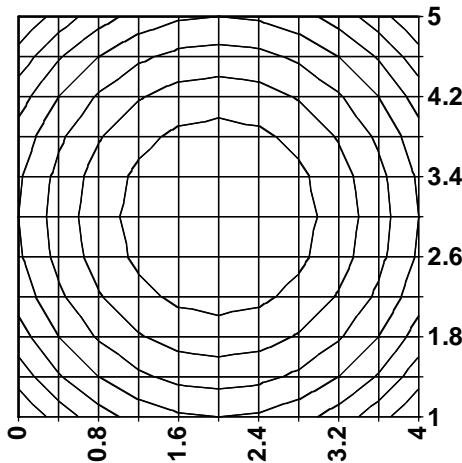
$$g(h) = (-4h - 2)^2 + (-2h - 1)^2$$

Setting $g'(h) = 0$ gives $h^* = -0.5$. Therefore,

$$x = 1 - 4(-0.5) = 3$$

$$y = 1 - 2(-0.5) = 2$$

Thus, for this special case, the approach converges on the correct answer after a single iteration. This occurs because the function is spherical as shown below. Thus, the gradient for any guess points directly at the solution.



14.7 The partial derivatives can be evaluated at the initial guesses, $x = 0$ and $y = 0$,

$$\frac{\partial f}{\partial x} = 4 + 2x - 8x^3 + 2y = 4 + 2(0) - 8(0)^3 + 2(0) = 4$$

$$\frac{\partial f}{\partial y} = 2 + 2x - 6y = 2 + 2(0) - 6(0) = 2$$

$$f(0 + 4h, 0 + 2h) = 20h + 20h^2 - 512h^4$$

$$g'(h) = 20 + 40h - 2048h^3$$

The root of this equation can be determined by bisection. Using initial guesses of $h = 0$ and 1 yields a root of $h^* = 0.244$ after 13 iterations with $\varepsilon_a = 0.05\%$.

iteration	h_l	h_u	h_r	$g(h)$	$g(h_r)$	$g(h)^*g(h_r)$	ε_a
1	0.00000	1.00000	0.50000	20.000	-216.000	-4320.00	100.00%

2	0.00000	0.50000	0.25000	20.000	-2.000	-40.00	100.00%
3	0.00000	0.25000	0.12500	20.000	21.000	420.00	100.00%
4	0.12500	0.25000	0.18750	21.000	14.000	294.00	33.33%
5	0.18750	0.25000	0.21875	14.000	7.313	102.38	14.29%
6	0.21875	0.25000	0.23438	7.313	3.008	21.99	6.67%
7	0.23438	0.25000	0.24219	3.008	0.595	1.79	3.23%
8	0.24219	0.25000	0.24609	0.595	-0.680	-0.40	1.59%
9	0.24219	0.24609	0.24414	0.595	-0.037	-0.02	0.80%
10	0.24219	0.24414	0.24316	0.595	0.280	0.17	0.40%
11	0.24316	0.24414	0.24365	0.280	0.122	0.03	0.20%
12	0.24365	0.24414	0.24390	0.122	0.043	0.01	0.10%
13	0.24390	0.24414	0.24402	0.043	0.003	0.00	0.05%

Therefore,

$$x = 0 + 4(0.244) = 0.976$$

$$y = 0 + 2(0.244) = 0.488$$

14.8

$$\frac{\partial f}{\partial x} = -8 + 2x - 2y$$

$$\frac{\partial f}{\partial y} = 12 + 8y - 2x$$

At $x = y = 0$,

$$\frac{\partial f}{\partial x} = -8$$

$$\frac{\partial f}{\partial y} = 12$$

$$f(0 - 8h, 0 + 12h) = g(h)$$

$$g(h) = 832h^2 + 208h$$

At $g'(h) = 0$, $h^* = -0.125$.

Therefore,

$$x = 0 - 8(-0.125) = 1$$

$$y = 0 + 12(-0.125) = -1.5$$

14.9

The following code implements the random search algorithm in VBA. It is set up to solve Prob. 14.7.

```
Option Explicit
```

```

Sub RandSearch()
Dim n As Long
Dim xmin As Double, xmax As Double, ymin As Double, ymax As Double
Dim maxf As Double, maxx As Double, maxy As Double
xmin = -2: xmax = 2: ymin = -2: ymax = 2
n = InputBox("n=")
Call RndSrch(n, xmin, xmax, ymin, ymax, maxx, maxx, maxf)
MsgBox maxf
MsgBox maxx
MsgBox maxy
End Sub

Sub RndSrch(n, xmin, xmax, ymin, ymax, maxy, maxx, maxf)
Dim j As Long
Dim x As Double, y As Double, fn As Double
maxf = -1000000000#
For j = 1 To n
    x = xmin + (xmax - xmin) * Rnd
    y = ymin + (ymax - ymin) * Rnd
    fn = f(x, y)
    If fn > maxf Then
        maxf = fn
        maxx = x
        maxy = y
    End If
Next j
End Sub

Function f(x, y)
f = 4 * x + 2 * y + x ^ 2 - 2 * x ^ 4 + 2 * x * y - 3 * y ^ 2
End Function

```

The result of running this program for different number of iterations yields the results in the following table. We have also included to exact result.

<i>n</i>	<i>f(x, y)</i>	<i>x</i>	<i>y</i>
1000	4.31429	1.011782	0.613747
10000	4.34185	0.961359	0.678104
100000	4.34338	0.964238	0.641633
1000000	4.34397	0.965868	0.656765
10000000	4.34401	0.967520	0.655715
truth	4.34401	0.967580	0.655860

14.10 The following code implements the grid search algorithm in VBA:

```

Option Explicit

Sub GridSearch()
Dim nx As Long, ny As Long
Dim xmin As Double, xmax As Double, ymin As Double, ymax As Double
Dim maxf As Double, maxx As Double, maxy As Double
xmin = -2: xmax = 2: ymin = 1: ymax = 3
nx = 1000
ny = 1000
Call GridSrch(nx, ny, xmin, xmax, ymin, ymax, maxy, maxx, maxf)
MsgBox maxf
MsgBox maxx
MsgBox maxy

```

```

End Sub

Sub GridSrch(nx, ny, xmin, xmax, ymin, ymax, maxy, maxx, maxf)
Dim i As Long, j As Long
Dim x As Double, y As Double, fn As Double
Dim xinc As Double, yinc As Double
xinc = (xmax - xmin) / nx
yinc = (ymax - ymin) / ny
maxf = -1000000000#
x = xmin
For i = 0 To nx
    y = ymin
    For j = 0 To ny
        fn = f(x, y)
        If fn > maxf Then
            maxf = fn
            maxx = x
            maxy = y
        End If
        y = y + yinc
    Next j
    x = x + xinc
Next i
End Sub

Function f(x, y)
f = y - x - 2 * x ^ 2 - 2 * x * y - y ^ 2
End Function

```

14.11

$$f(x, y) = 6x^2y - 9y^2 - 8x^2$$

$$\frac{\partial f}{\partial x} = 12xy - 16x \Rightarrow 12(2)(4) - 16(4) = 32$$

$$\frac{\partial f}{\partial y} = 6x^2 - 18y \Rightarrow 6(4)^2 - 18(2) = 60$$

$$\nabla f = 32\hat{i} + 60\hat{j}$$

$$\begin{aligned}
f\left(x_o + \frac{\partial f}{\partial x}h, y_o + \frac{\partial f}{\partial y}h\right) &= f(4 + 32h, 2 + 60h) \\
&= 6(4 + 32h)^2(2 + 60h) - 9(2 + 60h)^2 - 8(4 + 32h)^2
\end{aligned}$$

$$g(x) = 368,640h^3 + 63,856h^2 + 4,624h + 28$$

14.12

$$f(x, y) = 2x^3y^2 - 7yx + x^2 + 3y$$

$$\frac{\partial f}{\partial x} = 6x^2y^2 - 7y + 2x \Rightarrow 6(1)(1) - 7(1) + 2(1) = 1$$

$$\frac{\partial f}{\partial x} = 4x^3y - 7x + 3 \Rightarrow 4(1)(1) - 7(1) + 3 = 0$$

$$\nabla f = 1\hat{i} + 0\hat{j}$$

$$\begin{aligned} f(x_o + \frac{\partial f}{\partial x} h, y_o + \frac{\partial f}{\partial y} h) &= f(1+h, 1+0h) \\ &= 2(1+h)^3(1)^2 - 7(1+h)(1) + (1+h)^2 + 3(1) \end{aligned}$$

$$g(x) = 2h^3 + 7h^2 + h - 1$$

CHAPTER 15

15.1 (a) Define x_a = amount of product A produced, and x_b = amount of product B produced. The objective function is to maximize profit,

$$P = 45x_a + 20x_b$$

Subject to the following constraints

$20x_a + 5x_b \leq 9500$	{raw materials}
$0.04x_a + 0.12x_b \leq 40$	{production time}
$x_a + x_b \leq 550$	{storage}
$x_a, x_b \geq 0$	{positivity}

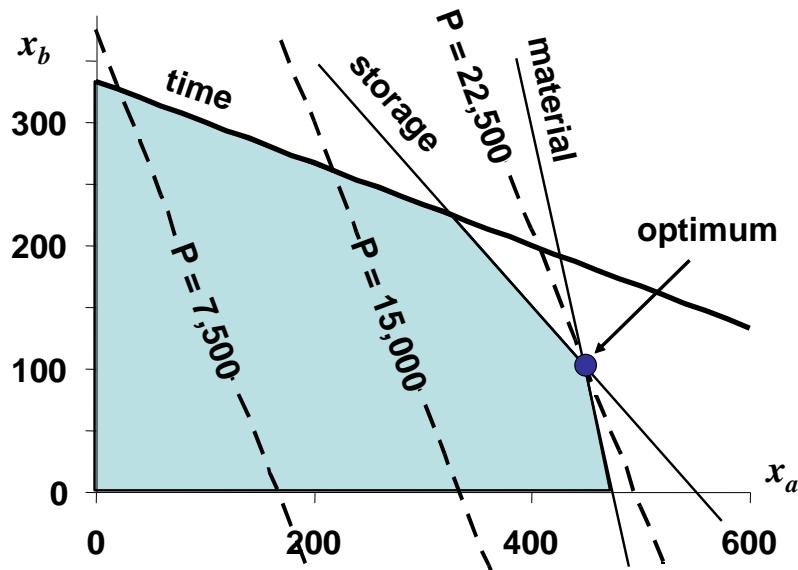
(b) To solve graphically, the constraints can be reformulated as the following straight lines

$x_b = 1900 - 4x_a$	{raw materials}
$x_b = 333.3333 - 0.333333x_a$	{production time}
$x_b = 550 - x_a$	{storage}

The objective function can be reformulated as

$$x_b = (1/20)P - 2.25x_a$$

The constraint lines can be plotted on the x_a - x_b plane to define the feasible space. Then the objective function line can be superimposed for various values of P until it reaches the boundary. The result is $P \approx 22,250$ with $x_a \approx 450$ and $x_b \approx 100$. Notice also that material and storage are the binding constraints and that there is some slack in the time constraint.



(c) The simplex tableau for the problem can be set up and solved as

Basis	P	x_a	x_b	S_1	S_2	S_3	Solution	Intercept
P	1	-45	-20	0	0	0	0	
S_1	0	20	5	1	0	0	9500	475
S_2	0	0.04	0.12	0	1	0	40	1000
S_3	0	1	1	0	0	1	550	550

Basis	P	x_a	x_b	S_1	S_2	S_3	Solution	Intercept
P	1	0	-8.75	2.25	0	0	21375	
x_a	0	1	0.25	0.05	0	0	475	1900
S_2	0	0	0.11	-0.002	1	0	21	190.9091
S_3	0	0	0.75	-0.05	0	1	75	100

Basis	P	x_a	x_b	S_1	S_2	S_3	Solution	Intercept
P	1	0	0	1.666667	0	11.666667	22250	
x_a	0	1	0	0.066667	0	-0.33333	450	
S_2	0	0	0	0.005333	1	-0.14667	10	
x_b	0	0	1	-0.06667	0	1.333333	100	

(d) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E
1		x_A	x_B	total	constraint
2	amount	0	0		
3	time	0.04	0.12	0	40
4	storage	1	1	0	550
5	raw material	20	5	0	9500
6	profit	45	20	0	

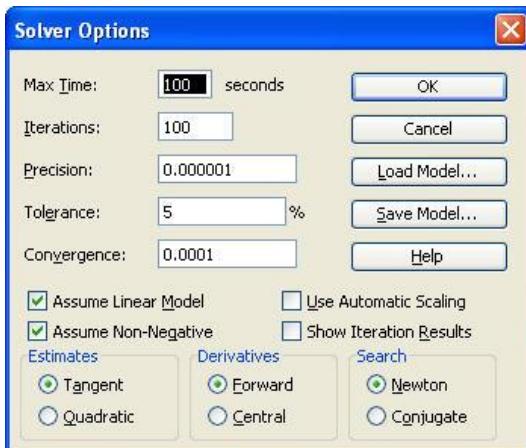
The formulas in column D are

	A	B	C	D	E
1		x_A	x_B	total	constraint
2	amount	0	0		
3	time	0.04	0.12	=B3*B\$2+C3*C\$2	40
4	storage	1	1	=B4*B\$2+C4*C\$2	550
5	raw material	20	5	=B5*B\$2+C5*C\$2	9500
6	profit	45	20	=B6*B\$2+C6*C\$2	

The Solver can be called and set up as



Before depressing the Solve button, depress the Options button and check the boxes to “Assume Linear Model” and “Assume Non-Negative.”



The resulting solution is

	A	B	C	D	E
1		xA	xB	total	constraint
2	amount	450	100		
3	time	0.04	0.12	30	40
4	storage	1	1	550	550
5	raw material	20	5	9500	9500
6	profit	45	20	22250	

In addition, a sensitivity report can be generated as

	A	B	C	D	E	F	G	H
1	Microsoft Excel 11.0 Sensitivity Report							
2	Worksheet: [prob1501.xls]Graphical							
3	Report Created: 6/30/2005 3:19:02 PM							
4								
5								
6	Adjustable Cells							
7			Final	Reduced	Objective	Allowable	Allowable	
8	Cell	Name	Value	Cost	Coefficient	Increase	Decrease	
9	\$B\$2	amount xA	450	0	45	35	25	
10	\$C\$2	amount xB	100	0	20	25	8.75	
11								
12	Constraints							
13			Final	Shadow	Constraint	Allowable	Allowable	
14	Cell	Name	Value	Price	R.H. Side	Increase	Decrease	
15	\$D\$3	time total	30	0	40	1E+30	10	
16	\$D\$4	storage total	550	11.66666667	550	68.18181818	75	
17	\$D\$5	raw material total	9500	1.666666667	9500	1500	1875	

- (e) The high shadow price for storage from the sensitivity analysis from (d) suggests that increasing storage will result in the best increase in profit.

15.2 (a) The LP formulation is given by

$$\text{Maximize } Z = 150x_1 + 175x_2 + 250x_3 \quad \{\text{Maximize profit}\}$$

subject to

$$\begin{aligned}
 7x_1 + 11x_2 + 15x_3 &\leq 154 && \{\text{Material constraint}\} \\
 10x_1 + 8x_2 + 12x_3 &\leq 80 && \{\text{Time constraint}\} \\
 x_1 &\leq 9 && \{\text{“Regular” storage constraint}\} \\
 x_2 &\leq 6 && \{\text{“Premium” storage constraint}\} \\
 x_3 &\leq 5 && \{\text{“Supreme” storage constraint}\} \\
 x_1, x_2, x_3 &\geq 0 && \{\text{Positivity constraints}\}
 \end{aligned}$$

(b) The simplex tableau for the problem can be set up and solved as

Basis	Z	x_1	x_2	x_3	S_1	S_2	S_3	S_4	S_5	Solution	Intercept
Z	1	-150	-175	-250	0	0	0	0	0	0	0
S1	0	7	11	15	1	0	0	0	0	154	10.2667
S2	0	10	8	12	0	1	0	0	0	80	6.66667
S3	0	1	0	0	0	0	1	0	0	9	∞
S4	0	0	1	0	0	0	0	1	0	6	∞
S5	0	0	0	1	0	0	0	0	1	5	5

Basis	Z	x_1	x_2	x_3	S_1	S_2	S_3	S_4	S_5	Solution	Intercept
Z	1	-150	-175	0	0	0	0	0	250	1250	
S1	0	7	11	0	1	0	0	0	-15	79	7.18182
S2	0	10	8	0	0	1	0	0	-12	20	2.5
S3	0	1	0	0	0	0	1	0	0	9	∞
S4	0	0	1	0	0	0	0	1	0	6	6

x3	0	0	0	1	0	0	0	0	1	5	∞
Basis	Z	x_1	x_2	x_3	S_1	S_2	S_3	S_4	S_5	Solution	Intercept
Z	1	68.75	0	0	0	21.88	0	0	-12.5	1687.5	
S1	0	-6.75	0	0	1	-1.375	0	0	1.5	51.5	34.3333
x2	0	1.25	1	0	0	0.125	0	0	-1.5	2.5	-1.66667
S3	0	1	0	0	0	0	1	0	0	9	∞
S4	0	-1.25	0	0	0	-0.125	0	1	1.5	3.5	2.33333
x3	0	0	0	1	0	0	0	0	1	5	5

Basis	Z	x_1	x_2	x_3	S_1	S_2	S_3	S_4	S_5	Solution
Z	1	58.3333	0	0	0	20.83	0	8.33	0	1716.7
S1	0	-5.5	0	0	1	-1.25	0	-1	0	48
x2	0	0	1	0	0	0	0	1	0	6
S3	0	1	0	0	0	0	1	0	0	9
S5	0	-0.8333	0	0	0	-0.083	0	0.67	1	2.3333
x3	0	0.8333	0	1	0	0.083	0	-0.67	0	2.6667

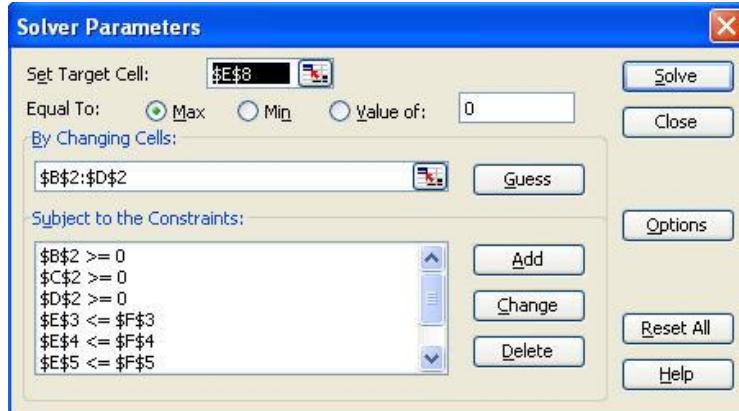
(c) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E	F
1		regular	premium	supreme	total	constraint
2	amount	0	0	0		
3	material	7	11	15	0	154
4	time	10	8	12	0	80
5	reg stor	1	0	0	0	9
6	prem stor	0	1	0	0	6
7	sup stor	0	0	1	0	5
8	profit	150	175	250	0	

The formulas in column E are

	A	B	C	D	E	F
1		regular	premium	supreme	total	constraint
2	amount	0	0	0		
3	material	7	11	15	=B3*B\$2+C3*C\$2+D3*D\$2	154
4	time	10	8	12	=B4*B\$2+C4*C\$2+D4*D\$2	80
5	reg stor	1	0	0	=B5*B\$2+C5*C\$2+D5*D\$2	9
6	prem stor	0	1	0	=B6*B\$2+C6*C\$2+D6*D\$2	6
7	sup stor	0	0	1	=B7*B\$2+C7*C\$2+D7*D\$2	5
8	profit	150	175	250	=B8*B\$2+C8*C\$2+D8*D\$2	

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E	F
1		regular	premium	supreme	total	constraint
2	amount	0	6	2.6666667		
3	material	7	11	15	106	154
4	time	10	8	12	80	80
5	reg stor	1	0	0	0	9
6	prem stor	0	1	0	6	6
7	sup stor	0	0	1	2.6666667	5
8	profit	150	175	250	1716.667	

In addition, a sensitivity report can be generated as

	A	B	C	D	E	F	G	H
1	Microsoft Excel 11.0 Sensitivity Report							
2	Worksheet: [Book1]Sheet1							
3	Report Created: 6/24/2005 2:41:55 PM							
4								
5								
6	Adjustable Cells							
7		Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
8		\$B\$2	amount regular	0	-58.33333333	150	58.33333333	1E+30
9		\$C\$2	amount premium	6	0	175	1E+30	8.333333334
10		\$D\$2	amount supreme	2.666666667	0	250	12.5	70
11								
12								
13	Constraints							
14		Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
15		\$E\$3	material total	106	0	154	1E+30	48
16		\$E\$4	time total	80	20.83333333	80	28	32
17		\$E\$5	reg stor total	0	0	9	1E+30	9
18		\$E\$6	prem stor total	6	8.333333334	6	4	3.5
19		\$E\$7	sup stor total	2.666666667	0	5	1E+30	2.333333333
20								

(d) The high shadow price for time from the sensitivity analysis from (c) suggests that increasing time will result in the best increase in profit.

15.3 (a) To solve graphically, the constraints can be reformulated as the following straight lines

$$y = 6.22222 - 0.53333x$$

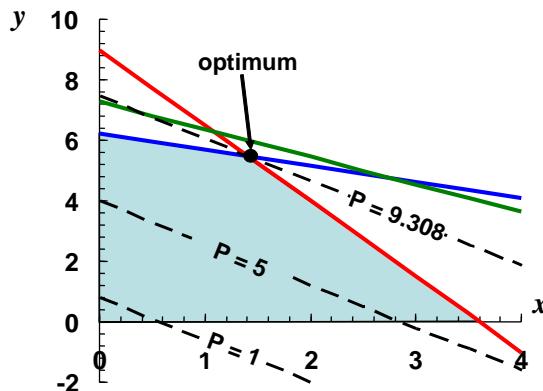
$$y = 7.2727 - 0.90909x$$

$$y = 9 - 2.5x$$

The objective function can be reformulated as

$$y = 0.8P - 1.4x$$

The constraint lines can be plotted on the x - y plane to define the feasible space. Then the objective function line can be superimposed for various values of P until it reaches the boundary. The result is $P \cong 9.30791$ with $x \cong 1.4$ and $y \cong 5.5$.



(b) The simplex tableau for the problem can be set up and solved as

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	-1.75	-1.25	0	0	0	0	
S_1	0	1.2	2.25	1	0	0	14	11.66667
S_2	0	1	1.1	0	1	0	8	8
S_3	0	2.5	1	0	0	1	9	3.6

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	0	-0.55	0	0	0.7	6.3	
S_1	0	0	1.77	1	0	-0.48	9.68	5.468927
S_2	0	0	0.7	0	1	-0.4	4.4	6.285714
x	0	1	0.4	0	0	0.4	3.6	9

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	0	0	0.310734	0	0.550847	9.30791	
y	0	0	1	0.564972	0	-0.27119	5.468927	
S_2	0	0	0	-0.39548	1	-0.21017	0.571751	
x	0	1	0	-0.22599	0	0.508475	1.412429	

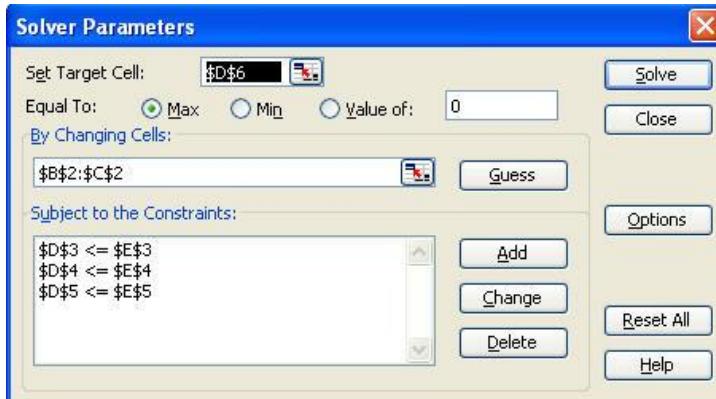
(c) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E
1		x	y	total	constraint
2	amount	0	0		
3	constraint 1	1.2	2.25	0	14
4	constraint 2	1	1.1	0	8
5	constraint 3	2.5	1	0	9
6	profit	1.75	1.25	0	

The formulas in column D are

	A	B	C	D	E
1		x	y	total	constraint
2	amount	0	0		
3	constraint 1	1.2	2.25	=B3*B\$2+C3*C\$2	14
4	constraint 2	1	1.1	=B4*B\$2+C4*C\$2	8
5	constraint 3	2.5	1	=B5*B\$2+C5*C\$2	9
6	profit	1.75	1.25	=B6*B\$2+C6*C\$2	

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E
1		x	y	total	constraint
2	amount	1.412429	5.468927		
3	constraint 1	1.2	2.25	14	14
4	constraint 2	1	1.1	7.428249	8
5	constraint 3	2.5	1	9	9
6	profit	1.75	1.25	9.30791	

15.4 (a) To solve graphically, the constraints can be reformulated as the following straight lines

$$y = 20 - 2.5x$$

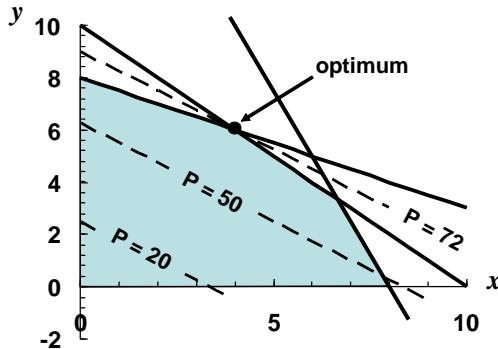
$$y = 10 - 10x$$

$$y = 8 - 0.5x$$

The objective function can be reformulated as

$$y = 0.125P - 0.75x$$

The constraint lines can be plotted on the x - y plane to define the feasible space. Then the objective function line can be superimposed for various values of P until it reaches the boundary. The result is $P \geq 72$ with $x \geq 4$ and $y \geq 6$.



(b) The simplex tableau for the problem can be set up and solved as

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	-6	-8	0	0	0	0	
S_1	0	5	2	1	0	0	40	20
S_2	0	6	6	0	1	0	60	10
S_3	0	2	4	0	0	1	32	8

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	-2	0	0	0	2	64	
S_1	0	4	0	1	0	-0.5	24	6
S_2	0	3	0	0	1	-1.5	12	4
y	0	0.5	1	0	0	0.25	8	16

Basis	P	x	y	S_1	S_2	S_3	Solution	Intercept
P	1	0	0	0	0.666667	1	72	
S_1	0	0	0	1	-1.333333	1.5	8	
x	0	1	0	0	0.333333	-0.5	4	
y	0	0	1	0	-0.166667	0.5	6	

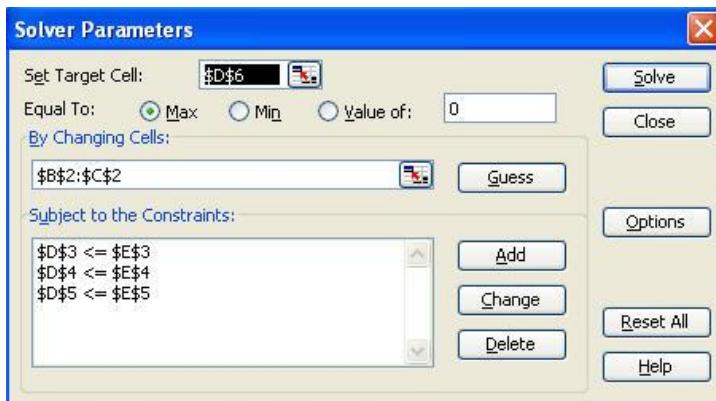
(c) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E
1			x	y	total constraint
2 amount		0	0		
3 constraint 1		5	2	0	40
4 constraint 2		6	6	0	60
5 constraint 3		2	4	0	32
6 profit		6	8	0	

The formulas in column D are

	A	B	C	D	E
1		x	y	total	constraint
2	amount	0	0		
3	constraint 1	5	2	=B3*B\$2+C3*C\$2	40
4	constraint 2	6	6	=B4*B\$2+C4*C\$2	60
5	constraint 3	2	4	=B5*B\$2+C5*C\$2	32
6	profit	6	8	=B6*B\$2+C6*C\$2	

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E
1		x	y	total	constraint
2	amount	4	6		
3	constraint 1	5	2	32	40
4	constraint 2	6	6	60	60
5	constraint 3	2	4	32	32
6	profit	6	8	72	

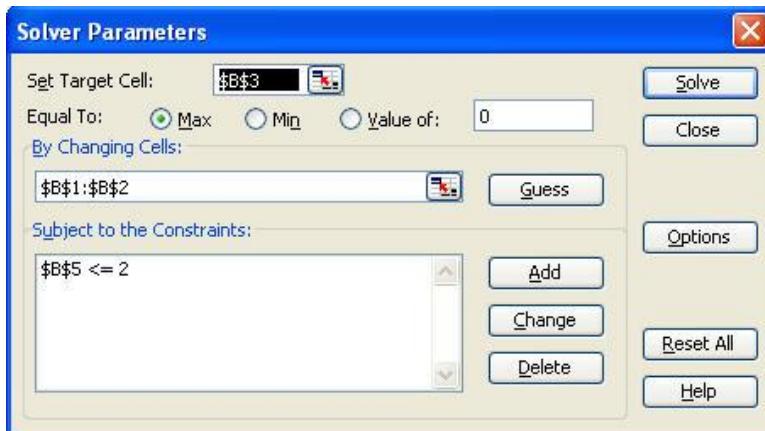
15.5 An Excel spreadsheet can be set up to solve the problem as

	A	B
1	x	0
2	y	0
3	f(x,y)	0
4	Constraint:	
5	2x+y=	0

The formulas are

	A	B
1	x	0
2	y	0
3	f(x,y)	=1.2*B1+2*B2-B2^3
4	Constraint:	
5	2x+y=	=2*B1+B2

The Solver can be called and set up as



The resulting solution is

	A	B
1	x	0.658435
2	y	0.68313
3	f(x,y)	1.837588
4	Constraint:	
5	2x+y=	2

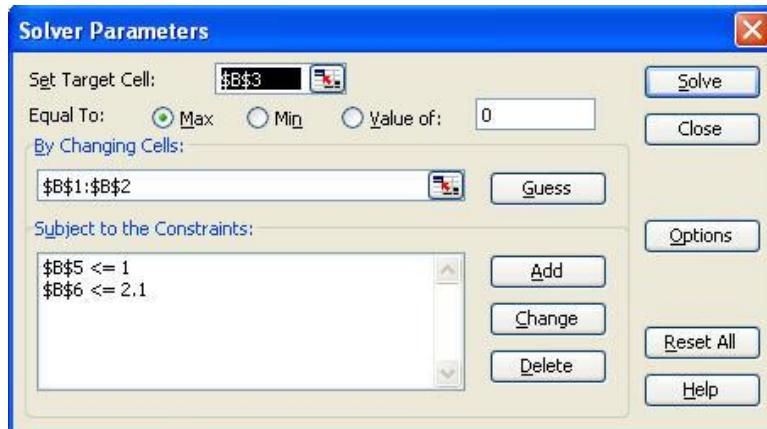
15.6 An Excel spreadsheet can be set up to solve the problem as

	A	B
1	x	0
2	y	0
3	f(x,y)	0
4	Constraints:	
5	x^2+y^2	0
6	$x+2y$	0

The formulas are

	A	B
1	x	0
2	y	0
3	f(x,y)	=15*B1+15*B2
4	Constraints:	
5	x^2+y^2	=B1^2+B2^2
6	$x+2y$	=B1+2*B2

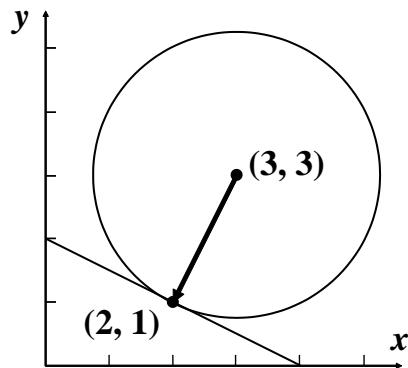
The Solver can be called and set up as



The resulting solution is

	A	B
1	x	0.727247
2	y	0.686377
3	f(x,y)	21.20435
4	Constraints:	
5	x^2+y^2	1.000001
6	x+2y	2.1

- 15.7 (a) The function and the constraint can be plotted and as shown indicate a solution of $x = 2$ and $y = 1$.



- (b) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D
1	x	0		
2	y	0		
3	Minimize			
4	f(x,y)	18		
5	Subject to			
6	x+2y =	0	=	4

The formulas are

	A	B	C	D
1	x	0		
2	y	0		
3	Minimize			
4	f(x,y)	=B1^2+(B2-3)^2		
5	Subject to			
6	x+2y =	=B1+2*B2	=	4

The Solver can be called and set up as



The resulting solution is

	A	B	C	D
1	x	2		
2	y	1		
3	Minimize			
4	f(x,y)	5		
5	Subject to			
6	x+2y =	4	=	4

15.8 This problem can be solved with a variety of software tools.

Excel: An Excel spreadsheet can be set up to solve the problem as

	A	B
1	x	0
2	y	0
3	Maximize	
4	f(x,y)	0

The formulas are

	A	B
1	x	0
2	y	0
3	Maximize	
4	f(x,y)	=2.25*B1*B2+1.75*B2-1.5*B1^2-2*B2^2

The Solver can be called and set up as



The resulting solution is

	A	B
1	x	0.567568
2	y	0.756757
3	Maximize	
4	f(x,y)	0.662162

MATLAB: Set up an M-file to hold the negative of the function

```
function f=fxy(x)
f = -(2.25*x(1)*x(2)+1.75*x(2)-1.5*x(1)^2-2*x(2)^2);
```

Then, the MATLAB function `fminsearch` can be used to determine the maximum:

```
>> x=fminsearch(@fxy, [0,0])
x =
    0.5676    0.7568
>> fopt=-fxy(x)
fopt =
    0.6622
```

15.9 This problem can be solved with a variety of software tools.

Excel: An Excel spreadsheet can be set up to solve the problem as

	A	B
1	x	0
2	y	0
3	Maximize	
4	f(x,y)	0

The formulas are

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

	A	B
1	x	0
2	y	0
3	Maximize	
4	f(x,y)	=4*B1+2*B2+B1^2-2*B1^4+2*B1*B2-3*B2^2

The Solver can be called and set up as



The resulting solution is

	A	B
1	x	0.96758
2	y	0.65586
3	Maximize	
4	f(x,y)	4.344006

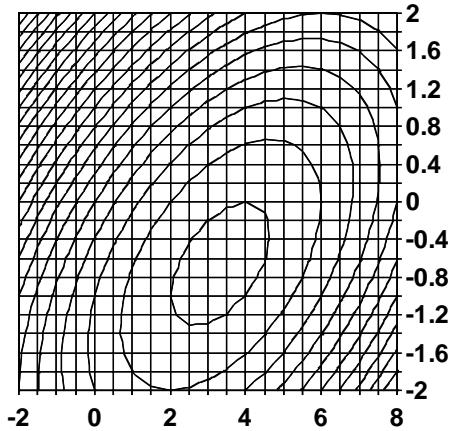
MATLAB: Set up an M-file to hold the negative of the function

```
function f=fxy(x)
f = -(4*x(1)+2*x(2)+x(1)^2-2*x(1)^4+2*x(1)*x(2)-3*x(2)^2);
```

Then, the MATLAB function `fminsearch` can be used to determine the maximum:

```
>> x=fminsearch(@fxy, [1,1])
x =
    0.9676    0.6559
>> fopt=-fxy(x)
fopt =
    4.3440
```

- 15.10 (a)** This problem can be solved graphically by using a software package to generate a contour plot of the function. For example, the following plot can be developed with Excel. As can be seen, a minimum occurs at approximately $x = 3.3$ and $y = -0.7$.



(b) We can use a software package like MATLAB to determine the minimum by first setting up an M-file to hold the function as

```
function f=fxy(x)
f = -8*x(1)+x(1)^2+12*x(2)+4*x(2)^2-2*x(1)*x(2);
```

Then, the MATLAB function `fminsearch` can be used to determine the location of the minimum as:

```
>> x=fminsearch(@fxy, [0, 0])
x =
    3.3333   -0.6666
```

Thus, $x = 3.3333$ and $y = -0.6666$.

(c) A software package like MATLAB can then be used to evaluate the function value at the minimum as in

```
>> fopt=fxy(x)
fopt =
-17.3333
```

(d) We can verify that this is a minimum as follows

$$\frac{\partial^2 f}{\partial x^2} = 2 \quad \frac{\partial^2 f}{\partial y^2} = 8 \quad \frac{\partial}{\partial y} \left(\frac{\partial f}{\partial x} \right) = \frac{\partial}{\partial x} \left(\frac{\partial f}{\partial y} \right) = -2$$

$$H = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

$$|H| = 2 \times 8 - (-2)(-2) = 12$$

Therefore the result is a minimum because $|H| > 0$ and $\partial^2 f / \partial x^2 > 0$.

15.11 The volume of a right circular cone can be computed as

$$V = \frac{\pi r^2 h}{3}$$

where r = the radius and h = the height. The area of the cone's side is computed as

$$A_s = \pi r s$$

where s = the length of the side which can be computed as

$$s = \sqrt{r^2 + h^2}$$

The area of the circular cover is computed as

$$A_c = \pi r^2$$

(a) Therefore, the optimization problem with no side slope constraint can be formulated as

$$\text{minimize } C = 100V + 50A_s + 25A_c$$

subject to

$$V \geq 50$$

A solution can be generated in a number of different ways. For example, using Excel

	A	B	C	D	E
1	Decision variables:				
2	rad	1			
3	h	1			
4					
5	Computed values:				
6	s	1.414213562			
7	slope	0.785398163 radians	45	degrees	
8	Side area	4.442882938			
9	Lid area	3.141592654			
10					
11	Constraints:				
12	Volume	1.047197551	>=	50	
13					
14					
15	Objective function:				
16	Area cost	\$ 50.00			
17	Volume cost	\$ 100.00			
18	Lid cost	\$ 25.00			
19					
20	Total cost	\$ 405.40			

The underlying formulas can be displayed as

	A	B	C	D	E
1	Decision variables:				
2	rad	1			
3	h	1			
4					
5	Computed values:				
6	s	=SQRT(B2^2+B3^2)			
7	slope	=ATAN(B3/B2)	radians	=B7*180/PI()	degrees
8	Side area	=PI()*B2*B6			
9	Lid area	=PI()*B2^2			
10					
11	Constraints:				
12	Volume	=PI()*B2^2*B3/3	>=	50	
13					
14					
15	Objective function:				
16	Area cost	50			
17	Volume cost	100			
18	Lid cost	25			
19					
20	Total cost	=B16*B8+B17*B12+B18*B9			

The Solver can be implemented as



The result is

	A	B	C	D	E
1	Decision variables:				
2	rad	2.844611637			
3	h	5.900589766			
4					
5	Computed values:				
6	s	6.550478986			
7	slope	1.121579609	radians	64.26178	degrees
8	Side area	58.5390827			
9	Lid area	25.4211877			
10					
11	Constraints:				
12	Volume	50 >=		50	
13					
14					
15	Objective function:				
16	Area cost	\$ 50.00			
17	Volume cost	\$ 100.00			
18	Lid cost	\$ 25.00			
19					
20	Total cost	\$ 8,562.48			

(b) The optimization problem with the side slope constraint can be formulated as

$$\text{minimize } C = 100V + 50A_s + 25A_c$$

subject to

$$V \geq 50$$

$$\frac{h}{r} \leq 1$$

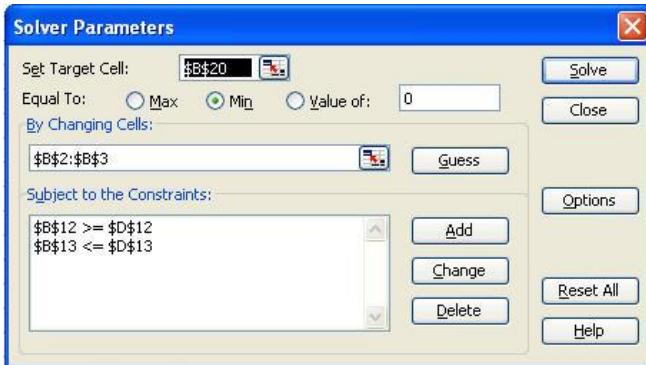
A solution can be generated in a number of different ways. For example, using Excel

	A	B	C	D	E
1	Decision variables:				
2	rad		1		
3	h		1		
4					
5	Computed values:				
6	s	1.414213562			
7	slope	0.785398163 radians		45 degrees	
8	Side area	4.442882938			
9	Lid area	3.141592654			
10					
11	Constraints:				
12	Volume	1.047197551	>=	50	
13	slope		45	<=	45
14					
15	Objective function:				
16	Area cost	\$ 50.00			
17	Volume cost	\$ 100.00			
18	Lid cost	\$ 25.00			
19					
20	Total cost	\$ 405.40			

The underlying formulas can be displayed as

	A	B	C	D	E
1	Decision variables:				
2	rad	1			
3	h	1			
4					
5	Computed values:				
6	s	=SQRT(B2^2+B3^2)			
7	slope	=ATAN(B3/B2)	radians	=B7*180/PI()	degrees
8	Side area	=PI()*B2*B6			
9	Lid area	=PI()*B2^2			
10					
11	Constraints:				
12	Volume	=PI()*B2^2*B3/3	>=	50	
13	slope	=D7	<=	45	
14					
15	Objective function:				
16	Area cost	50			
17	Volume cost	100			
18	Lid cost	25			
19					
20	Total cost	=B16*B8+B17*B12+B18*B9			

The Solver can be implemented as



The result is

	A	B	C	D	E
1	Decision variables:				
2	rad	3.627831676			
3	h	3.627831676			
4					
5	Computed values:				
6	s	5.130528758			
7	slope	0.785398163 radians	45	degrees	
8	Side area	58.47350507			
9	Lid area	41.34701196			
10					
11	Constraints:				
12	Volume	49.99999999	>=	50	
13	slope	45	<=	45	
14					
15	Objective function:				
16	Area cost	\$ 50.00			
17	Volume cost	\$ 100.00			
18	Lid cost	\$ 25.00			
19					
20	Total cost	\$ 8,957.35			

15.12 Assuming that the amounts of the two-door and four-door models are x_1 and x_2 , respectively, the linear programming problem can be formulated as

$$\text{Maximize: } z = 13,500x_1 + 15,000x_2$$

subject to

$$\begin{aligned} 15x_1 + 20x_2 &\leq 8,000 \\ 700x_1 + 500x_2 &\leq 240,000 \\ x_1 &\leq 400 \\ x_2 &\leq 350 \\ x_1, x_2 &\geq 0 \end{aligned}$$

(a) To solve graphically, the constraints can be reformulated as the following straight lines

$$x_2 = 400 - 0.75x_1$$

$$x_2 = 480 - 1.4x_1$$

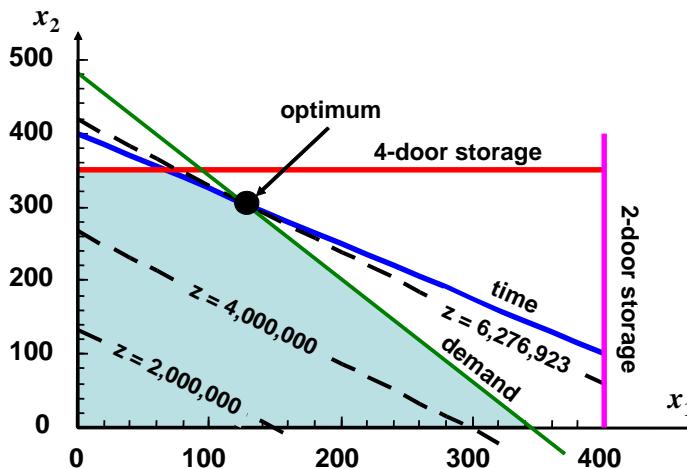
$$x_1 = 400$$

$$x_2 = 350$$

The objective function can be reformulated as

$$x_2 = (1/15,000)z - 0.9x_1$$

The constraint lines can be plotted on the x_1 - x_2 plane to define the feasible space. Then the objective function line can be superimposed for various values of z until it reaches the boundary. The result is $z \approx \$6,276,923$ with $x_1 \approx 123.08$ and $x_2 \approx 307.69$.



(b) The solution can be generated with Excel as in the following worksheet

	A	B	C	D	E
1		x_1	x_2	total	constraint
2 amount		0	0		
3 time		15	20	0	8000
4 demand		700	500	0	240000
5 Storage		1		0	400
6 Storage			1	0	350
7 profit		13500	15000	0	

The underlying formulas can be displayed as

	A	B	C	D	E
1		x_1	x_2	total	constraint
2 amount	122	308			
3 time	15	20	=B3*B\$2+C3*C\$2	8000	
4 demand	700	500	=B4*B\$2+C4*C\$2	240000	
5 Storage	1		=B5*B\$2+C5*C\$2	400	
6 Storage		1	=B6*B\$2+C6*C\$2	350	
7 profit	13500	15000	=B7*B\$2+C7*C\$2		

The Solver can be implemented as

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.



Notice how, along with the other constraints, we have specified that the decision variables must be integers. The result of running Solver is

	A	B	C	D	E
1		x_1	x_2	total	constraint
2	amount	122	308		
3	time	15	20	7990	8000
4	demand	700	500	239400	240000
5	Storage	1		122	400
6	Storage		1	308	350
7	profit	13500	15000	6267000	

Thus, because we have constrained the decision variables to be integers, the maximum profit is slightly smaller than that obtained graphically in part (a).

15.13 (a) First, we define the decision variables as

$$\begin{aligned}x_1 &= \text{number of clubs produced} \\x_2 &= \text{number of axes produced}\end{aligned}$$

The damages can be parameterized as

$$\begin{aligned}\text{damage/club} &= 2(0.45) + 1(0.65) = 1.55 \text{ main equivalents} \\ \text{damage/axe} &= 2(0.70) + 1(0.35) = 1.75 \text{ main equivalents}\end{aligned}$$

The linear programming problem can then be formulated as

$$\text{maximize } Z = 1.55x_1 + 1.75x_2$$

subject to

$$\begin{aligned}5.1x_1 + 3.2x_2 &\leq 240 && (\text{materials}) \\ 2.1x_1 + 4.3x_2 &\leq 200 && (\text{time}) \\ x_1, x_2 &\geq 0 && (\text{positivity})\end{aligned}$$

(b) and (c) To solve graphically, the constraints can be reformulated as the following straight lines

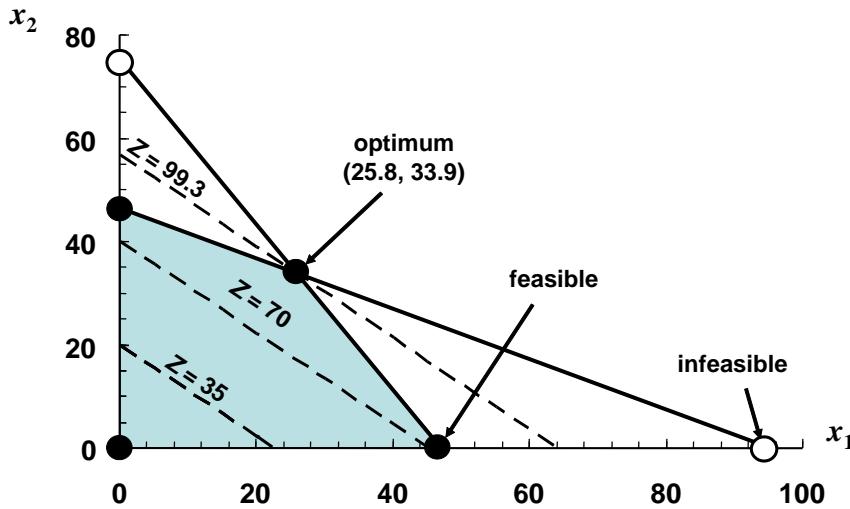
$$x_2 = 75 - 1.59375 x_1$$

$$x_2 = 46.51163 - 0.488372 x_1$$

The objective function can be reformulated as

$$x_2 = (1/1.75)Z - 0.885714 x_1$$

The constraint lines can be plotted on the x_1 - x_2 plane to define the feasible space. Then the objective function line can be superimposed for various values of Z until it reaches the boundary. The result is $Z \cong 99.3$ with $x_1 \cong 25.8$ and $x_2 \cong 33.9$.



(d) The solution can be generated with Excel as in the following worksheet

	A	B	C	D	E
1		club	axe	value	
2	kills	0.45	0.7	2	
3	maims	0.65	0.35	1	
4					
5		x1	x2	total	constraint
6	quantity	0	0		
7	materials	5.1	3.2	0	240
8	time	2.1	4.3	0	200
9					
10	damage	1.55	1.75	0	

The underlying formulas can be displayed as

	A	B	C	D	E
1		club	axe	value	
2	kills	0.45	0.7	2	
3	maims	0.65	0.35	1	
4					
5		x1	x2	total	constraint
6	quantity	25	34		
7	materials	5.1	3.2	=B7*B6+C7*C6	240
8	time	2.1	4.3	=B8*B6+C8*C6	200
9					
10	damage	=D2*B2+D3*B3	=D2*C2+D3*C3	=B10*B6+C10*C6	

The Solver can be implemented as



Notice how, along with the other constraints, we have specified that the decision variables must be integers. The result of running Solver is

	A	B	C	D	E
1		club	axe	value	
2	kills	0.45	0.7	2	
3	maims	0.65	0.35	1	
4					
5		x1	x2	total	constraint
6	quantity	25	34		
7	materials	5.1	3.2	236.3	240
8	time	2.1	4.3	198.7	200
9					
10	damage	1.55	1.75	98.25	

Thus, because we have constrained the decision variables to be integers, the maximum damage is slightly smaller than that obtained graphically in part (c).

CHAPTER 16

16.1 The area and volume can be computed as

$$A = \pi r^2 + 2\pi rh \quad (1)$$

$$V = \pi r^2 h \quad (2)$$

An Excel spreadsheet can be set up to solve the problem as

	A	B
1	radius	1
2	height	1
3		
4	volume	3.141593
5	desired volume	0.5
6		
7	side area	6.283185
8	bottom area	3.141593
9	total area	9.424778

The formulas are

	A	B
1	radius	1
2	height	1
3		
4	volume	=PI()*B1^2*B2
5	desired volume	0.5
6		
7	side area	=2*PI()*B1*B2
8	bottom area	=PI()*B1^2
9	total area	=SUM(B7:B8)

The Solver can be called and set up as



The resulting solution is

	A	B
1	radius	0.542187
2	height	0.541403
3		
4	volume	0.499999
5	desired volume	0.5
6		
7	side area	1.844378
8	bottom area	0.923525
9	total area	2.767902

Thus, the Solver says that the optimal cylindrical container is one where the radius equals the height. For the case of the desired $V = 0.5 \text{ m}^3$, the dimensions are $r = h = 0.542 \text{ m}$.

The general result of $r = h$ can be verified using calculus as follows. First, we can solve the volume equation for h as

$$h = \frac{V}{\pi r^2} \quad (3)$$

This can be substituted into the area equation to give

$$A = \pi r^2 + 2\pi r \frac{V}{\pi r^2} = \pi r^2 + \frac{2V}{r}$$

We can differentiate this equation with respect to r to yield

$$\frac{dA}{dr} = 2\pi r - \frac{2V}{r^2}$$

which can be set equal to zero and solved for

$$r = \sqrt[3]{\frac{V}{\pi}}$$

This result can then be substituted into Eq. 3 which can be solved for

$$h = \sqrt[3]{\frac{V}{\pi}}$$

Thus, we prove that the optimal container has $r = h = (V/\pi)^{1/3}$. For our desired volume of 0.5 m^3 , this means that $r = h = (0.5/\pi)^{1/3} = 0.541926 \text{ m}$, which confirms the result obtained numerically with the Excel Solver.

16.2 (a) The area and volume can be computed as

$$A = \pi r^2 + \pi r \sqrt{r^2 + h^2} \quad (1)$$

$$V = \frac{\pi r^2 h}{3} \quad (2)$$

An Excel spreadsheet can be set up to solve the problem as

	A	B
1	radius	1
2	height	1
3		
4	volume	1.047198
5	desired volume	0.5
6		
7	top area	3.141593
8	side area	4.442883
9	total area	7.584476

The formulas are

	A	B
1	radius	1
2	height	1
3		
4	volume	=PI()*B1^2*B2/3
5	desired volume	0.5
6		
7	top area	=PI()*B1^2
8	side area	=PI()*B1*SQRT(B1^2+B2^2)
9	total area	=B7+B8

The Solver can be called and set up as



The resulting solution is

	A	B
1	radius	0.552892
2	height	1.561923
3		
4	volume	0.5
5	desired volume	0.5
6		
7	top area	0.960353
8	side area	2.877962
9	total area	3.838315

(b) For this case, the area and volume can be computed as

$$A = \pi r \sqrt{r^2 + h^2}$$

$$V = \frac{\pi r^2 h}{3}$$

An Excel spreadsheet can be set up to solve the problem in a similar fashion to part (a) with the result: $r = 0.6964$ m and $h = 0.9844$ m.

16.3 This problem can be solved in a number of different ways. For example, using the golden section search, the result is

i	c_l	$g(c_l)$	c_2	$g(c_2)$	c_1	$g(c_1)$	c_u	$g(c_u)$	d	c_{opt}	ε_a
1	0.0000	0.0000	3.8197	0.2330	6.1803	0.1310	10.0000	0.0641	6.1803	3.8197	100.00%
2	0.0000	0.0000	2.3607	0.3350	3.8197	0.2330	6.1803	0.1310	3.8197	2.3607	100.00%
3	0.0000	0.0000	1.4590	0.3686	2.3607	0.3350	3.8197	0.2330	2.3607	1.4590	100.00%
4	0.0000	0.0000	0.9017	0.3174	1.4590	0.3686	2.3607	0.3350	1.4590	1.4590	61.80%
5	0.9017	0.3174	1.4590	0.3686	1.8034	0.3655	2.3607	0.3350	0.9017	1.4590	38.20%
6	0.9017	0.3174	1.2461	0.3593	1.4590	0.3686	1.8034	0.3655	0.5573	1.4590	23.61%
7	1.2461	0.3593	1.4590	0.3686	1.5905	0.3696	1.8034	0.3655	0.3444	1.5905	13.38%
8	1.4590	0.3686	1.5905	0.3696	1.6718	0.3688	1.8034	0.3655	0.2129	1.5905	8.27%
9	1.4590	0.3686	1.5403	0.3696	1.5905	0.3696	1.6718	0.3688	0.1316	1.5905	5.11%
10	1.5403	0.3696	1.5905	0.3696	1.6216	0.3694	1.6718	0.3688	0.0813	1.5905	3.16%
11	1.5403	0.3696	1.5713	0.3696	1.5905	0.3696	1.6216	0.3694	0.0502	1.5713	1.98%
12	1.5403	0.3696	1.5595	0.3696	1.5713	0.3696	1.5905	0.3696	0.0311	1.5713	1.22%
13	1.5595	0.3696	1.5713	0.3696	1.5787	0.3696	1.5905	0.3696	0.0192	1.5713	0.75%

Thus, after 13 iterations, the method is converging on the true value of $c = 1.5679$ which corresponds to a maximum specific growth rate of $g = 0.36963$.

16.4 (a) The LP formulation is given by

$$\text{Maximize } C = 30X + 30Y + 35Z \quad \{ \text{Maximize profit} \}$$

subject to

$$6X + 4Y + 12Z \leq 2500 \quad \{ \text{Raw chemical constraint} \}$$

$$0.05X + 0.1Y + 0.2Z \leq 55 \quad \{ \text{Time constraint} \}$$

$$\begin{array}{l} X + Y + Z \leq 450 \\ X, Y, Z \geq 0 \end{array}$$

{Storage constraint}
 {Positivity constraints}

(b) The simplex tableau for the problem can be set up and solved as

Basis	C	X	Y	Z	S1	S2	S3	Solution	Intercept
P	1	-30	-30	-35	0	0	0	0	0
S1	0	6	4	12	1	0	0	2500	208.3333
S2	0	0.05	0.1	0.2	0	1	0	55	275
S3	0	1	1	1	0	0	1	450	450

Basis	C	X	Y	Z	S1	S2	S3	Solution	Intercept
P	1	-12.5	-18.3333	0	2.91667	0	0	7291.667	
Z	0	0.5	0.33333	1	0.08333	0	0	208.3333	625
S2	0	-0.05	0.03333	0	-0.0167	1	0	13.33333	400
S3	0	0.5	0.66667	0	-0.0833	0	1	241.6667	362.5

Basis	C	X	Y	Z	S1	S2	S3	Solution	Intercept
P	1	1.25	0	0	0.625	0	27.5	13937.5	
Z	0	0.25	0	1	0.125	0	-0.5	87.5	700
S2	0	-0.075	0	0	-0.0125	1	-0.05	1.25	-100
Y	0	0.75	1	0	-0.125	0	1.5	362.5	-2900

(c) An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E	F
1		Product 1	Product 2	Product 3	Total	Constraint
2	amount	0	0	0		
3	material	6	4	12	0	2500
4	time	0.05	0.1	0.2	0	55
5	storage	1	1	1	0	450
6	profit	30	30	35	0	

The formulas are

	A	B	C	D	E	F
1		Product 1	Product 2	Product 3	Total	Constraint
2	amount	0	0	0		
3	material	6	4	12	=B3*B\$2+C3*C\$2+D3*D\$2	2500
4	time	0.05	0.1	0.2	=B4*B\$2+C4*C\$2+D4*D\$2	55
5	storage	1	1	1	=B5*B\$2+C5*C\$2+D5*D\$2	450
6	profit	30	30	35	=B6*B\$2+C6*C\$2+D6*D\$2	

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E	F
1		Product 1	Product 2	Product 3	Total	Constraint
2	amount	0	362.5	87.5		
3	material	6	4	12	2500	2500
4	time	0.05	0.1	0.2	53.75	55
5	storage	1	1	1	450	450
6	profit	30	30	35	13937.5	

In addition, a sensitivity report can be generated as

	A	B	C	D	E	F	G	H
1	Microsoft Excel 11.0 Sensitivity Report							
2	Worksheet: [Prob1604.xls]Sheet1							
3	Report Created: 6/29/2005 8:38:14 AM							
4								
5								
6	Adjustable Cells							
7								
8	Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease	
9	\$B\$2	amount Product 1	0	-1.25	30	1.25	1E+30	
10	\$C\$2	amount Product 2	362.5	0	30	5	1.6666666667	
11	\$D\$2	amount Product 3	87.5	0	35	55	5	
12								
13	Constraints							
14								
15	Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease	
16	\$E\$3	material Total	2500	0.625	2500	100	700	
17	\$E\$4	time Total	53.75	0	55	1E+30	1.25	
18	\$E\$5	storage Total	450	27.5	450	25	241.66666667	

- (d) The high shadow price for storage from the sensitivity analysis from (c) suggests that increasing storage will result in the best increase in profit.

16.5 An LP formulation for this problem can be set up as

$$\text{Maximize } P = 2000Z_1 - 75Z_2 + 250Z_3 - 300W \quad \{\text{Maximize profit}\}$$

subject to

$$\begin{aligned} Z_1 + Z_2 &\leq 7500 && \{X \text{ material constraint}\} \\ 2.5Z_1 + Z_3 &\leq 12,500 && \{Y \text{ material constraint}\} \\ Z_1 - Z_2 - Z_3 - W &= 0 && \{\text{Waste constraint}\} \end{aligned}$$

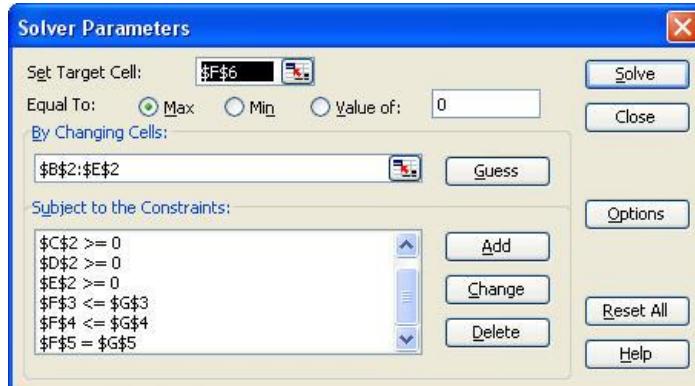
An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E	F	G
1		Z1	Z2	Z3	W	total	constraint
2	amount	0	0	0	0		
3	amount X	1	1	0	0	0	7500
4	amount Y	2.5	0	1	0	0	12500
5	amount W	1	-1	-1	-1	0	
6	profit	2000	-75	250	-300	0	

The formulas are

	A	B	C	D	E	F	G
1		Z1	Z2	Z3	W	total	constraint
2	amount	0	0	0	0		
3	amount X	1	1	0	0	=B3*B\$2+C3*C\$2+D3*D\$2+E3*E\$2	7500
4	amount Y	2.5	0	1	0	=B4*B\$2+C4*C\$2+D4*D\$2+E4*E\$2	12500
5	amount W	1	-1	-1	-1	=B5*B\$2+C5*C\$2+D5*D\$2+E5*E\$2	
6	profit	2000	-75	250	-300	=B6*B\$2+C6*C\$2+D6*D\$2+E6*E\$2	

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E	F	G
1		Z1	Z2	Z3	W	total	constraint
2	amount	5000	2500	0	2500		
3	amount X	1	1	0	0	7500	7500
4	amount Y	2.5	0	1	0	12500	12500
5	amount W	1	-1	-1	-1	0	
6	profit	2000	-75	250	-300	9062500	

This is an interesting result which might seem counterintuitive at first. Notice that we create some of the unprofitable Z_2 while producing none of the profitable Z_3 . This occurred because

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

we used up all of Y in producing the highly profitable Z_1 . Thus, there was none left to produce Z_3 .

16.6 Substitute $x_B = 1 - x_T$ into the pressure equation,

$$(1 - x_T)P_{sat_B} + x_T P_{sat_T} = P$$

and solve for x_T ,

$$x_T = \frac{P - P_{sat_B}}{P_{sat_T} - P_{sat_B}} \quad (1)$$

where the partial pressures are computed as

$$P_{sat_B} = 10^{\left(6.905 - \frac{1211}{T+221}\right)}$$

$$P_{sat_T} = 10^{\left(6.953 - \frac{1344}{T+219}\right)}$$

The solution then consists of maximizing Eq. 1 by varying T subject to the constraint that $0 \leq x_T \leq 1$. The Excel Solver can be used to obtain the solution. Here is how the worksheet can be set up:

	A	B	C	D	E	F	G	H
1	Prob16.6							
2								
3	T	0						
4								
5	P	800						
6								
7	PsatB	26.62944	<-----	=10^(6.905-1211/(T+221))				
8	PsatT	6.546568	<-----	=10^(6.953-1344/(T+219))				
9								
10	xT	-38.509	<-----	= (P-PsatB)/(PsatT-PsatB)				
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								

Solver Parameters

Set Target Cell: \$B\$10

Equal To: Max Min Value of: 0

By Changing Cells: \$B\$3

Subject to the Constraints:

- \$B\$10 <= 1
- \$B\$10 >= 0

Buttons: Solve, Close, Options, Guess, Add, Change, Delete, Reset All, Help

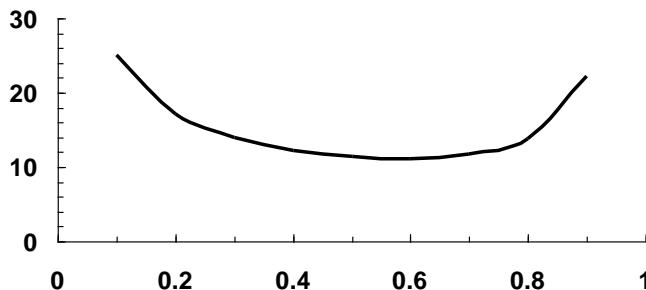
The result is $T = 112.8592$ as shown below:

	A	B	C	D	E	F
1	Prob16.6					
2						
3	T	112.8592				
4						
5	P	800				
6						
7	PsatB	1895.496	<-----	= $10^{(6.905-1211/(T+221))}$		
8	PsatT	800.0004	<-----	= $10^{(6.953-1344/(T+219))}$		
9						
10	xT	1	<-----	= $(P-PsatB)/(PsatT-PsatB)$		

16.7 This is a straightforward problem of varying x_A in order to minimize

$$f(x_A) = \left(\frac{1}{(1-x_A)^2} \right)^{0.6} + 6 \left(\frac{1}{x_A} \right)^{0.6}$$

First, the function can be plotted versus x_A



The result indicates a minimum between 0.5 and 0.6. Using golden section search or a package like Excel or MATLAB yields a minimum of 0.587683.

16.8 This is a case of constrained nonlinear optimization. The conversion factors range between 0 and 1. In addition, the cost function can not be evaluated for certain combinations of X_{A1} and X_{A2} . The problem is the second term,

$$\left(\frac{1 - \frac{x_{A1}}{x_{A2}}}{(1 - x_{A2})^2} \right)^{0.6}$$

If $x_{A1} > x_{A2}$, the numerator will be negative and the term cannot be evaluated.

Excel Solver can be used to solve the problem:

The screenshot shows a Microsoft Excel spreadsheet with a Solver Parameters dialog box overlaid. The spreadsheet has columns A through I and rows 1 through 20. Row 1 contains 'XA1' and '0.5'. Row 2 contains 'XA2' and '0.6'. Row 4 contains 'Cost' and '11.23608'. The Solver Parameters dialog box is open, with the target cell set to '\$B\$4', the objective to minimize ('Min'), and the changing cells set to '\$B\$1:\$B\$2'. The constraints listed are \$XA1 <= 1, \$XA1 <= \$XA2, \$XA1 >= 0, \$XA2 <= 1, and \$XA2 >= 0.

	A	B	C	D	E	F	G	H	I
1	XA1	0.5							
2	XA2	0.6							
4	Cost	11.23608							
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									

The result is

	A	B
1	XA1	0.368949
2	XA2	0.627265
3		
4	Cost	11.12039

16.9 This problem can be set up on Excel and the answer generated with Solver. Note that we have named the cells with the labels in the adjacent left columns.

The screenshot shows a Microsoft Excel spreadsheet with a Solver Parameters dialog box overlaid. The spreadsheet has columns A through L and rows 1 through 16. Row 1 contains 'Prob. 16.9'. Rows 3 through 6 contain labels K, BO, CA, and CC respectively. Rows 8 through 11 contain labels AD, A, B, and Kcalc with their corresponding values. Row 16 contains 'Profit' with a value of 700. The Solver Parameters dialog box is open, with the target cell set to 'Profit', the objective to maximize ('Max'), and the changing cells set to '\$B\$8:\$B\$9'. The constraints listed are \$A >= 0, \$A0 >= 0, \$B >= 0, \$C >= 0, and \$K = \$Kcalc.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Prob. 16.9											
2												
3	K	2										
4	BO	100										
5	CA	-1										
6	CC	10										
7												
8	AD	200										
9	C	90										
10												
11	A	20	<-----	=AD-2*C								
12	B	10	<-----	=BO-C								
13												
14	Kcalc	0.0225	<-----	=C/(A^2*B)								
15												
16	Profit	700	<-----	=CA*AD+CC*C								

The solution is

	A	B	C	D
1	Prob. 16.9			
2				
3	K	2		
4	B0	100		
5	CA	-1		
6	CC	10		
7				
8	AD	208.085		
9	C	99.41872		
10				
11	A	9.247574 <-----	=AD-2*C_	
12	B	0.581276 <-----	=B0-C_	
13				
14	Kcalc	2 <-----	=C_/(A^2*B)	
15				
16	Profit	786.1022 <-----	=CA*AD+CC*C_	

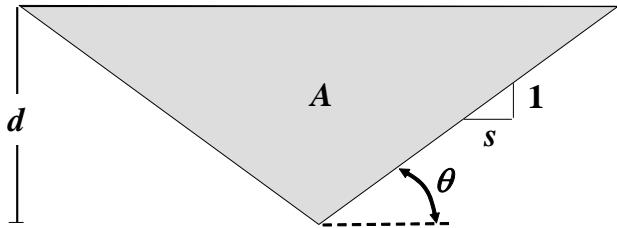
16.10 The problem can be set up in Excel Solver. Note that we have named the cells with the labels in the adjacent left columns.

	A	B	C	D	E	F	G	H	I	J
1	Prob. 16.10									
2										
3	Unitcost1	\$ 0.50	\$/L							
4	Unitcost2	\$ 1.00	\$/L							
5	Unitcost3	\$ 1.20	\$/L							
6										
7	conc1	135	mg/L							
8	conc2	100	mg/L							
9	conc3	75	mg/L							
10										
11	Supply1	2000000	L/d							
12	Supply2	1000000	L/d							
13	Supply3	500000	L/d							
14										
15	Flow1	300000	L/d							
16	Flow2	300000	L/d							
17	Flow3	300000	L/d							
18										
19	Flowt	900000	L/d	=Flow1+Flow2+Flow3						
20										
21	Flowr	1000000	L/d							
22										
23	ConcBulk	103.3333333	mg/L	=(Flow1*conc1+Flow2*conc2+Flow3*conc3)/Flowt						
24										
25	Concr	100	mg/L							
26										
27	Total cost	\$ 810,000.00		=Unitcost1*Flow1+Unitcost2*Flow2+Unitcost3*Flow3						

The solution is

15	Flow1	357143	L/d
16	Flow2	142857	L/d
17	Flow3	500000	L/d
18			
19	Flowt	1000000	L/d
20			
21	Flowr	1000000	L/d
22			
23	ConcBulk	100.0000001	mg/L
24			
25	Concr	100	mg/L
26			
27	Total cost	\$ 921,428.57	

16.11 Here is a diagram for this problem:



The following formulas can be developed:

$$\theta = \tan^{-1} \frac{1}{s} \quad (1)$$

$$P = 2d\sqrt{1+s^2} \quad (2)$$

$$A = sd^2 \quad (3)$$

Then the following Excel worksheet and Solver application can be set up:

	A	B	C	D
1	s	0.5		
2	d	5		
3				
4	A	12.5	<-----	=B1*B2^2
5	Agoal	50		
6				
7	P	11.18034	<-----	=2*SQRT(1+B1^2)*B2
8				
9	angle (radians)	1.107149	<-----	=ATAN(1/B1)
10	angle (degrees)	63.43495	<-----	=B9*180/PI()
11				
12				
13				
14				
15				

Solver Parameters

Set Target Cell:

Equal To: Max Min Value of:

By Changing Cells:

Subject to the Constraints:

Our goal is to minimize the wetted perimeter by varying the side slope and the depth. We apply the constraint that the computed area must equal the desired area. The result is

	A	B	C	D
1	s	1.000088		
2	d	7.070756		
3				
4	A	50	<-----	=B1*B2^2
5	Agoal	50		
6				
7	P	20	<-----	=2*SQRT(1+B1^2)*B2
8				
9	angle (radians)	0.785354	<-----	=ATAN(1/B1)
10	angle (degrees)	44.99747	<-----	=B9*180/PI()

Thus, this specific application indicates that a 45° angle yields the minimum wetted perimeter.

The verification that this result is universal can be attained inductively or deductively. The inductive approach involves trying several different desired areas in conjunction with our solver solution. As long as the desired area is greater than 0, the result for the optimal design will be 45° .

The deductive verification involves calculus. First, Eq. 3 can be solved for d and the result substituted into Eq. 2 to give

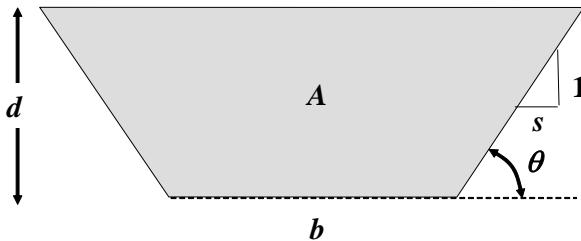
$$P = 2\sqrt{A\left(s + \frac{1}{s}\right)} \quad (4)$$

The minimum wetted perimeter should occur when the derivative of the perimeter with respect to s flattens out. That is, the slope is zero. Setting the derivative of Eq. 4 to zero yields,

$$\frac{dP}{ds} = \frac{1 - \frac{1}{s^2}}{\sqrt{s + \frac{1}{s}}} = 0 \quad (5)$$

We can see that the derivative is zero if $s = 1$. According to Eq. 1, this corresponds to $\theta = 45^\circ$. Thus, the result obtained numerically is shown to be universal.

16.12 Here is a diagram for this problem:



The following formulas can be developed:

$$\theta = \tan^{-1} \frac{1}{s} \quad (1)$$

$$P = b + 2d\sqrt{1+s^2} \quad (2)$$

$$A = (b + sd)d \quad (3)$$

Then the following Excel worksheet and Solver application can be set up:

The Solver dialog box is open, showing the following settings:

- Set Target Cell:** \$B\$8
- Equal To:** Min
- By Changing Cells:** \$B\$1:\$B\$3
- Subject to the Constraints:** \$B\$5 = \$B\$6

The spreadsheet contains the following data:

	A	B	C	D
1	b	1		
2	s	1		
3	d	1		
4				
5	A	2	<-----	=(B1+B2*B3)*B3
6	Agoal	100		
7				
8	P	3.828427	<-----	=B1+2*SQRT(1+B2^2)*B3
9				
10	angle (radians)	0.785398	<-----	=ATAN(1/B2)
11	angle (degrees)	45	<-----	=B10*180/PI()
12				
13				
14				
15				

Our goal is to minimize the wetted perimeter by varying the depth, side slope and bottom width. We apply the constraint that the computed area must equal the desired area. The result is

	A	B	C	D
1	b	8.773829		
2	s	0.577343		
3	d	7.598379		
4				
5	A	100	<-----	=(B1+B2*B3)*B3
6	Agoal	100		
7				
8	P	26.32148	<-----	=B1+2*SQRT(1+B2^2)*B3
9				
10	angle (radians)	1.047203	<-----	=ATAN(1/B2)
11	angle (degrees)	60.0003	<-----	=B10*180/PI()

Thus, this specific application indicates that a 60° angle yields the minimum wetted perimeter.

The verification of whether this result is universal can be attained inductively or deductively. The inductive approach involves trying several different desired areas in conjunction with our solver solution. As long as the desired area is greater than 0, the result for the optimal design will be 60° .

The deductive verification involves calculus. First, we can solve Eq. 3 for b and substitute the result into Eq. 2 to give,

$$P = \frac{A}{d} + d \left(2\sqrt{1+s^2} - s \right) \quad (4)$$

If both A and d are constants and s is a variable, the condition for the minimum perimeter is $dP/ds = 0$. Differentiating Eq. 4 with respect to s and setting the resulting equation to zero,

$$\frac{dP}{ds} = d \left(\frac{2s}{\sqrt{1+s^2}} - 1 \right) = 0 \quad (4)$$

Therefore, we obtain $s = 1/\sqrt{3}$. Using Eq. 1, this corresponds to $\theta = 60^\circ$.

16.13

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$A_{ends} = 2\pi r^2$$

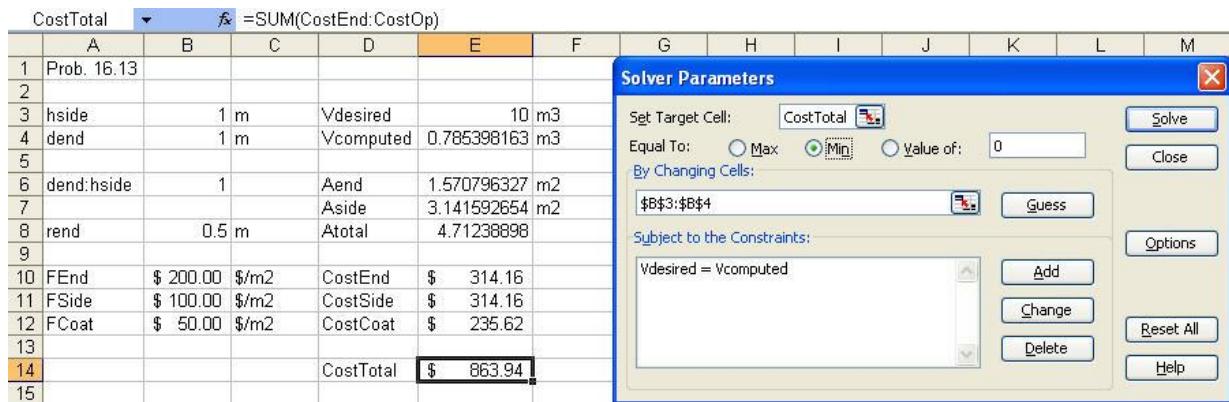
$$A_{side} = 2\pi rh$$

$$A_{total} = A_{ends} + A_{side}$$

$$V_{computed} = \pi r^2 h$$

$$Cost = F_{ends} A_{ends} + F_{side} A_{side} + F_{coating} A_{coating}$$

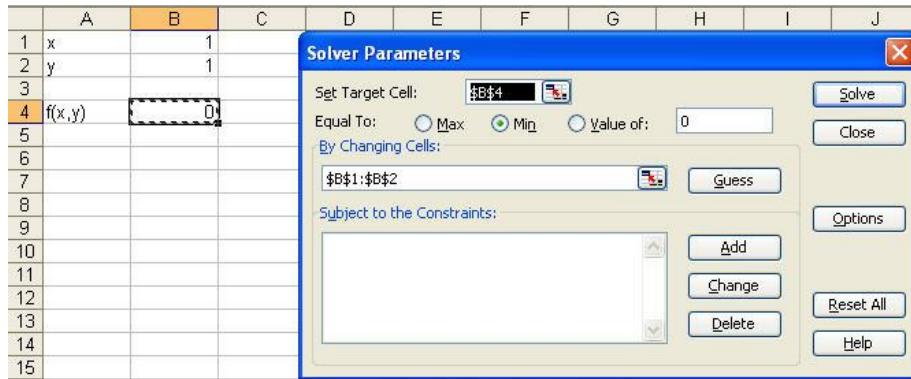
Then the following Excel worksheet and Solver application can be set up:



which results in the following solution:

	A	B	C	D	E	F
1	Prob. 16.13					
2						
3	hside	3.282282 m		Vdesired		10 m³
4	dend	1.96955 m		Vcomputed		10 m³
5						
6	dendl:hside	0.600055		Aend	6.093321721 m²	
7				Aside	20.30920276 m²	
8	rend	0.984775 m		Atotal	26.40252448	
9						
10	FEnd	\$ 200.00	\$/m²	CostEnd	\$ 1,218.66	
11	FSide	\$ 100.00	\$/m²	CostSide	\$ 2,030.92	
12	FCoat	\$ 50.00	\$/m²	CostCoat	\$ 1,320.13	
13						
14				CostTotal	\$ 4,569.71	

16.14 As shown below, Excel Solver gives: $x = 0.5$, $y = 0.8$ and $f_{min} = -0.85$.



	A	B
1	x	0.5
2	y	0.8
4	f(x,y)	-0.85

16.15 An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E
1	Parameters				
2	c1	4	d1		1
3	c2	2	d2		10
4	H	275	t1		0.1
5	P	2000	t2		1
6	E	900000			
7	rho	0.0025			
8	sigmamax	550			
9					
10	Decision variables				
11	t	0.5			
12	d	10			
13					
14	Computed quantities		goals:		
15	W	10.79922			
16	I	196.8404			
17	sigma	127.324	<	550	
18	sigmab	1471.876			
19					
20	Objective function				
21	C	63.1969			

The formulas are

	A	B	C	D	E
1	Parameters				
2	c1	4		d1	1
3	c2	2		d2	10
4	H	275		t1	0.1
5	P	2000		t2	1
6	E	900000			
7	rho	0.0025			
8	sigmamax	550			
9					
10	Decision va				
11	t	0.5			
12	d	10			
13					
14	Computed c			goals:	
15	W	=PI()*B12*B11*B4*B7			
16	I	=PI()/8*B12*B11*(B12^2+B11^2)			
17	sigma	=B5/PI()/B12/B11	<	550	
18	sigmab	=PI()*B6*B16/B4^2/B12/B11			
19					
20	Objective fu				
21	C	=B2*B15+B3*B12			

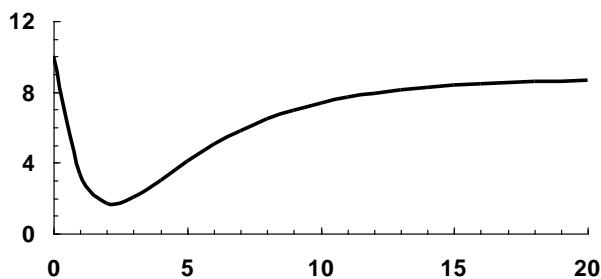
The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E
1	Parameters				
2	c1	4	d1	1	
3	c2	2	d2	10	
4	H	275	t1	0.1	
5	P	2000	t2	1	
6	E	900000			
7	rho	0.0025			
8	sigmamax	550			
9					
10	Decision variables				
11	t	0.189207			
12	d	6.117589			
13					
14	Computed quantities		goals:		
15	W	2.5			
16	I	17.02759			
17	sigma	550	<	550	
18	sigmab	550			
19					
20	Objective function				
21	C	22.23518			
22					

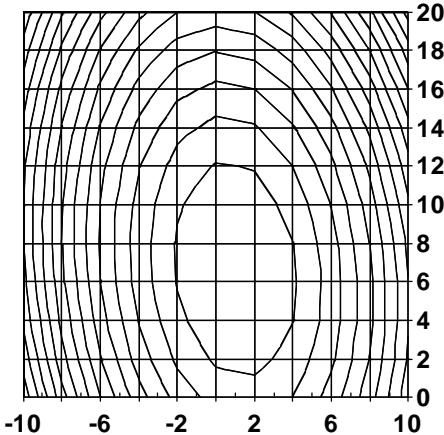
16.16 A plot of the function indicates a minimum at about $t = 2.2$.



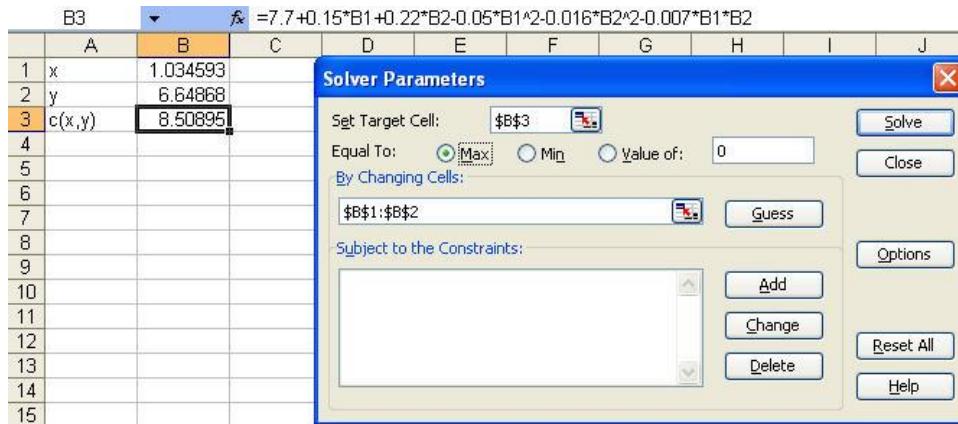
The Excel Solver can be used to determine that a minimum of $o = 1.699$ occurs at a value of $t = 2.2023$.

1	os	10									
2	kd	0.2									
3	ka	0.8									
4	ks	0.06									
5	Lo	50									
6	Sb	1									
7											
8	t	2.202314									
9	$o(t)$	1.699299	<-----	=os-kd*Lo/(kd+ks-ka)*(EXP(-ka*B8)-EXP(-(kd+ks)*B8))-Sb/ka*(1-EXP(-ka*B8))							
10											
11	Solver Parameters										
12	Set Target Cell:	\$B\$9	<input type="button" value="..."/>	Solve							
13	Equal To:	<input checked="" type="radio"/> Min	<input type="radio"/> Max	<input type="radio"/> Value of:	0						
14	By Changing Cells:	\$B\$8	<input type="button" value="..."/>	Guess							
15	Subject to the Constraints:		Add								
16			Change								
17			Delete								
18			Options								
19			Reset All								
20			Help								
21											
22											
23											
24											
25											

16.17 This problem can be solved graphically by using a software package to generate a contour plot of the function. For example, the following plot can be developed with Excel. As can be seen, a minimum occurs at approximately $x = 1$ and $y = 7$.



We can use a software package like Excel to determine the maximum precisely as $x = 1.034593$ and $y = 6.64868$.



16.18 (a) The problem consists of

$$\min P = B + 2 * H$$

Subject to

$$\frac{1}{n} BH \left(\frac{BH}{B + 2H} \right)^{2/3} S^{1/2} = Q$$

The problem can be set up and solved with the Excel Solver as in

	A	B	C	D	E	F	G	H	I	J	K	L
1	Problem 16.18											
2												
3	n	0.03										
4	S	0.0004										
5	Q	1										
6												
7	B	2.13514										
8	H	1.067614										
9	B/H	1.999917										
10												
11	Ac	2.279507 =B*H										
12	P	4.270369 =B+2*H										
13	R	0.533796 =Ac/P										
14												
15	Qmanning	1 =1/n*Ac*R_^(2/3)*S^0.5										

Solver Parameters

Set Target Cell: Equal To: Max Min Value of: 0

By Changing Cells:

Subject to the Constraints:

Qmanning = Q

Buttons: Solve, Close, Options, Guess, Add, Change, Delete, Reset All, Help

As can be seen, the result shows that the dimensions for the minimum wetted perimeter correspond to having the bottom width that is twice the length of each vertical side.

(b) Now we can redo the problem as a cost minimization:

$$\min C = 100A_c + 50P$$

Subject to

$$\frac{1}{n} BH \left(\frac{BH}{B + 2H} \right)^{2/3} S^{1/2} = Q$$

The problem can be set up and solved with the Excel Solver as in

	Cost		=100*Ac+50*P	C	D	E	F	G	H	I	J	K	L
1	Problem 16.18												
2													
3	n	0.03											
4	S	0.0004											
5	Q	1											
6													
7	B	2.135512	2.000614										
8	H	1.067429											
9	B/H	2.000614											
10													
11	Ac	2.279507 =B*H											
12	P	4.27037 =B+2*H											
13	R	0.533796 =Ac/P											
14													
15	Cost	441.4692											
16													
17	Qmanning	1 =1/n*Ac*R_^(2/3)*S^0.5											

Solver Parameters

Set Target Cell: Equal To: Max Min Value of: 0

By Changing Cells:

Subject to the Constraints:

Qmanning = Q

Buttons: Solve, Close, Options, Guess, Add, Change, Delete, Reset All, Help

Very interestingly, the result is identical to that obtained when cost was not an issue!!!

(c) The constraint can be rewritten as

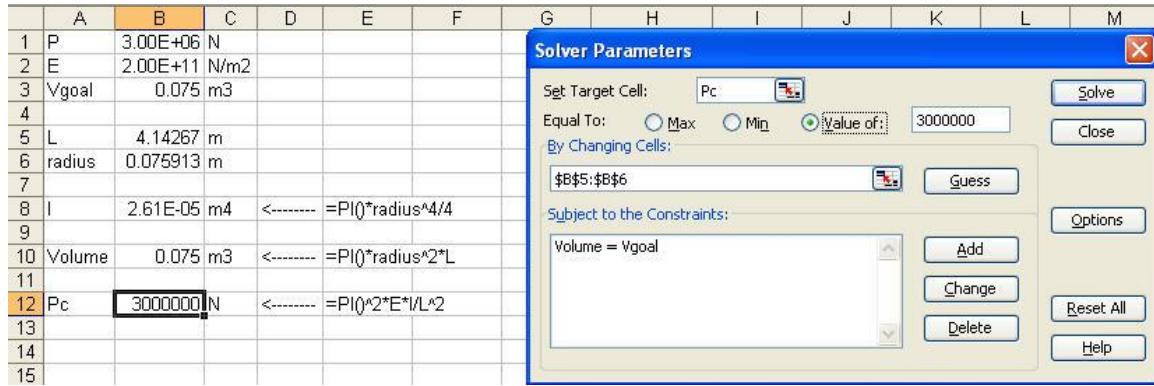
$$\frac{(BH)^{5/2}}{B + 2H} = \left(\frac{nQ}{S^{1/2}} \right)^{3/2} = \text{constant}$$

or

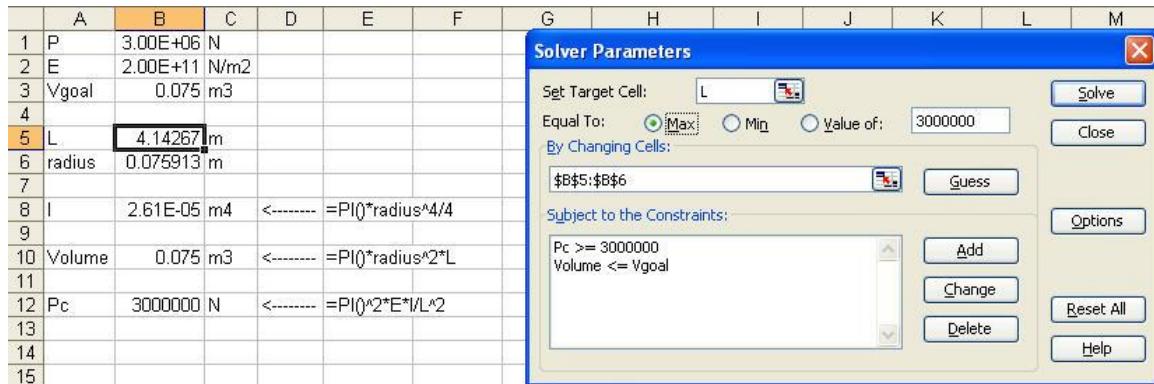
$$BH = \text{constant} \times (B + 2H)^{2/5}$$

Therefore, both A_c and P are minimized simultaneously. This is great, because the excavation costs will be proportional to the cross-sectional area. Hence, by having the bottom width twice the length of each vertical side, we will minimize both excavation and lining costs simultaneously!!!

16.19 Using Excel Solver,



An alternative solution can be developed by maximizing L subject to Volume ≤ 0.075 m³ and $P_c \geq 3,000,000$ N,



16.20 The total flow in the river: $F = 2 \times 10^6$ m³/d.

The flow into the channels:

$$f_1 + f_2 \leq 0.7F = 1.4 \times 10^6 \text{ m}^3/\text{d}$$

Minimum channel flows for navigation:

$$f_1 \geq 0.3 \times 10^6 \text{ m}^3/\text{d}$$

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$f_2 \geq 0.2 \times 10^6 \text{ m}^3/\text{d}$$

Political constraints:

$$\frac{|f_1 - f_2|}{f_1 + f_2} \leq 0.4$$

leads to

$$\begin{aligned} f_2 &\geq \frac{3}{7} f_1 \\ f_2 &\leq \frac{7}{3} f_1 \end{aligned}$$

Maintenance cost per year, $C \leq \$1.8 \times 10^6$

Channel 1: $C_1 = 1.1f_1$

Channel 2: $C_2 = 1.4f_2$

leads to

$$1.1f_1 + 1.4f_2 \leq 1.8 \times 10^6$$

Power revenue (revenue per year):

Channel 1: $r_{p1} = 4f_1$

Channel 2: $r_{p2} = 3f_2$

Irrigation revenue (revenue per year):

Channel 1: loss, $\alpha_1 = 0.3$

value/yr: $i_1 = 3.2(1 - \alpha)f_1 = 2.24f_1$

Channel 2: loss, $\alpha_2 = 0.2$

value/yr: $i_2 = 3.2(1 - \alpha)f_2 = 2.56f_2$

Net revenue = Revenue – losses

$$P = 4f_1 + 3f_2 + 2.24f_1 + 2.56f_2 - 1.1f_1 - 1.4f_2$$

$$P = 5.14f_1 + 4.16f_2$$

Therefore, the problem is formulated as

Decision variables:

f_1 : flow in channel 1

f_2 : flow in channel 2

Maximize: $P = 5.14f_1 + 4.16f_2$

Subject to

$f_1 + f_2 \leq 1.4 \times 10^6$	channel flow
$1.1f_1 + 1.4f_2 \leq 1.8 \times 10^6$	maintenance
$0.43f_1 - f_2 \leq 0$	political constraint 1
$-2.33f_1 + f_2 \leq 0$	political constraint 2
$f_1 \geq 0.3 \times 10^6$	minimum channel flow 1
$f_2 \geq 0.2 \times 10^6$	minimum channel flow 2

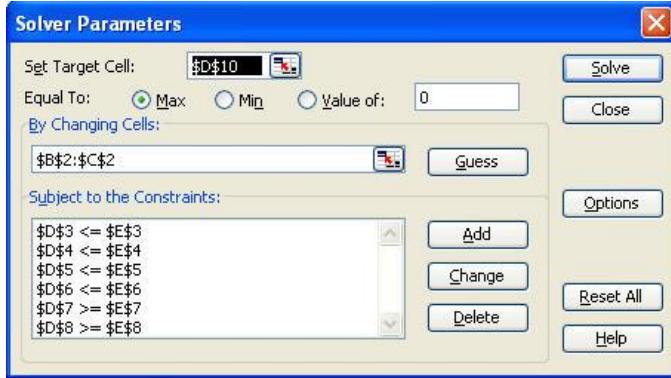
The problem can then be set up and solved with a tool such as Excel:

	A	B	C	D	E
1		Channel 1	Channel 2	total	constraint
2	Flow	0	0		
3		1	1	0	1.40E+06
4		1.1	1.4	0	1.80E+06
5		0.43	-1	0	0
6		-2.33	1	0	0
7		1		0	3.00E+05
8			1	0	2.00E+05
9					
10	Profit	5.14	4.16	0	

The cell formulas are

	A	B	C	D	E
1		Channel 1	Channel 2	total	constraint
2	Flow	0	0		
3		1	1	=B3*B\$2+C3*C\$2	1400000
4		1.1	1.4	=B4*B\$2+C4*C\$2	1800000
5		0.43	-1	=B5*B\$2+C5*C\$2	0
6		-2.33	1	=B6*B\$2+C6*C\$2	0
7		1		=B7*B\$2+C7*C\$2	300000
8			1	=B8*B\$2+C8*C\$2	200000
9					
10	Profit	5.14	4.16	=B10*B\$2+C10*C\$2	

The Excel Solver can be invoked as



The resulting solution is

	A	B	C	D	E
1		Channel 1	Channel 2	total	constraint
2	Flow	979021	420979		
3		1	1	1400000	1.40E+06
4		1.1	1.4	1666294	1.80E+06
5		0.43	-1	0	0
6		-2.33	1	-1860140	0
7		1		979021	3.00E+05
8			1	420979	2.00E+05
9					
10	Profit	5.14	4.16	6783441	

16.21 The weight of the truss is equal to

$$W = \rho(L_1 A_c + L_2 A_t + L_3 A_c)$$

where ρ = density, L_i = length of member i , A_c = cross-sectional area of compression member, and A_t = cross-sectional area of tension member. The lengths of the 3 members can be determined as $L_1 = 43.3013$, $L_2 = 50$, and $L_3 = 25$. Therefore, the solution can be formulated as a linear programming problem as

$$\text{Minimize: } W = 3.5(43.3013A_c + 50A_t + 25A_c)$$

subject to

$$A_c \geq 50$$

$$A_t \geq 43.3$$

The solution can be developed in Excel using the Solver tool,

B4 =3.5*(43.3013*Ac+50*At+25*Ac)

	A	B	C	D	E	F	G	H	I	J
1	Ac	50								
2	At	43.3								
3	weight	19530.23								
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										

16.22 The solution can be developed in Excel using the Solver tool,

B7 =1/(4*PI()*e0)*q*QQ*B6/(B6^2+rad^2)^1.5

	A	B	C	D	E	F	G	H	I	J
1	e0	8.85E-12								
2	QQ	2.00E-05								
3	q	2.00E-05								
4	rad	0.90								
5	x	0.636396								
6	F	1.70911								
7										
8										
9										
10										
11										
12										
13										
14										
15										

16.23 The problem can be formulated as

$$\text{Minimize} \quad C = 2p_1 + 10p_2 + 2$$

subject to

$$0.6p_1 + 0.4p_2 \geq 30$$

$$p_1 \leq 42$$

Using the Excel Solver:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	Individual power						Constraint						
3	p1		42				<	42					
4	p2		12										
5	Losses												
6	L1		9.6	<-----	=0.2*B3+0.1*B4								
7	L2		14.4	<-----	=0.2*B3+0.5*B4								
8	Total power		30	<-----		=	30						
9													
10	Costs												
11	F1		86	<-----	=2*B3+2								
12	F2		120	<-----	=10*B4								
13	Total cost		206	<-----	=B11+B12								
14													
15													

Solver Parameters

Set Target Cell: \$B\$13

Equal To: Max Min Value of: 0

By Changing Cells: \$B\$3:\$B\$4 Guess

Subject to the Constraints:

- \$B\$3 <= \$F\$3
- \$B\$8 >= \$F\$8

16.24 This is a trick question. Because of the presence of $(1 - s)$ in the denominator, the function will experience a division by zero at the maximum. This can be rectified by merely canceling the $(1 - s)$ terms in the numerator and denominator to give

$$T = \frac{15s}{4s^2 - 3s + 4}$$

Any of the optimizers described in this section can then be used to determine that the maximum of $T = 3$ occurs at $s = 1$.

16.25 (a) An LP formulation for this problem can be set up as

$$\text{Maximize } P = 500x_1 + 400x_2$$

subject to

$$300x_1 + 400x_2 \leq 127,000$$

$$20x_1 + 10x_2 \leq 4,270$$

$$x_1, x_2 \geq 0$$

An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Device	Capital (\$/unit)	Labor (hrs/unit)	Profit									
2	Scanner	300	20	500									
3	Printer	400	10	400									
4													
5	Devise1	88											
6	Devise2	251											
7							Constraint						
8	Capital	126800	<-----	=B2*B5+B3*B6	<	127000							
9	Labor	4270	<-----	=C2*B5+C3*B6	<	4270							
10													
11	Profit	144400	<-----	=D2*B5+D3*B6									
12													
13													
14													
15													

Solver Parameters

Set Target Cell: \$B\$11

Equal To: Max Min Value of: 0

By Changing Cells: \$B\$5:\$B\$6 Guess

Subject to the Constraints:

- \$B\$5 = integer
- \$B\$5 >= 0
- \$B\$6 = integer
- \$B\$6 >= 0
- \$B\$8 <= \$F\$8
- \$B\$9 <= \$F\$9

(b) This problem can be formulated as

$$\text{Maximize } P = 500x_1 + (400 - x_2)x_2$$

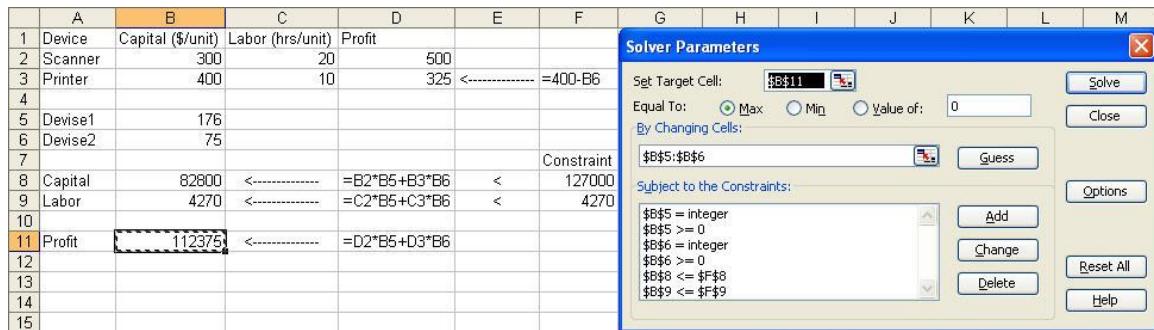
subject to

$$300x_1 + 400x_2 \leq 127,000$$

$$20x_1 + 10x_2 \leq 4,270$$

$$x_1, x_2 \geq 0$$

An Excel spreadsheet can be set up to solve the problem as



16.26 An LP formulation for this problem can be set up as

Decision variables: x_{ri} = chips produced in regular time for month i

x_{oi} = chips produced in overtime for month i

x_{si} = chips stored for month i

$$\text{Minimize } C = 100x_{r1} + 100x_{r2} + 120x_{r3} + 110x_{o1} + 120x_{o2} + 130x_{o3} + 5x_{s1} + 5x_{s2}$$

subject to

$$x_{r1} + x_{o1} - x_{s1} \geq 1,000$$

$$x_{s1} + x_{r2} + x_{o2} - x_{s2} \geq 2,500$$

$$x_{s2} + x_{r3} + x_{o3} \geq 2,200$$

$$1.5x_{r1} \leq 2,400$$

$$1.5x_{r2} \leq 2,400$$

$$1.5x_{r3} \leq 2,400$$

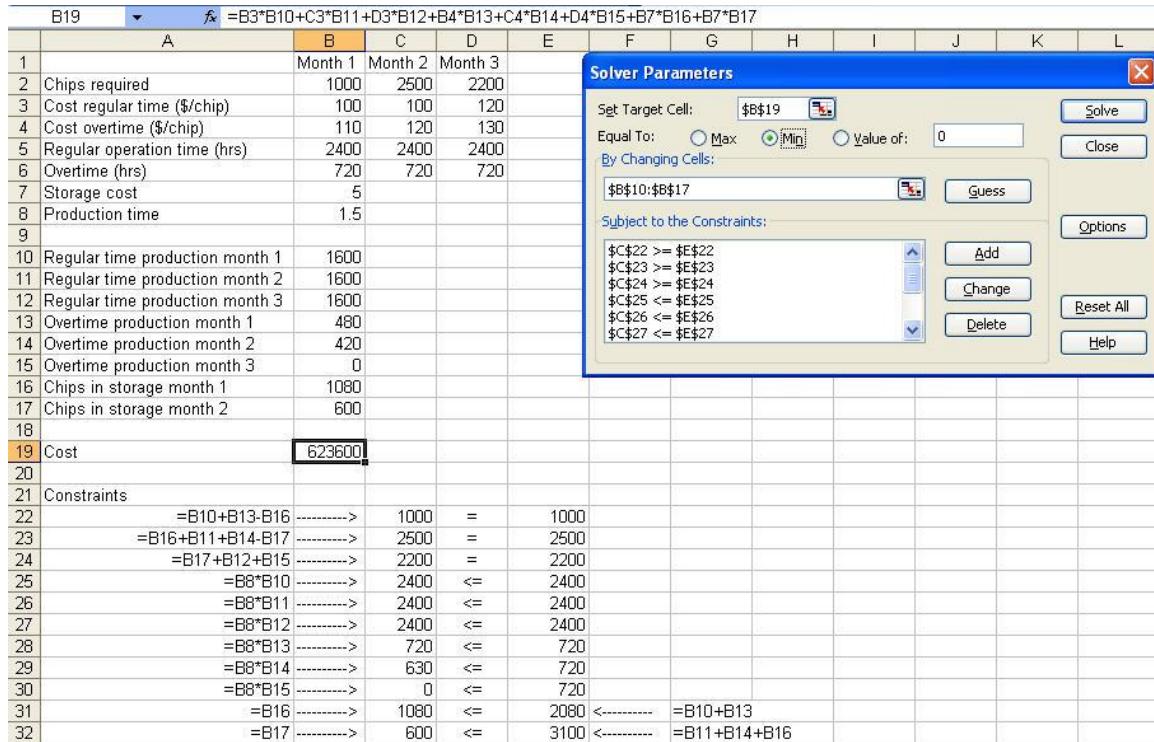
$$1.5x_{o1} \leq 720$$

$$1.5x_{o2} \leq 720$$

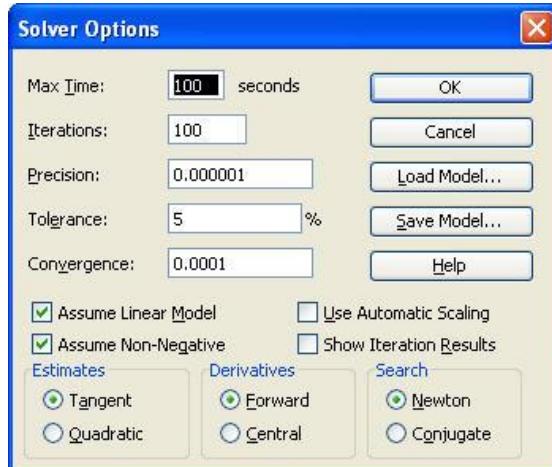
$$1.5x_{o3} \leq 720$$

$$x_{r1}, x_{r2}, x_{r3}, x_{o1}, x_{o2}, x_{o3}, x_{s1}, x_{s2} \geq 0$$

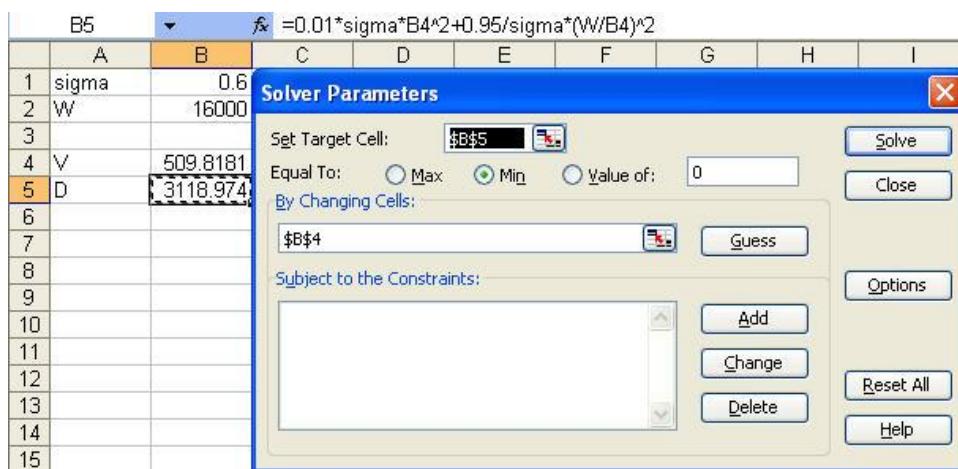
An Excel spreadsheet can be set up to solve the problem as



Note that before depressing the Solve button, the Options button should be depressed and the following boxes should be selected: "Assume Linear Model" and "Assume Non-Negative."



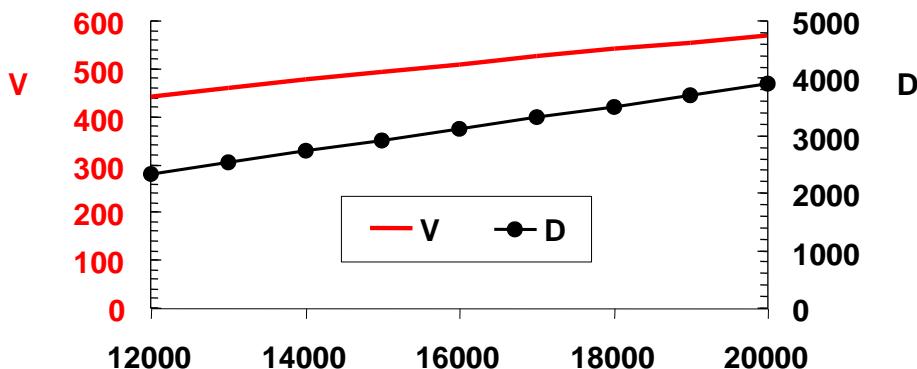
16.27 A tool such as the Excel Solver can be used to determine the solution as



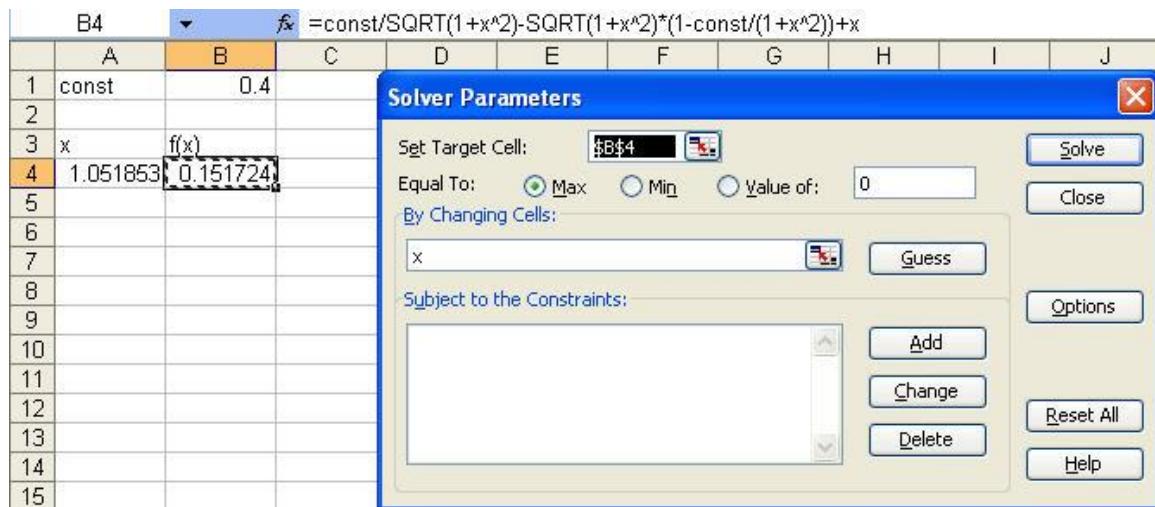
The approach can be implemented to evaluate other values of W with a constant σ to yield the following results:

W	V	D
12000	441.5154	2339.231
13000	459.5438	2534.167
14000	476.8912	2729.102
15000	493.6293	2924.038
16000	509.8181	3118.974
17000	525.5085	3313.910
18000	540.7438	3508.846
19000	555.5614	3703.782
20000	569.9940	3898.718

The optimal velocity along with the minimal drag can be plotted versus weight. As shown below, the relationship is fairly linear for the specified range.



16.28 A tool such as the Excel Solver can be used to determine the solution as



16.29 An LP formulation for this problem can be set up as

$$\text{Minimize } C = 0.05X + 0.025Y + 0.15Z \quad \{\text{Minimize cost}\}$$

subject to

$$\begin{aligned}
 X + Y + Z &\geq 6 && \{\text{Performance constraint}\} \\
 X + Y &< 2.5 && \{\text{Safety constraint}\} \\
 X - Y &\geq 0 && \{X-Y \text{ Relationship constraint}\} \\
 Z - 0.5Y &\geq 0 && \{Y-Z \text{ Relationship constraint}\}
 \end{aligned}$$

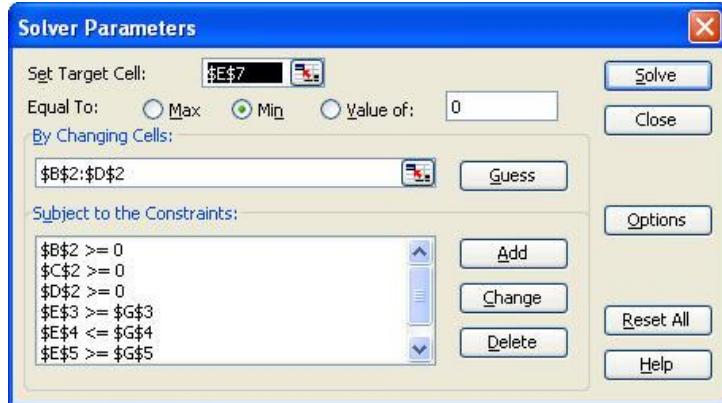
An Excel spreadsheet can be set up to solve the problem as

	A	B	C	D	E	F	G
1		X	Y	Z	Total		Constraint
2	Amount	0	0	0			
3	Performance	1	1	1	0	\geq	6
4	Safety	1	1	0	0	\leq	2.5
5	X-Y	1	-1	0	0	\geq	0
6	Z-0.5Y	0	-0.5	1	0	\geq	0
7	Cost	0.05	0.025	0.15	0		

The formulas are

	A	B	C	D	E	F	G
1		X	Y	Z	Total		Constraint
2	Amount	0	0	0			
3	Performance	1	1	1	$=B3*B\$2+C3*C\$2+D3*D\$2$	\geq	6
4	Safety	1	1	0	$=B4*B\$2+C4*C\$2+D4*D\$2$	\leq	2.5
5	X-Y	1	-1	0	$=B5*B\$2+C5*C\$2+D5*D\$2$	\geq	0
6	Z-0.5Y	0	-0.5	1	$=B6*B\$2+C6*C\$2+D6*D\$2$	\geq	0
7	Cost	0.05	0.025	0.15	$=B7*B\$2+C7*C\$2+D7*D\$2$		

The Solver can be called and set up as



The resulting solution is

	A	B	C	D	E	F	G
1		X	Y	Z	Total		Constraint
2	Amount	1.25	1.25	3.5			
3	Performance	1	1	1	6 >=		6
4	Safety	1	1	0	2.5 <=		2.5
5	X-Y	1	-1	0	0 >=		0
6	Z-0.5*Y	0	-0.5	1	2.875 >=		0
7	Cost	0.05	0.025	0.15	0.61875		

16.30 An LP formulation for this problem can be set up as

Decision variables: x_i = quantity of part i

$$\text{Minimize } P = 375x_A + 275x_B + 475x_C + 325x_D$$

subject to

$$2.5x_A + 1.5x_B + 2.75x_C + 2x_D \leq 640$$

$$3.5x_A + 3x_B + 3x_C + 2x_D \leq 960$$

A tool such as the Excel Solver can be used to determine the solution as

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			Part										
2			A	B	C	D							
3	Fabrication time (hr/100 units)	2.5	1.5	2.75	2								
4	Finishing time (hr/100 units)	3.5	3	3	2								
5	Profit (\$/100 units)	375	275	475	325								
6													
7	Quantity	0	192	128	0								
8	Profit												
9	=B5*B7+C5*C7+D5*D7+E5*E7		----->	113600									
10	Constraints:												
11	Fab Time												
12	=B3*B\$7+C3*C\$7+D3*D\$7+E3*E\$7		----->	640	<=	640							
13	Finish time												
14	=B4*B\$7+C4*C\$7+D4*D\$7+E4*E\$7		----->	960	<=	960							
15													

Solver Parameters

Set Target Cell: \$C\$9

Equal To: Max

By Changing Cells: \$B\$7:\$E\$7

Subject to the Constraints:

\$B\$7 >= 0
\$C\$12 <= \$E\$12
\$C\$14 <= \$E\$14
\$C\$7 >= 0
\$D\$7 >= 0
\$E\$7 >= 0

Thus, the results indicate that if we produce none of parts A and D and 192 and 128 of B and C, respectively, we will generate a maximum profit of \$113,600.

CHAPTER 17

17.1 The data can be tabulated as

<i>i</i>	<i>y</i>	$(y_i - \bar{y})^2$
1	8.8	0.725904
2	9.4	0.063504
3	10	0.121104
4	9.8	0.021904
5	10.1	0.200704
6	9.5	0.023104
7	10.1	0.200704
8	10.4	0.559504
9	9.5	0.023104
10	9.5	0.023104
11	9.8	0.021904
12	9.2	0.204304
13	7.9	3.069504
14	8.9	0.565504
15	9.6	0.002704
16	9.4	0.063504
17	11.3	2.715904
18	10.4	0.559504
19	8.8	0.725904
20	10.2	0.300304
21	10	0.121104
22	9.4	0.063504
23	9.8	0.021904
24	10.6	0.898704
25	<u>8.9</u>	<u>0.565504</u>
Σ	241.3	11.8624

$$(a) \bar{y} = \frac{241.3}{25} = 9.652$$

$$(b) s_y = \sqrt{\frac{11.8624}{25-1}} = 0.703041$$

$$(c) s_y^2 = 0.703041^2 = 0.494267$$

$$(d) c.v. = \frac{0.703041}{9.652} \times 100\% = 7.28\%$$

$$(e) t_{0.05/2,25-1} = 2.063899$$

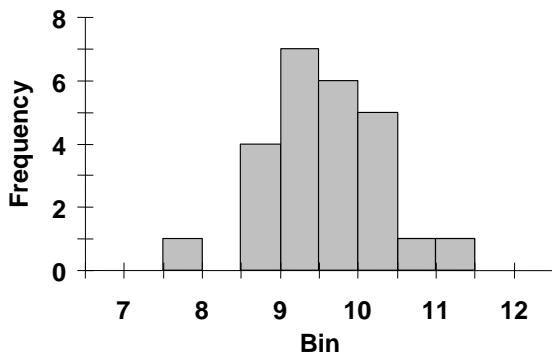
$$L = 9.652 - \frac{0.703041}{\sqrt{25}} 2.063899 = 9.361799$$

$$U = 9.652 + \frac{0.703041}{\sqrt{25}} 2.063899 = 9.942201$$

17.2 The data can be sorted and then grouped. We assume that if a number falls on the border between bins, it is placed in the lower bin.

lower	upper	Frequency
7.5	8	1
8	8.5	0
8.5	9	4
9	9.5	7
9.5	10	6
10	10.5	5
10.5	11	1
11	11.5	1

The histogram can then be constructed as



17.3 The data can be tabulated as

i	y	$(y_i - \bar{y})^2$
1	28.65	0.390625
2	28.65	0.390625
3	27.65	0.140625
4	29.25	1.500625
5	26.55	2.175625
6	29.65	2.640625
7	28.45	0.180625
8	27.65	0.140625
9	26.65	1.890625
10	27.85	0.030625
11	28.65	0.390625
12	28.65	0.390625
13	27.65	0.140625
14	27.05	0.950625
15	28.45	0.180625
16	27.65	0.140625
17	27.35	0.455625
18	28.25	0.050625
19	31.65	13.140625

20	28.55	0.275625
21	28.35	0.105625
22	28.85	0.680625
23	26.35	2.805625
24	27.65	0.140625
25	26.85	1.380625
26	26.75	1.625625
27	27.75	0.075625
28	<u>27.25</u>	<u>0.600625</u>
Σ	784.7	33.0125

(a) $\bar{y} = \frac{784.7}{28} = 28.025$

(b) $s_y = \sqrt{\frac{33.0125}{28-1}} = 1.105751$

(c) $s_y^2 = 1.105751^2 = 1.222685$

(d) c.v. = $\frac{1.105751}{28.025} \times 100\% = 3.95\%$

(e) $t_{0.1/2, 28-1} = 1.703288$

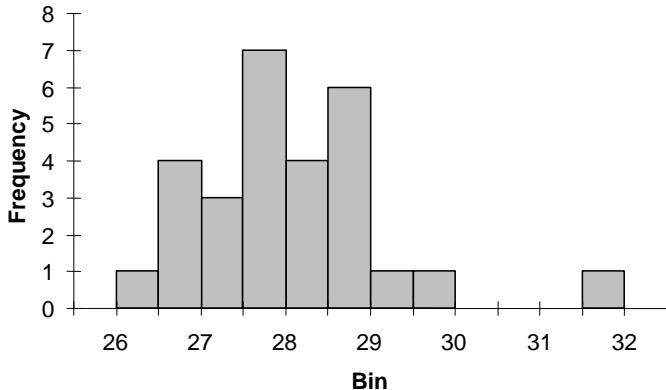
$$L = 28.025 - \frac{1.105751}{\sqrt{28}} 1.703288 = 27.66907$$

$$U = 28.025 + \frac{1.105751}{\sqrt{28}} 1.703288 = 28.38093$$

(f) The data can be sorted and grouped.

Lower	Upper	Frequency
26	26.5	1
26.5	27	4
27	27.5	3
27.5	28	7
28	28.5	4
28.5	29	6
29	29.5	1
29.5	30	1
30	30.5	0
30.5	31	0
31	31.5	0
31.5	32	1

The histogram can then be constructed as

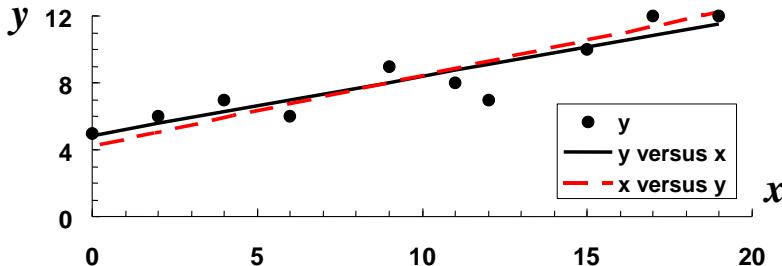


(g) 68% of the readings should fall between $\bar{y} - s_y$ and $\bar{y} + s_y$. That is, between $28.025 - 1.10575096 = 26.919249$ and $28.025 + 1.10575096 = 29.130751$. Twenty values fall between these bounds which is equal to $20/28 = 71.4\%$ of the values which is not that far from 68%.

17.4 The results can be summarized as

	y versus x	x versus y
Best fit equation	$y = 4.851535 + 0.35247x$	$x = -9.96763 + 2.374101y$
Standard error	1.06501	2.764026
Correlation coefficient	0.914767	0.914767

We can also plot both lines on the same graph

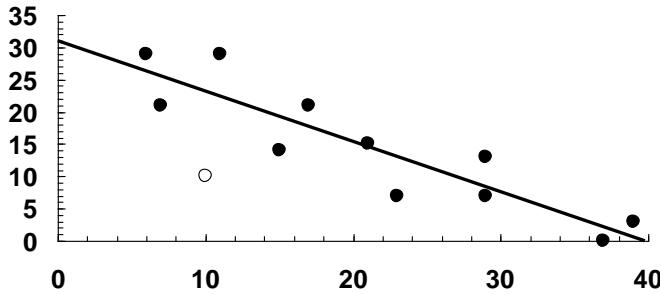


Thus, the “best” fit lines and the standard errors differ. This makes sense because different errors are being minimized depending on our choice of the dependent (ordinate) and independent (abscissa) variables. In contrast, the correlation coefficients are identical since the same amount of uncertainty is explained regardless of how the points are plotted.

17.5 The results can be summarized as

$$y = 31.0589 - 0.78055x \quad (s_{y/x} = 4.476306; r = 0.901489)$$

At $x = 10$, the best fit equation gives 23.2543. The line and data can be plotted along with the point (10, 10).



The value of 10 is nearly 3 times the standard error away from the line,

$$23.2543 - 3(4.476306) = 9.824516$$

Thus, we can tentatively conclude that the value is probably erroneous. It should be noted that the field of statistics provides related but more rigorous methods to assess whether such points are “outliers.”

17.6 The sum of the squares of the residuals for this case can be written as

$$S_r = \sum_{i=1}^n (y_i - a_1 x_i)^2$$

The partial derivative of this function with respect to the single parameter a_1 can be determined as

$$\frac{\partial S_r}{\partial a_1} = -2 \sum [(y_i - a_1 x_i) x_i]$$

Setting the derivative equal to zero and evaluating the summations gives

$$0 = \sum y_i x_i - a_1 \sum x_i^2$$

which can be solved for

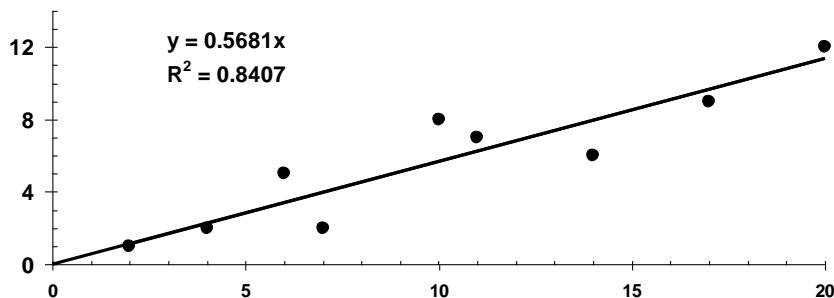
$$a_1 = \frac{\sum y_i x_i}{\sum x_i^2}$$

So the slope that minimizes the sum of the squares of the residuals for a straight line with a zero intercept is merely the ratio of the sum of the dependent variables (y) times the sum of the independent variables (x) over the sum of the independent variables squared (x^2). Application to the data gives

x	y	xy	x^2
2	1	2	4
4	2	8	16

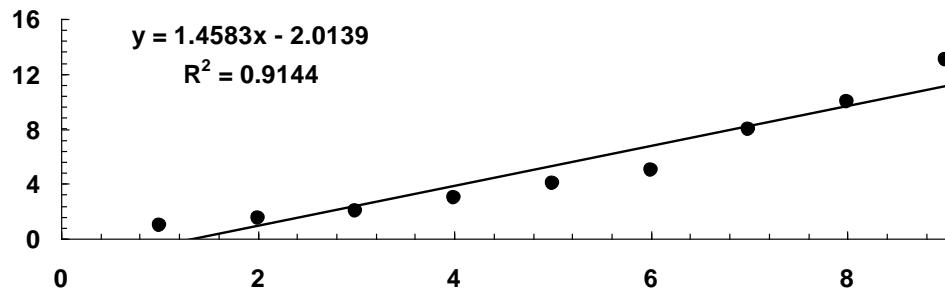
6	5	30	36
7	2	14	49
10	8	80	100
11	7	77	121
14	6	84	196
17	9	153	289
20	12	<u>240</u>	<u>400</u>
		688	1211

Therefore, the slope can be computed as $688/1211 = 0.5681$. The fit along with the data can be displayed as



17.7 (a) The results can be summarized as

$$y = -2.01389 + 1.458333x \quad (s_{y/x} = 1.306653; r = 0.956222)$$

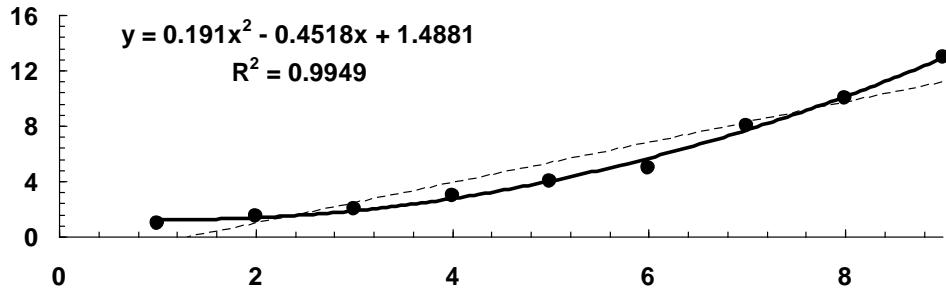


As can be seen, although the correlation coefficient appears to be close to 1, the straight line does not describe the data trend very well.

(b) The results can be summarized as

$$y = 1.488095 - 0.45184x + 0.191017x^2 \quad (s_{y/x} = 0.344771; r = 0.997441)$$

A plot indicates that the quadratic fit does a much better job of fitting the data.



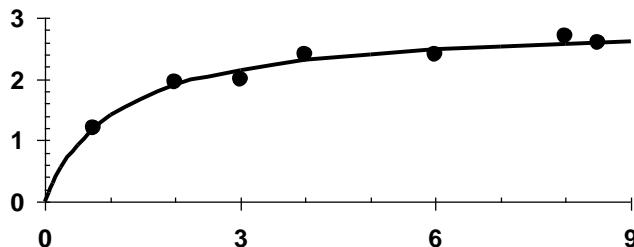
17.8 (a) We regress $1/y$ versus $1/x$ to give

$$\frac{1}{y} = 0.34154 + 0.36932 \frac{1}{x}$$

Therefore, $\alpha_3 = 1/0.34154 = 2.927913$ and $\beta_3 = 0.36932(2.927913) = 1.081337$, and the saturation-growth-rate model is

$$y = 2.927913 \frac{x}{1.081337 + x}$$

The model and the data can be plotted as



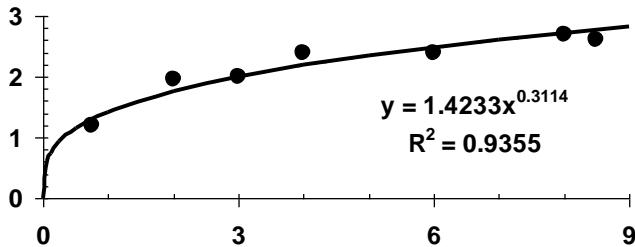
(b) We regress $\log_{10}(y)$ versus $\log_{10}(x)$ to give

$$\log_{10} y = 0.153296 + 0.311422 \log_{10} x$$

Therefore, $\alpha_2 = 10^{0.153296} = 1.423297$ and $\beta_2 = 0.311422$, and the power model is

$$y = 1.423297 x^{0.311422}$$

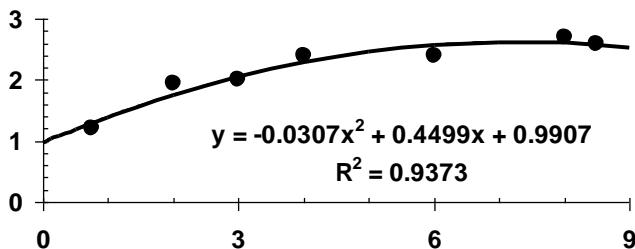
The model and the data can be plotted as



(c) Polynomial regression can be applied to develop a best-fit parabola

$$y = -0.03069 x^2 + 0.449901 x + 0.990728$$

The model and the data can be plotted as



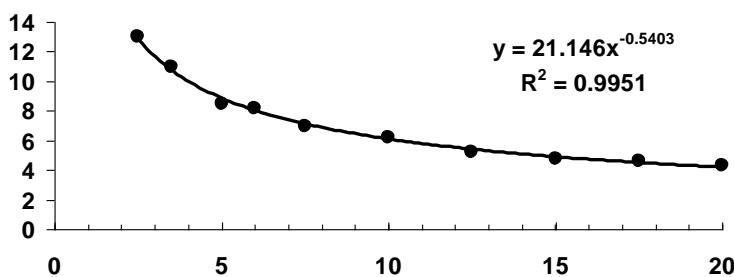
17.9 We regress $\log_{10}(y)$ versus $\log_{10}(x)$ to give

$$\log_{10} y = 1.325225 - 0.54029 \log_{10} x$$

Therefore, $\alpha_2 = 10^{1.325225} = 21.14583$ and $\beta_2 = -0.54029$, and the power model is

$$y = 21.14583 x^{-0.54029}$$

The model and the data can be plotted as



The model can be used to predict a value of $21.14583(9)^{-0.54029} = 6.451453$.

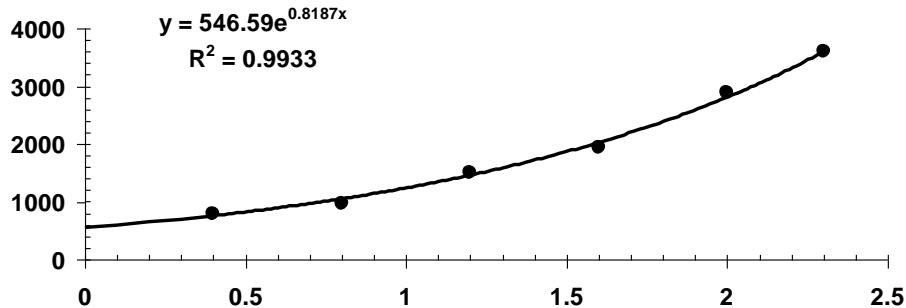
17.10 We regress $\ln(y)$ versus x to give

$$\ln y = 6.303701 + 0.818651x$$

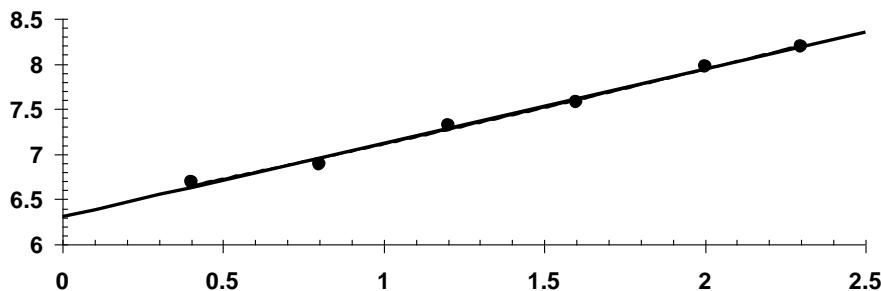
Therefore, $\alpha_1 = e^{6.303701} = 546.5909$ and $\beta_1 = 0.818651$, and the exponential model is

$$y = 546.5909 e^{0.818651x}$$

The model and the data can be plotted as



A semi-log plot can be developed by plotting the natural log versus x . As expected, both the data and the best-fit line are linear when plotted in this way.



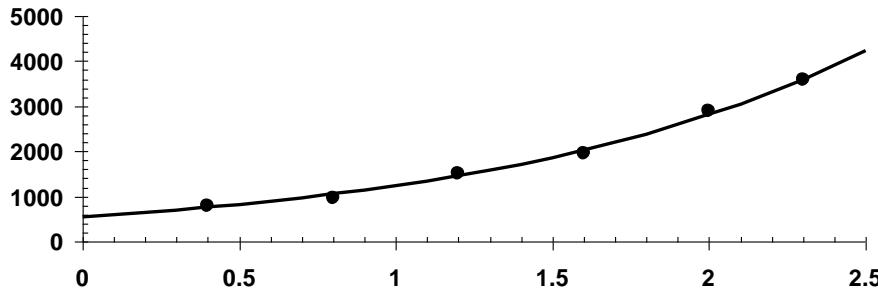
17.11 For the data from Prob. 17.10, we regress $\log_{10}(y)$ versus x to give

$$\log_{10} y = 2.737662 + 0.355536x$$

Therefore, $\alpha_5 = 10^{2.737662} = 546.5909$ and $\beta_5 = 0.355536$, and the base-10 exponential model is

$$y = 546.5909 \times 10^{0.355536x}$$

The model and the data can be plotted as



This plot is identical to the graph that was generated with the base- e model derived in Prob. 17.10. Thus, although the models have a different base, they yield identical results.

The relationship between β_1 and β_5 can be developed as in

$$e^{-\beta_1 t} = 10^{-\beta_5 t}$$

Take the natural log of this equation to yield

$$-\beta_1 t = -\beta_5 t \ln 10$$

or

$$\beta_1 = 2.302585 \beta_5$$

This result can be verified by substituting the value of β_5 into this equation to give

$$\beta_1 = 2.302585(0.355536) = 0.818651$$

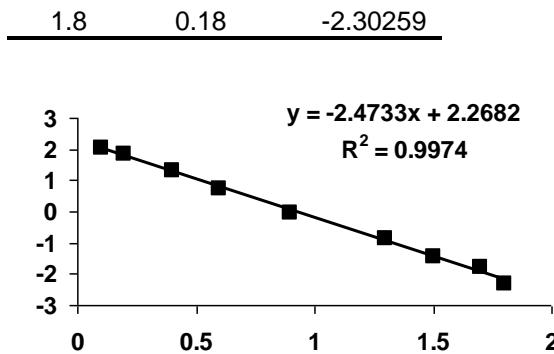
This is identical to the result derived in Prob. 17.10.

17.12 The function can be linearized by dividing it by x and taking the natural logarithm to yield

$$\ln(y/x) = \ln \alpha_4 + \beta_4 x$$

Therefore, if the model holds, a plot of $\ln(y/x)$ versus x should yield a straight line with an intercept of $\ln \alpha_4$ and a slope of β_4 .

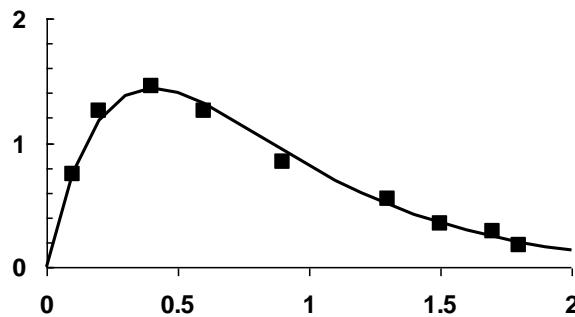
x	y	$\ln(y/x)$
0.1	0.75	2.014903
0.2	1.25	1.832581
0.4	1.45	1.287854
0.6	1.25	0.733969
0.9	0.85	-0.057116
1.3	0.55	-0.8602
1.5	0.35	-1.45529
1.7	0.28	-1.80359



Therefore, $\beta_4 = -2.4733$ and $\alpha_4 = e^{2.2682} = 9.661786$, and the fit is

$$y = 9.661786 xe^{-2.4733x}$$

This equation can be plotted together with the data:



17.13 The equation can be linearized by inverting it to yield

$$\frac{1}{k} = \frac{c_s}{k_{\max}} \frac{1}{c^2} + \frac{1}{k_{\max}}$$

Consequently, a plot of $1/k$ versus $1/c$ should yield a straight line with an intercept of $1/k_{\max}$ and a slope of c_s/k_{\max}

c , mg/L	k , /d	$1/c^2$	$1/k$	$1/c^2 \times 1/k$	$(1/c^2)^2$
0.5	1.1	4.000000	0.909091	3.636364	16.000000
0.8	2.4	1.562500	0.416667	0.651042	2.441406
1.5	5.3	0.444444	0.188679	0.083857	0.197531
2.5	7.6	0.160000	0.131579	0.021053	0.025600
4	8.9	0.062500	0.112360	0.007022	0.003906
Sum →		6.229444	1.758375	4.399338	18.66844

The slope and the intercept can be computed as

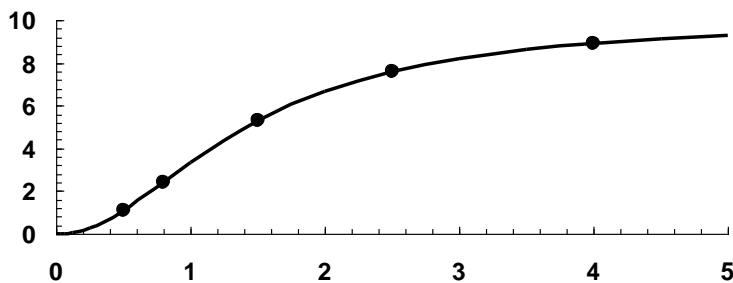
$$a_1 = \frac{5(4.399338) - 6.229444(1.758375)}{5(18.66844) - (6.229444)^2} = 0.202489$$

$$a_0 = \frac{1.758375}{5} - 0.202489 \frac{6.229444}{5} = 0.099396$$

Therefore, $k_{\max} = 1/0.099396 = 10.06074$ and $c_s = 10.06074(0.202489) = 2.037189$, and the fit is

$$k = \frac{10.06074 c^2}{2.037189 + c^2}$$

This equation can be plotted together with the data:



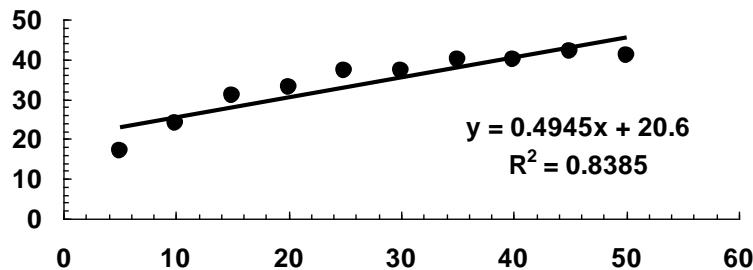
The equation can be used to compute

$$k = \frac{10.06074(2)^2}{2.037189 + (2)^2} = 6.666$$

17.14 (a) We regress y versus x to give

$$y = 20.6 + 0.494545x$$

The model and the data can be plotted as



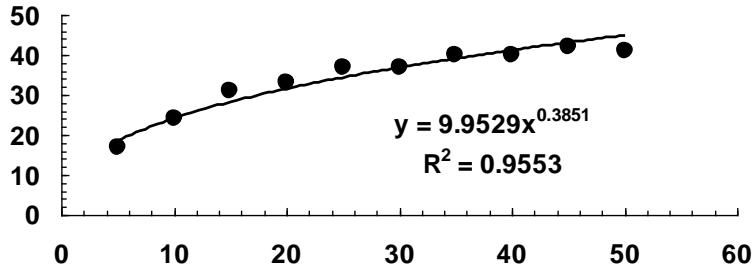
(b) We regress $\log_{10}y$ versus $\log_{10}x$ to give

$$\log_{10} y = 0.99795 + 0.385077 \log_{10} x$$

Therefore, $\alpha_2 = 10^{0.99795} = 9.952915$ and $\beta_2 = 0.385077$, and the power model is

$$y = 9.952915 x^{0.385077}$$

The model and the data can be plotted as



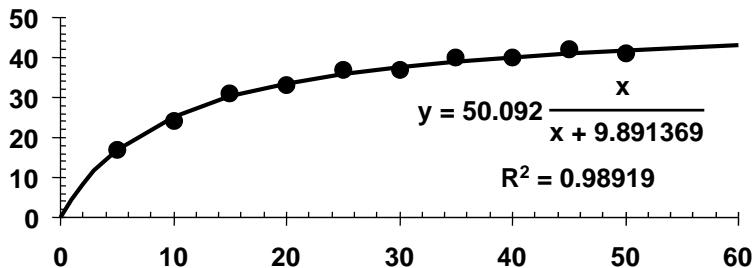
(c) We regress $1/y$ versus $1/x$ to give

$$\frac{1}{y} = 0.019963 + 0.197464 \frac{1}{x}$$

Therefore, $\alpha_3 = 1/0.01996322 = 50.09212$ and $\beta_3 = 0.19746357(50.09212) = 9.89137$, and the saturation-growth-rate model is

$$y = 50.09212 \frac{x}{9.89137 + x}$$

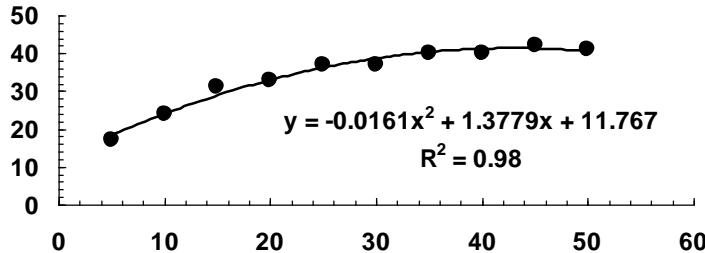
The model and the data can be plotted as



(d) We employ polynomial regression to fit a parabola

$$y = -0.01606 x^2 + 1.377879 x + 11.76667$$

The model and the data can be plotted as



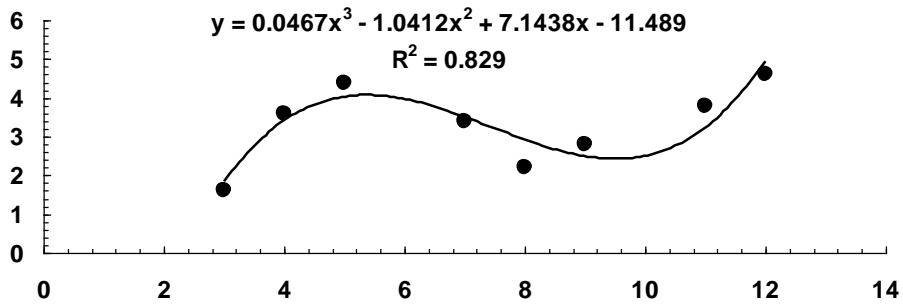
Comparison of fits: The linear fit is obviously inadequate. Although the power fit follows the general trend of the data, it is also inadequate because (1) the residuals do not appear to be randomly distributed around the best fit line and (2) it has a lower r^2 than the saturation and parabolic models.

The best fits are for the saturation-growth-rate and the parabolic models. They both have randomly distributed residuals and they have similar high coefficients of determination. The saturation model has a slightly higher r^2 . Although the difference is probably not statistically significant, in the absence of additional information, we can conclude that the saturation model represents the best fit.

17.15 We employ polynomial regression to fit a cubic equation to the data

$$y = 0.046676x^3 - 1.04121x^2 + 7.143817x - 11.4887 \quad (r^2 = 0.828981; s_{y/x} = 0.570031)$$

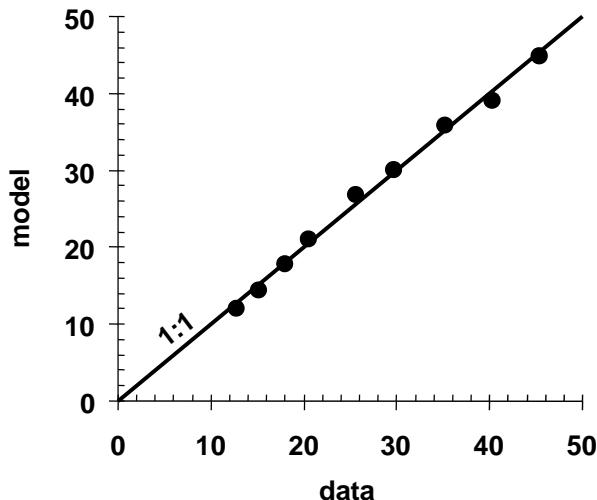
The model and the data can be plotted as



17.16 We employ multiple linear regression to fit the following equation to the data

$$y = 14.46087 + 9.025217x_1 - 5.70435x_2 \quad (r^2 = 0.995523; s_{y/x} = 0.888787)$$

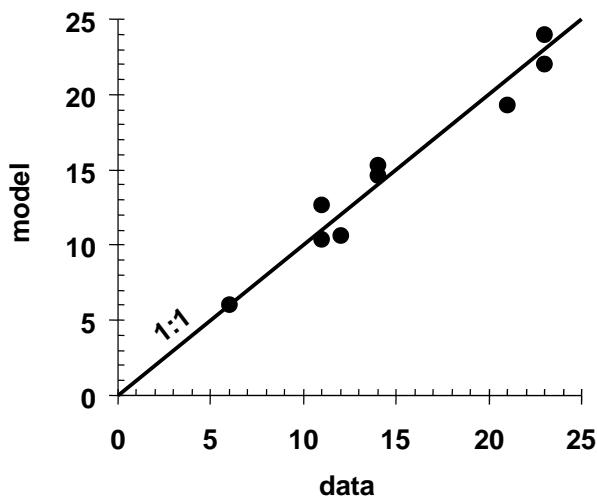
The model and the data can be compared graphically by plotting the model predictions versus the data. A 1:1 line is included to indicate a perfect fit.



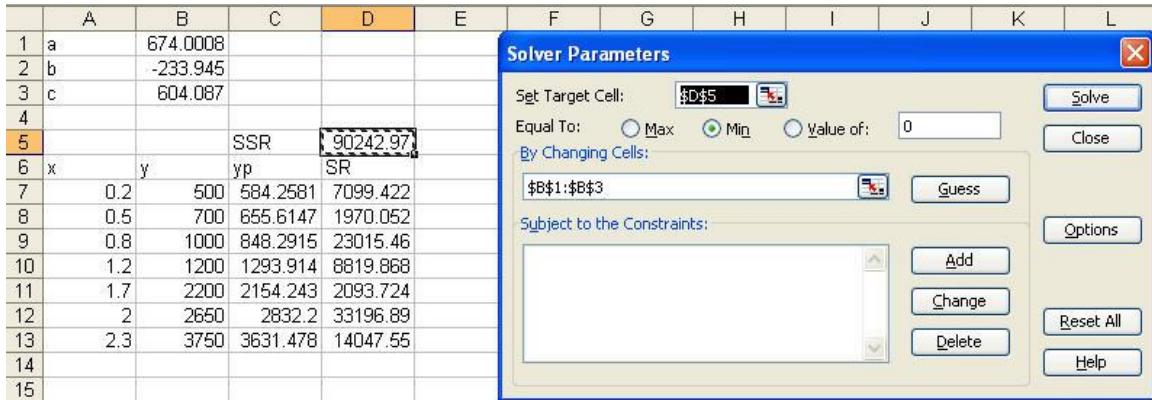
17.17 We employ multiple linear regression to fit the following equation to the data

$$y = 14.66667 - 6.66667 x_1 + 2.33333 x_2 \quad (r^2 = 0.958333; s_{y/x} = 1.414214)$$

The model and the data can be compared graphically by plotting the model predictions versus the data. A 1:1 line is included to indicate a perfect fit.



17.18 We can employ nonlinear regression to fit a parabola to the data. A simple way to do this is to use the Excel Solver to minimize the sum of the squares of the residuals as in the following worksheet,



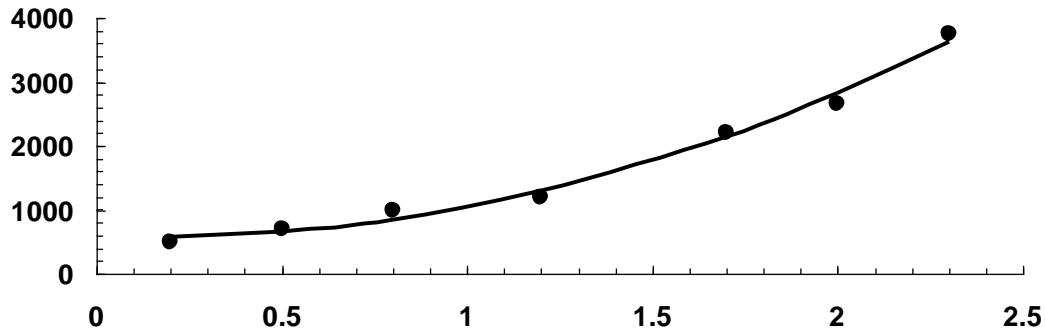
The formulas are

	A	B	C	D
1	a	674.000780425739		
2	b	-233.945011343731		
3	c	604.087039907068		
4				
5			SSR	=SUM(D7:D13)
6	x	y	yp	SR
7	0.2	500	584.2581	7099.422
8	0.5	700	655.6147	1970.052
9	0.8	1000	848.2915	23015.46
10	1.2	1200	1293.914	8819.868
11	1.7	2200	2154.243	2093.724
12	2	2650	2832.2	33196.89
13	2.3	3750	3631.478	14047.55
14				
15				

Thus, the best-fit equation is

$$y = 674.001x^2 - 233.945x + 604.087$$

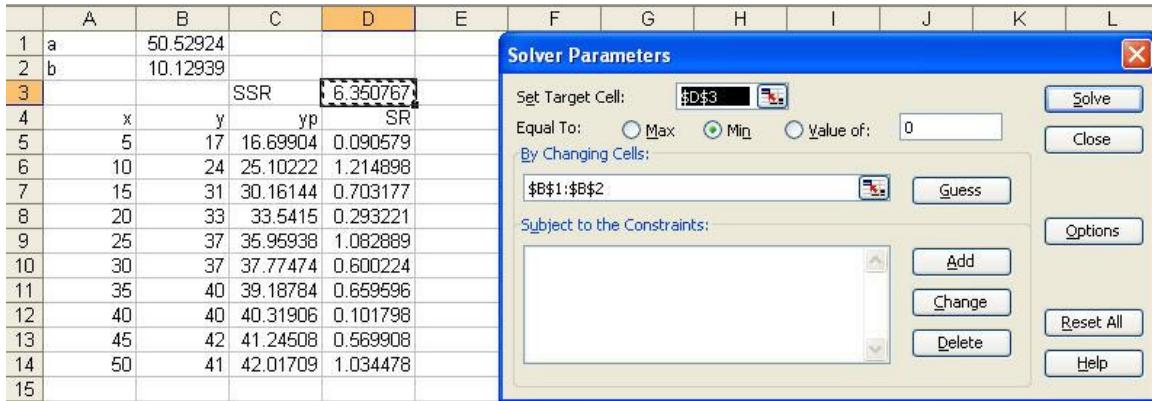
The model and the data can be displayed graphically as



Note that if polynomial regression were used, a slightly different fit would result,

$$y = 674.007x^2 - 233.961x + 604.094$$

17.19 We can employ nonlinear regression to fit the saturation-growth-rate equation to the data from Prob. 17.14. A simple way to do this is to use the Excel Solver to minimize the sum of the squares of the residuals as in the following worksheet,



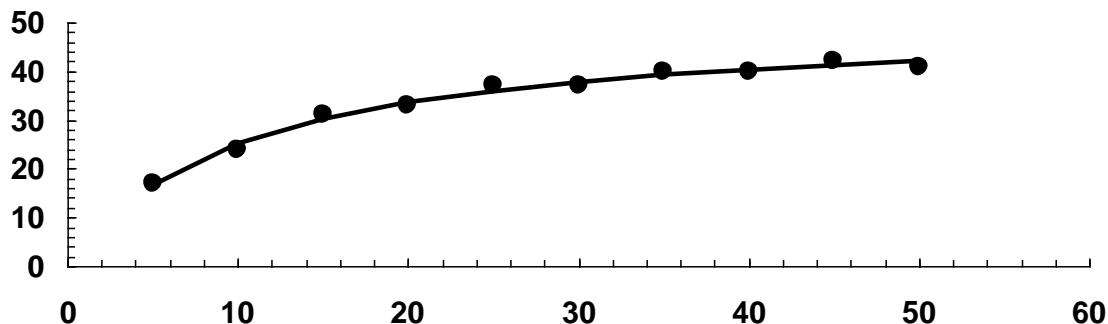
The formulas are

A	B	C	D
1 a	50.5292424213902		
2 b	10.1293886897916		
3		SSR	=SUM(D5:D14)
4 x	y	yp	SR
5 5	17	=\$B\$1*A5/(\$B\$2+A5)	=(B5-C5)^2
6 10	24	=\$B\$1*A6/(\$B\$2+A6)	=(B6-C6)^2
7 15	31	=\$B\$1*A7/(\$B\$2+A7)	=(B7-C7)^2
8 20	33	=\$B\$1*A8/(\$B\$2+A8)	=(B8-C8)^2
9 25	37	=\$B\$1*A9/(\$B\$2+A9)	=(B9-C9)^2
10 30	37	=\$B\$1*A10/(\$B\$2+A10)	=(B10-C10)^2
11 35	40	=\$B\$1*A11/(\$B\$2+A11)	=(B11-C11)^2
12 40	40	=\$B\$1*A12/(\$B\$2+A12)	=(B12-C12)^2
13 45	42	=\$B\$1*A13/(\$B\$2+A13)	=(B13-C13)^2
14 50	41	=\$B\$1*A14/(\$B\$2+A14)	=(B14-C14)^2

Thus, the best-fit equation is

$$y = 50.529 \frac{x}{10.129 + x}$$

The model and the data can be displayed graphically as



Recall that for Prob. 17.14c, a slightly different fit resulted,

$$y = 50.09212 \frac{x}{9.89137 + x}$$

17.20 MATLAB provides a very nice environment for solving this problem:

(a) Prob. 17.4:

First, we can enter the data

```
>> X=[0 2 4 6 9 11 12 15 17 19]';
>> Y=[5 6 7 6 9 8 7 10 12 12]';
```

Then, we can create the Z matrix which consists of a column of ones and a second column of the x 's.

```
>> Z=[ones(size(X)) X]
```

```
Z =
1 0
1 2
1 4
1 6
1 9
1 11
1 12
1 15
1 17
1 19
```

Next we can develop the coefficients of the normal equations as

```
>> ZTZ=Z' * Z
ZTZ =
10 95
95 1277
```

We can compute the right-hand side of the normal equations with

```
>> ZTY=Z' * Y
ZTY =
82
911
```

We can then determine the coefficients for the linear regression as

```
>> A=inv(ZTZ) * ZTY
```

```
A =
4.8515
0.3525
```

This result is identical to that obtained in Prob. 17.4. Next, we can determine the r^2 and s_{yx} ,

```
>> Sr=sum((Y-Z*A).^2)

Sr =
9.0740

>> r2=1-Sr/sum((Y-mean(Y)).^2)

r2 =
0.8368

>> syx=sqrt(Sr/(length(X)-length(A)))

syx =
1.0650
```

In order to determine the confidence intervals we can first calculate the inverse of $[Z]^T[Z]$ as

```
>> ZTZI=inv(ZTZ)

ZTZI =
0.3410 -0.0254
-0.0254 0.0027
```

The standard errors of the coefficients can be computed as

```
>> sa0=sqrt(ZTZI(1,1)*syx^2)

sa0 =
0.6219

>> sa1=sqrt(ZTZI(2,2)*syx^2)

sa1 =
0.0550
```

The t statistic can be determined as $TINV(0.1, 10 - 2) = 1.8595$. We can then compute the confidence intervals as

```
>> a0min=A(1)-1.8595*sa0;
>> a0max=A(1)+1.8595*sa0;
>> a1min=A(2)-1.8595*sa1;
>> a1max=A(2)+1.8595*sa1;
```

which yields the confidence intervals for a_0 and a_1 as [3.6951, 6.0080] and [0.2501, 0.4548], respectively.

(b) Prob. 17.15:

First, we can determine the coefficients

```
>> X=[3 4 5 7 8 9 11 12]';
>> Y=[1.6 3.6 4.4 3.4 2.2 2.8 3.8 4.6]';
>> Z=[ones(size(X)) X X.^2 X.^3];
>> ZTZ=Z'*Z;
>> ZTY=Z'*Y;
>> A=inv(ZTZ)*ZTY

A =
-11.4887
 7.1438
-1.0412
 0.0467
```

The standard error can be computed as

```
>> Sr=sum((Y-Z*A).^2);
>> syx=sqrt(Sr/(length(Y)-length(A)))

syx =
 0.5700
```

The standard errors of the coefficients can be computed as

```
>> ZTZI=inv(ZTZ)

ZTZI =
 49.3468 -23.4771   3.2960  -0.1412
 -23.4771  11.4162  -1.6270   0.0705
   3.2960  -1.6270   0.2349  -0.0103
  -0.1412   0.0705  -0.0103   0.0005

>> sa0=sqrt(ZTZI(1,1)*syx^2)

sa0 =
 4.0043

>> sa1=sqrt(ZTZI(2,2)*syx^2)

sa1 =
 1.9260

>> sa2=sqrt(ZTZI(3,3)*syx^2)

sa2 =
 0.2763

>> sa3=sqrt(ZTZI(4,4)*syx^2)

sa3 =
 0.0121
```

The t statistic can be determined as $\text{TINV}(0.1, 8 - 4) = 2.13185$. We can then compute the confidence intervals for a_0, a_1, a_2 , and a_3 as $[-20.0253, -2.9521]$, $[3.0379, 11.2498]$, $[-1.6302, -0.45219]$, and $[0.02078, 0.072569]$, respectively.

17.21 Here's VBA code to implement linear regression:

```

Option Explicit

Sub Regres()
    Dim n As Integer
    Dim x(20) As Double, y(20) As Double, a1 As Double, a0 As Double
    Dim syx As Double, r2 As Double
    n = 7
    x(1) = 1: x(2) = 2: x(3) = 3: x(4) = 4: x(5) = 5
    x(6) = 6: x(7) = 7
    y(1) = 0.5: y(2) = 2.5: y(3) = 2: y(4) = 4: y(5) = 3.5
    y(6) = 6: y(7) = 5.5
    Call Linreg(x, y, n, a1, a0, syx, r2)
    MsgBox "slope= " & a1
    MsgBox "intercept= " & a0
    MsgBox "standard error= " & syx
    MsgBox "coefficient of determination= " & r2
    MsgBox "correlation coefficient= " & Sqr(r2)
End Sub

Sub Linreg(x, y, n, a1, a0, syx, r2)
    Dim i As Integer
    Dim sumx As Double, sumy As Double, sumxy As Double
    Dim sumx2 As Double, st As Double, sr As Double
    Dim xm As Double, ym As Double
    sumx = 0
    sumy = 0
    sumxy = 0
    sumx2 = 0
    st = 0
    sr = 0
    'determine summations for regression
    For i = 1 To n
        sumx = sumx + x(i)
        sumy = sumy + y(i)
        sumxy = sumxy + x(i) * y(i)
        sumx2 = sumx2 + x(i) ^ 2
    Next i
    'determine means
    xm = sumx / n
    ym = sumy / n
    'determine coefficients
    a1 = (n * sumxy - sumx * sumy) / (n * sumx2 - sumx * sumx)
    a0 = ym - a1 * xm
    'determine standard error and coefficient of determination
    For i = 1 To n
        st = st + (y(i) - ym) ^ 2
        sr = sr + (y(i) - a1 * x(i) - a0) ^ 2
    Next i
    syx = Sqr(st / (n - 2))
    r2 = st / sumy
End Sub

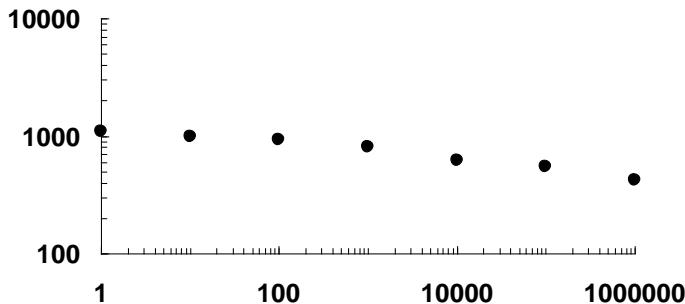
```

```

Next i
syx = (sr / (n - 2)) ^ 0.5
r2 = (st - sr) / st
End Sub

```

17.22 A log-log plot of stress versus N suggests a linear relationship.



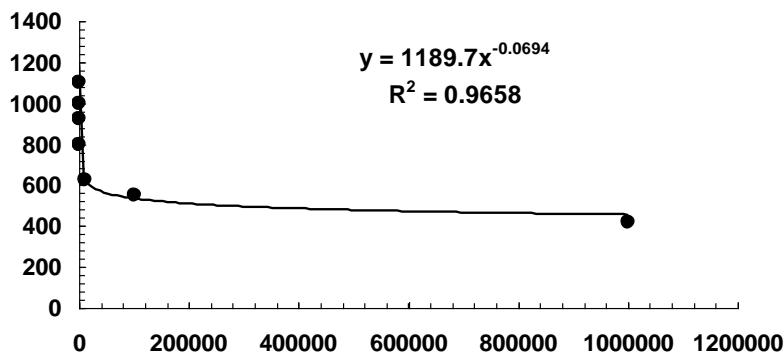
We regress $\log_{10}(\text{stress})$ versus $\log_{10}(N)$ to give

$$\log_{10}(\text{stress}) = 3.075442 - 0.06943 \log_{10} N$$

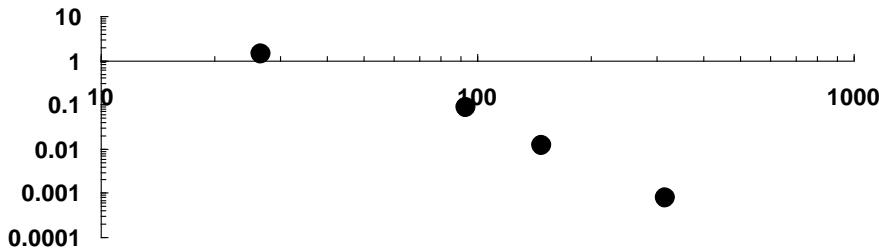
Therefore, $\alpha_2 = 10^{3.075442} = 1189.711$ and $\beta_2 = -0.06943$, and the power model is

$$\text{stress} = 1189.711 N^{-0.06943}$$

The model and the data can be plotted on untransformed scales as



17.23 A log-log plot of μ versus T suggests a linear relationship.



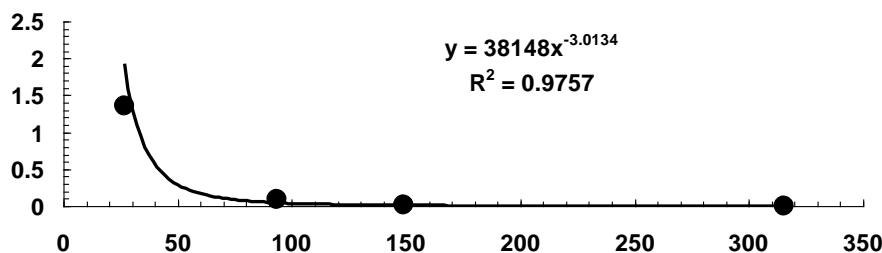
We regress $\log_{10}\mu$ versus $\log_{10}T$ to give

$$\log_{10} \mu = 4.581471 - 3.01338 \log_{10} T \quad (r^2 = 0.975703)$$

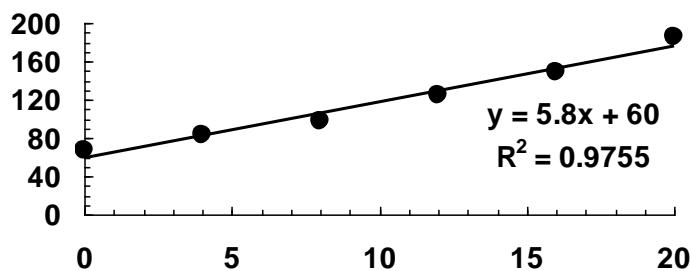
Therefore, $\alpha_2 = 10^{4.581471} = 38,147.94$ and $\beta_2 = -3.01338$, and the power model is

$$\mu = 38,147.94 T^{-3.01338}$$

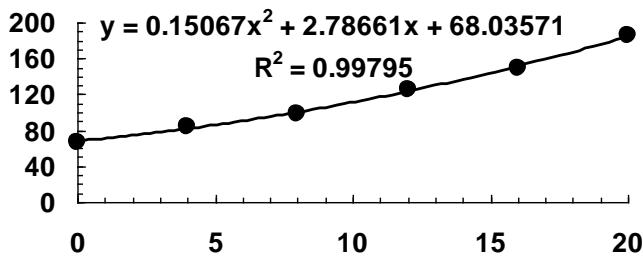
The model and the data can be plotted on untransformed scales as



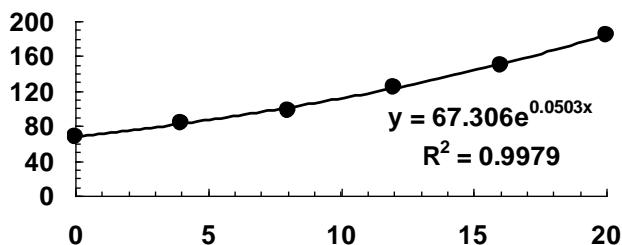
17.24 This problem was solved using an Excel spreadsheet and TrendLine. Linear regression gives



Polynomial regression yields a best-fit parabola



Exponential model:



The linear model is inadequate since it does not capture the curving trend of the data. At face value, the parabolic and exponential models appear to be equally good. However, knowledge of bacterial growth might lead you to choose the exponential model as it is commonly used to simulate the growth of microorganism populations. Interestingly, the choice matters when the models are used for prediction. If the exponential model is used, the result is

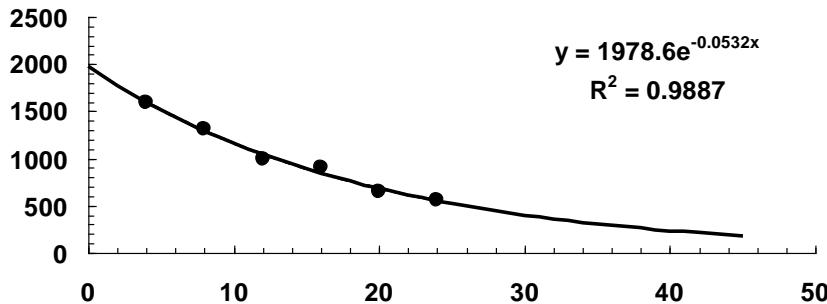
$$B = 67.306e^{0.0503(40)} = 503.3317$$

For the parabolic model, the prediction is

$$B = 0.15067t^2 + 2.78661t + 68.03571 = 420.5721$$

Thus, even though the models would yield very similar results within the data range, they yield dramatically different results for extrapolation outside the range.

17.25 The exponential model is ideal for this problem since (1) it does not yield negative results (as could be the case with a polynomial), and (2) it always decreases with time. Further, it is known that bacterial death is well approximated by the exponential model.



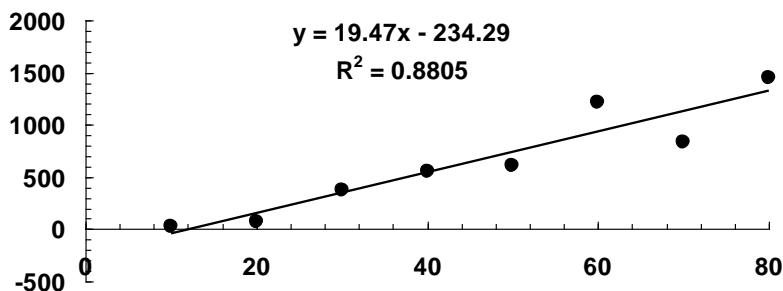
- (a) The model says that the concentration at $t = 0$ was 1978.6.
(b) The time at which the concentration reaches 200 can be computed as

$$200 = 1978.6e^{-0.0532t}$$

$$\ln\left(\frac{200}{1978.6}\right) = -0.0532t$$

$$t = \frac{\ln\left(\frac{200}{1978.6}\right)}{-0.0532} = 43.1 \text{ hr}$$

17.26 (a) Linear model



Although this model does a good job of capturing the trend of the data, it has the disadvantage that it yields a negative intercept. Since this is clearly a physically unrealistic result, another model would be preferable.

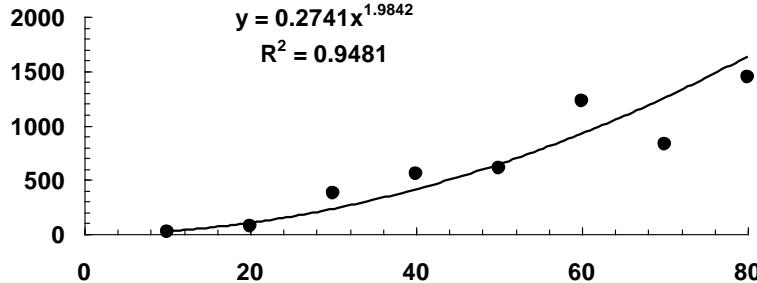
- (b) Power model based on log transformations. We regress $\log_{10}(F)$ versus $\log_{10}(v)$ to give

$$\log_{10} F = -0.56203 + 1.984176 \log_{10} v$$

Therefore, $\alpha_2 = 10^{-0.56203} = 0.274137$ and $\beta_2 = 1.984176$, and the power model is

$$F = 0.274137 v^{1.984176}$$

The model and the data can be plotted as



This model represents a superior fit of the data as it fits the data nicely (the r^2 is superior to that obtained with the linear model in (a)) while maintaining a physically realistic zero intercept.

(c) Power model based on nonlinear regression. We can use the Excel Solver to determine the fit.



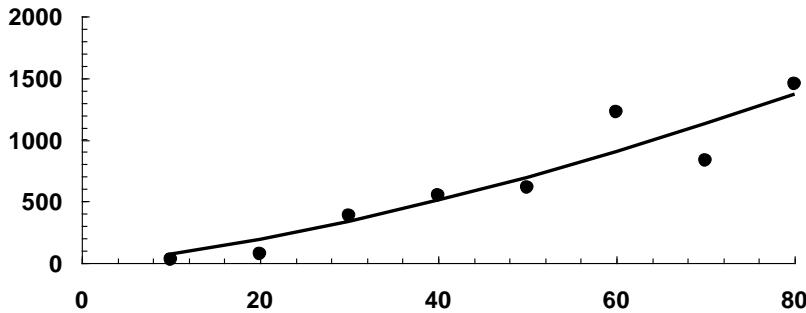
The cell formulas are

	A	B	C	D
1	a2	2.53812		
2	b2	1.43586		
3		SSR	222604.9	
4	v, m/s	F, N	F-pred	SR
5	10	25	69.24288	1957.433
6	20	70	187.3325	13766.92
7	30	380	335.3175	1996.529
8	40	550	506.817	1864.774
9	50	610	698.2335	7785.157
10	60	1220	907.1815	97855.43
11	70	830	1131.933	91163.39
12	80	1450	1371.163	6215.241
13				
14				
15				

Therefore, the best-fit model is

$$F = 2.53842 v^{1.43585}$$

The model and the data can be plotted as



This model also represents a superior fit of the data as it fits the data nicely while maintaining a physically realistic zero intercept. However, it is very interesting to note that the fit is quite different than that obtained with log transforms in (b).

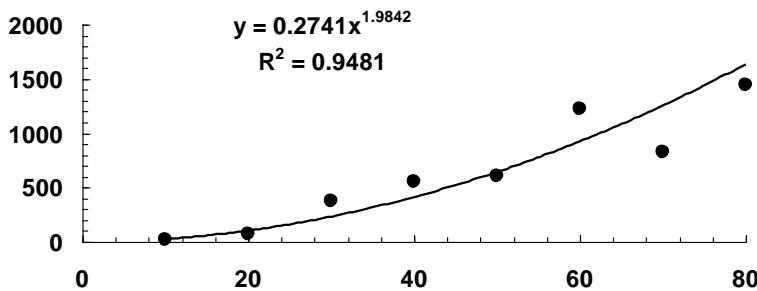
17.27 We can develop a power equation based on natural logarithms. To do this, we regress $\ln(F)$ versus $\ln(v)$ to give

$$\ln F = -1.29413 + 1.984176 \ln v$$

Therefore, $\alpha_2 = e^{-1.29413} = 0.274137$ and $\beta_2 = 1.984176$, and the power model is

$$F = 0.274137 v^{1.984176}$$

The model and the data can be plotted as



Note that this result is identical to that obtained with common logarithms in Prob. 17.26(b). Thus, we can conclude that any base logarithm would yield the same power model.

17.28 The sum of the squares of the residuals for this case can be written as

$$S_r = \sum_{i=1}^n (y_i - a_1 x_i - a_2 x_i^2)^2$$

The partial derivatives of this function with respect to the unknown parameters can be determined as

$$\frac{\partial S_r}{\partial a_1} = -2 \sum [(y_i - a_1 x_i - a_2 x_i^2) x_i]$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum [(y_i - a_1 x_i - a_2 x_i^2) x_i^2]$$

Setting the derivative equal to zero and evaluating the summations gives

$$(\sum x_i^2) a_1 + (\sum x_i^3) a_2 = \sum x_i y_i$$

$$(\sum x_i^3) a_1 + (\sum x_i^4) a_2 = \sum x_i^2 y_i$$

which can be solved for

$$a_1 = \frac{\sum x_i y_i \sum x_i^4 - \sum x_i^2 y_i \sum x_i^3}{\sum x_i^2 \sum x_i^4 - (\sum x_i^3)^2}$$

$$a_2 = \frac{\sum x_i^2 \sum x_i^2 y_i - \sum x_i y_i \sum x_i^3}{\sum x_i^2 \sum x_i^4 - (\sum x_i^3)^2}$$

The model can be tested for the data from Table 12.1.

x	y	x^2	x^3	x^4	xy	x^2y
10	25	100	1000	10000	250	2500
20	70	400	8000	160000	1400	28000
30	380	900	27000	810000	11400	342000
40	550	1600	64000	2560000	22000	880000
50	610	2500	125000	6250000	30500	1525000
60	1220	3600	216000	12960000	73200	4392000
70	830	4900	343000	24010000	58100	4067000
80	1450	6400	512000	40960000	116000	9280000
Σ		20400	1296000	87720000	312850	20516500

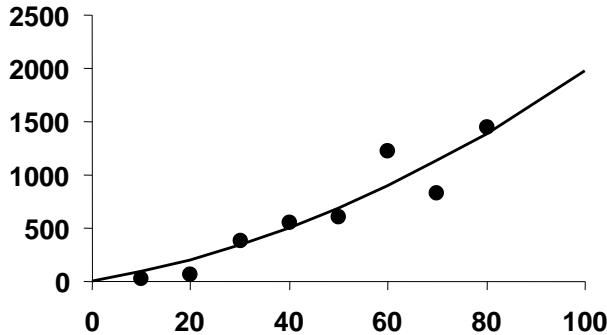
$$a_1 = \frac{312850(87720000) - 20516500(1296000)}{20400(87720000) - (1296000)^2} = 7.771024$$

$$a_2 = \frac{20400(20516500) - 312850(1296000)}{20400(87720000) - (1296000)} = 0.119075$$

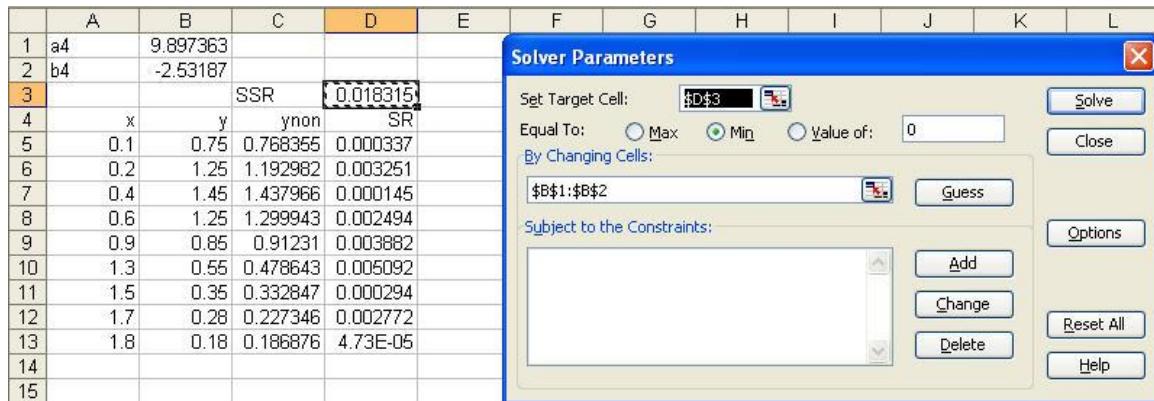
Therefore, the best-fit model is

$$y = 7.771024x + 0.119075x^2$$

The fit, along with the original data can be plotted as



17.29 We can use the Excel Solver to determine the fit.



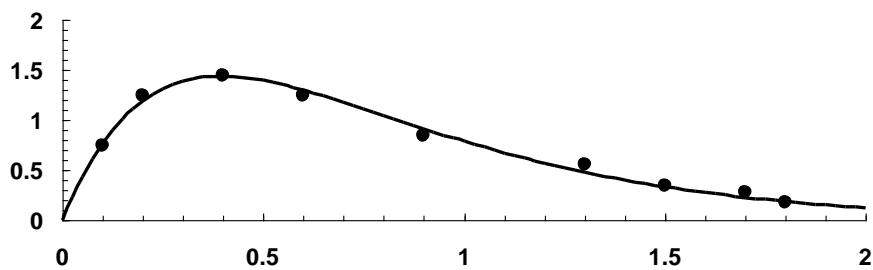
The cell formulas are

	A	B	C	D
1	a4	9.89736315656697		
2	b4	-2.53187086003733		
3		SSR	=SUM(D5:D13)	
4	x	y	ynon	SR
5	0.1	0.75	=\$B\$1*A5*EXP(\$B\$2*A5)	=(B5-C5)^2
6	0.2	1.25	=\$B\$1*A6*EXP(\$B\$2*A6)	=(B6-C6)^2
7	0.4	1.45	=\$B\$1*A7*EXP(\$B\$2*A7)	=(B7-C7)^2
8	0.6	1.25	=\$B\$1*A8*EXP(\$B\$2*A8)	=(B8-C8)^2
9	0.9	0.85	=\$B\$1*A9*EXP(\$B\$2*A9)	=(B9-C9)^2
10	1.3	0.55	=\$B\$1*A10*EXP(\$B\$2*A10)	=(B10-C10)^2
11	1.5	0.35	=\$B\$1*A11*EXP(\$B\$2*A11)	=(B11-C11)^2
12	1.7	0.28	=\$B\$1*A12*EXP(\$B\$2*A12)	=(B12-C12)^2
13	1.8	0.18	=\$B\$1*A13*EXP(\$B\$2*A13)	=(B13-C13)^2

Therefore, the best-fit model is

$$y = 9.8974xe^{-2.53187x}$$

The model and the data can be plotted as



PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

CHAPTER 18

18.1 (a)

$$f_1(10) = 0.90309 + \frac{1.0791812 - 0.90309}{12 - 8} (10 - 8) = 0.991136$$

$$\varepsilon_t = \frac{1 - 0.991136}{1} \times 100\% = 0.886\%$$

(b)

$$f_1(10) = 0.9542425 + \frac{1.0413927 - 0.9542425}{11 - 9} (10 - 9) = 0.997818$$

$$\varepsilon_t = \frac{1 - 0.997818}{1} \times 100\% = 0.218\%$$

18.2 First, order the points

$$\begin{array}{ll} x_0 = 9 & f(x_0) = 0.9542425 \\ x_1 = 11 & f(x_1) = 1.0413927 \\ x_2 = 8 & f(x_2) = 0.9030900 \end{array}$$

Applying Eq. (18.4)

$$b_0 = 0.9542425$$

Equation (18.5) yields

$$b_1 = \frac{1.0413927 - 0.9542425}{11 - 9} = 0.0435751$$

Equation (18.6) gives

$$b_2 = \frac{\frac{0.9030900 - 1.0413927}{8 - 11} - 0.0435751}{8 - 9} = \frac{\frac{0.0461009 - 0.0435751}{8 - 9}}{8 - 9} = -0.0025258$$

Substituting these values into Eq. (18.3) yields the quadratic formula

$$f_2(x) = 0.9542425 + 0.0435751(x - 9) - 0.0025258(x - 9)(x - 11)$$

which can be evaluated at $x = 10$ for

$$f_2(10) = 0.9542425 + 0.0435751(10 - 9) - 0.0025258(10 - 9)(10 - 11) = 1.0003434$$

18.3 First, order the points

$$\begin{array}{ll}
 x_0 = 9 & f(x_0) = 0.9542425 \\
 x_1 = 11 & f(x_1) = 1.0413927 \\
 x_2 = 8 & f(x_2) = 0.9030900 \\
 x_3 = 12 & f(x_3) = 1.0791812
 \end{array}$$

The first divided differences can be computed as

$$f[x_1, x_0] = \frac{1.0413927 - 0.9542425}{11 - 9} = 0.0435751$$

$$f[x_2, x_1] = \frac{0.9030900 - 1.0413927}{8 - 11} = 0.0461009$$

$$f[x_3, x_2] = \frac{1.0791812 - 0.9030900}{12 - 8} = 0.0440228$$

The second divided differences are

$$f[x_2, x_1, x_0] = \frac{0.0461009 - 0.0435751}{8 - 9} = -0.0025258$$

$$f[x_3, x_2, x_1] = \frac{0.0440228 - 0.0461009}{12 - 11} = -0.0020781$$

The third divided difference is

$$f[x_3, x_2, x_1, x_0] = \frac{-0.0020781 - (-0.0025258)}{12 - 9} = 0.00014924$$

Substituting the appropriate values into Eq. (18.7) gives

$$\begin{aligned}
 f_3(x) = & 0.9542425 + 0.0435751(x - 9) - 0.0025258(x - 9)(x - 11) \\
 & + 0.00014924(x - 9)(x - 11)(x - 8)
 \end{aligned}$$

which can be evaluated at $x = 10$ for

$$\begin{aligned}
 f_3(x) = & 0.9542425 + 0.0435751(10 - 9) - 0.0025258(10 - 9)(10 - 11) \\
 & + 0.00014924(10 - 9)(10 - 11)(10 - 8) = 1.0000449
 \end{aligned}$$

18.4 First, order the points so that they are as close to and as centered about the unknown as possible

$$\begin{array}{ll}
 x_0 = 2.5 & f(x_0) = 14 \\
 x_1 = 3.2 & f(x_1) = 15 \\
 x_2 = 2 & f(x_2) = 8 \\
 x_3 = 4 & f(x_3) = 8 \\
 x_4 = 1.6 & f(x_4) = 2
 \end{array}$$

Next, the divided differences can be computed and displayed in the format of Fig. 18.5,

i	x_i	$f(x_i)$	$f[x_{i+1}, x_i]$	$f[x_{i+2}, x_{i+1}, x_i]$	$f[x_{i+3}, x_{i+2}, x_{i+1}, x_i]$	$f[x_{i+4}, x_{i+3}, x_{i+2}, x_{i+1}, x_i]$
0	2.5	14	1.428571	-8.809524	1.011905	1.847718
1	3.2	15	5.833333	-7.291667	-0.651042	
2	2	8	0	-6.25		
3	4	8	2.5			
4	1.6	2				

The first through third-order interpolations can then be implemented as

$$f_1(2.8) = 14 + 1.428571(2.8 - 2.5) = 14.428571$$

$$f_2(2.8) = 14 + 1.428571(2.8 - 2.5) - 8.809524(2.8 - 2.5)(2.8 - 3.2) = 15.485714$$

$$f_3(2.8) = 14 + 1.428571(2.8 - 2.5) - 8.809524(2.8 - 2.5)(2.8 - 3.2) \\ + 1.011905(2.8 - 2.5)(2.8 - 3.2)(2.8 - 2) = 15.388571$$

The errors estimates for the first and second-order predictions can be computed with Eq. 18.19 as

$$R_1 = 15.485714 - 14.428571 = 1.057143$$

$$R_2 = 15.388571 - 15.485714 = -0.097143$$

The error for the third-order prediction can be computed with Eq. 18.18 as

$$R_3 = 1.847718(2.8 - 2.5)(2.8 - 3.2)(2.8 - 2)(2.8 - 4) = 0.212857$$

18.5 First, order the points so that they are as close to and as centered about the unknown as possible

$$\begin{array}{ll} x_0 = 3 & f(x_0) = 19 \\ x_1 = 5 & f(x_1) = 99 \\ x_2 = 2 & f(x_2) = 6 \\ x_3 = 7 & f(x_3) = 291 \\ x_4 = 1 & f(x_4) = 3 \end{array}$$

Next, the divided differences can be computed and displayed in the format of Fig. 18.5,

i	x_i	$f(x_i)$	$f[x_{i+1}, x_i]$	$f[x_{i+2}, x_{i+1}, x_i]$	$f[x_{i+3}, x_{i+2}, x_{i+1}, x_i]$	$f[x_{i+4}, x_{i+3}, x_{i+2}, x_{i+1}, x_i]$
0	3	19	40	9	1	0
1	5	99	31	13	1	
2	2	6	57	9		
3	7	291	48			
4	1	3				

The first through fourth-order interpolations can then be implemented as

$$f_1(4) = 19 + 40(4 - 3) = 59$$

$$f_2(4) = 59 + 9(4 - 3)(4 - 5) = 50$$

$$f_3(4) = 50 + 1(4 - 3)(4 - 5)(4 - 2) = 48$$

$$f_4(4) = 48 + 0(4 - 3)(4 - 5)(4 - 2)(4 - 7) = 48$$

Clearly this data was generated with a cubic polynomial since the difference between the 4th and the 3rd-order versions is zero.

18.6

18.1 (a):

$$\begin{aligned}x_0 &= 8 & f(x_0) &= 0.9030900 \\x_1 &= 12 & f(x_1) &= 1.0791812 \\f_1(10) &= \frac{10 - 12}{8 - 12} 0.9030900 + \frac{10 - 8}{12 - 8} 1.0791812 = 0.991136\end{aligned}$$

18.1 (b):

$$\begin{aligned}x_0 &= 9 & f(x_0) &= 0.9542425 \\x_1 &= 11 & f(x_1) &= 1.0413927 \\f_1(10) &= \frac{10 - 11}{9 - 11} 0.9542425 + \frac{10 - 9}{11 - 9} 1.0413927 = 0.997818\end{aligned}$$

18.2:

$$\begin{aligned}x_0 &= 8 & f(x_0) &= 0.9030900 \\x_1 &= 9 & f(x_1) &= 0.9542425 \\x_2 &= 11 & f(x_2) &= 1.0413927 \\x_3 &= 12 & f(x_3) &= 1.0791812 \\f_2(10) &= \frac{(10 - 9)(10 - 11)}{(8 - 9)(8 - 11)} 0.9030900 + \frac{(10 - 8)(10 - 11)}{(9 - 8)(9 - 11)} 0.9542425 \\&\quad + \frac{(10 - 8)(10 - 9)}{(11 - 8)(11 - 9)} 1.0413927 = 1.0003434\end{aligned}$$

18.3:

$$\begin{aligned}x_0 &= 8 & f(x_0) &= 0.9030900 \\x_1 &= 9 & f(x_1) &= 0.9542425 \\x_2 &= 11 & f(x_2) &= 1.0413927 \\x_3 &= 12 & f(x_3) &= 1.0791812 \\f_3(10) &= \frac{(10 - 9)(10 - 11)(10 - 12)}{(8 - 9)(8 - 11)(8 - 12)} 0.9030900 + \frac{(10 - 8)(10 - 11)(10 - 12)}{(9 - 8)(9 - 11)(9 - 12)} 0.9542425 \\&\quad + \frac{(10 - 8)(10 - 9)(10 - 12)}{(11 - 8)(11 - 9)(11 - 12)} 1.0413927 + \frac{(10 - 8)(10 - 9)(10 - 11)}{(12 - 8)(12 - 9)(12 - 11)} 1.0791812 = 1.0000449\end{aligned}$$

18.7

First order:

$$\begin{aligned}x_0 &= 3 & f(x_0) &= 19 \\x_1 &= 5 & f(x_1) &= 99\end{aligned}$$

$$f_1(10) = \frac{4-5}{3-5} 19 + \frac{4-3}{5-3} 99 = 59$$

Second order:

$$x_0 = 3 \quad f(x_0) = 19$$

$$x_1 = 5 \quad f(x_1) = 99$$

$$x_2 = 2 \quad f(x_2) = 6$$

$$f_2(10) = \frac{(4-5)(4-2)}{(3-5)(3-2)} 19 + \frac{(4-3)(4-2)}{(5-3)(5-2)} 99 + \frac{(4-3)(4-5)}{(2-3)(2-5)} 6 = 50$$

Third order:

$$x_0 = 3 \quad f(x_0) = 19$$

$$x_1 = 5 \quad f(x_1) = 99$$

$$x_2 = 2 \quad f(x_2) = 6$$

$$x_3 = 7 \quad f(x_3) = 291$$

$$f_3(10) = \frac{(4-5)(4-2)(4-7)}{(3-5)(3-2)(3-7)} 19 + \frac{(4-3)(4-2)(4-7)}{(5-3)(5-2)(5-7)} 99 \\ + \frac{(4-3)(4-5)(4-7)}{(2-3)(2-5)(2-7)} 6 + \frac{(4-3)(4-5)(4-2)}{(7-3)(7-5)(7-2)} 291 = 48$$

18.8 The following points are used to generate a cubic interpolating polynomial

$$x_0 = 3 \quad f(x_0) = 0.3333$$

$$x_1 = 4 \quad f(x_1) = 0.25$$

$$x_2 = 5 \quad f(x_2) = 0.2$$

$$x_3 = 6 \quad f(x_3) = 0.1667$$

The polynomial can be generated in a number of ways including simultaneous equations (Eq. 18.26) or a software tool. The result is

$$f_3(x) = 0.943 - 0.3261833x + 0.0491x^2 - 0.00271667x^3$$

The roots problem can then be developed by setting this polynomial equal to the desired value of 0.23

$$0 = 0.713 - 0.3261833x + 0.0491x^2 - 0.00271667x^3$$

Bisection can then be used to determine the root. Using initial guesses of $x_l = 4$ and $x_u = 5$, the first five iterations are

i	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l) \times f(x_r)$	ϵ_a
1	4.00000	5.00000	4.50000	0.02000	-0.00811	-0.00016	11.11%
2	4.00000	4.50000	4.25000	0.02000	0.00504	0.00010	5.88%
3	4.25000	4.50000	4.37500	0.00504	-0.00174	-0.00001	2.86%
4	4.25000	4.37500	4.31250	0.00504	0.00160	0.00001	1.45%
5	4.31250	4.37500	4.34375	0.00160	-0.00009	0.00000	0.72%

If the iterations are continued, the final result is $x = 4.34213$.

18.9 [Errata: Note that in the first printing, the function for this problem was erroneously shown as $f(x) = x^3/(2 + x^3)$. The correct formula, which appears in subsequent printings is $f(x) = x^2/(1 + x^2)$.]

(a) Analytically

$$\begin{aligned} 0.85 &= \frac{x^2}{1+x^2} \\ 0.85 + 0.85x^2 &= x^2 \\ x &= \sqrt{0.85/0.15} = 2.380476 \end{aligned}$$

(b) Cubic interpolation of x versus y

$$\begin{array}{ll} y_0 = 0.5 & x_0 = 1 \\ y_1 = 0.8 & x_1 = 2 \\ y_2 = 0.9 & x_2 = 3 \\ y_3 = 0.941176 & x_3 = 4 \end{array}$$

The polynomial can be generated as

$$x = -62.971 + 282.46y - 404.83y^2 + 191.59y^3$$

This function can then be used to compute

$$x = -62.971 + 282.46(0.85) - 404.83(0.85)^2 + 191.59(0.85)^3 = 2.290694$$

$$\varepsilon_t = \left| \frac{2.380476 - 2.290694}{2.380476} \right| \times 100\% = 3.77\%$$

(c) Quadratic interpolation of y versus x yields

$$\begin{array}{ll} x_0 = 2 & f(x_0) = 0.8 \\ x_1 = 3 & f(x_1) = 0.9 \\ x_2 = 4 & f(x_2) = 0.941176 \end{array}$$

The polynomial can be generated as

$$f_2(x) = 0.423529 + 0.247059x - 0.029412x^2$$

The roots problem can then be developed by setting this polynomial equal to the desired value of 0.85 to give

$$f_2(x) = -0.42647 + 0.247059x - 0.029412x^2$$

The quadratic formula can then be used to determine the root as

$$x = \frac{-0.247059 + \sqrt{(-0.247059)^2 - 4(-0.42647)(-0.029412)}}{2(-0.029412)} = 2.428009$$

$$\varepsilon_t = \left| \frac{2.380476 - 2.428009}{2.380476} \right| \times 100\% = 2.00\%$$

(d) Cubic interpolation of y versus x yields

$$\begin{array}{ll} x_0 = 1 & f(x_0) = 0.5 \\ x_1 = 2 & f(x_1) = 0.8 \\ x_2 = 3 & f(x_2) = 0.9 \\ x_3 = 4 & f(x_3) = 0.941176 \end{array}$$

The polynomial can be generated as

$$f_3(x) = -0.14118 + 0.858824x - 0.24118x^2 + 0.023529x^3$$

The roots problem can then be developed by setting this polynomial equal to the desired value of 0.85

$$f_3(x) = -0.99118 + 0.858824x - 0.24118x^2 + 0.023529x^3$$

Bisection can then be used to determine the root. Using initial guesses of $x_l = 2$ and $x_u = 3$, the first five iterations are

<i>i</i>	x_l	x_u	x_r	$f(x_l)$	$f(x_r)$	$f(x_l) \times f(x_r)$	ε_a
1	2.00000	3.00000	2.50000	-0.05000	0.01617	-0.00081	20.00%
2	2.00000	2.50000	2.25000	-0.05000	-0.01177	0.00059	11.11%
3	2.25000	2.50000	2.37500	-0.01177	0.00335	-0.00004	5.26%
4	2.25000	2.37500	2.31250	-0.01177	-0.00390	0.00005	2.70%
5	2.31250	2.37500	2.34375	-0.00390	-0.00020	0.00000	1.33%

If the iterations are continued, the final result is $x = 2.345481$.

$$\varepsilon_t = \left| \frac{2.380476 - 2.345481}{2.380476} \right| \times 100\% = 1.47\%$$

18.10 For the present problem, we have five data points and $n = 4$ intervals. Therefore, $3(4) = 12$ unknowns must be determined. Equations 18.29 and 18.30 yield $2(4) - 2 = 6$ conditions

$$\begin{aligned} 4a_1 + 2b_1 + c_1 &= 8 \\ 4a_2 + 2b_2 + c_2 &= 8 \\ 6.25a_2 + 2.5b_2 + c_2 &= 14 \end{aligned}$$

$$\begin{aligned} 6.25a_3 + 2.5b_3 + c_3 &= 14 \\ 10.24a_3 + 3.2b_3 + c_3 &= 15 \\ 10.24a_4 + 3.2b_4 + c_4 &= 15 \end{aligned}$$

Passing the first and last functions through the initial and final values adds 2 more

$$\begin{aligned} 2.56a_1 + 1.6b_1 + c_1 &= 2 \\ 16a_4 + 4b_4 + c_4 &= 8 \end{aligned}$$

Continuity of derivatives creates an additional $4 - 1 = 3$.

$$\begin{aligned} 4a_1 + b_1 &= 4a_2 + b_2 \\ 5a_2 + b_2 &= 5a_3 + b_3 \\ 6.4a_3 + b_3 &= 6.4a_4 + b_4 \end{aligned}$$

Finally, Eq. 18.34 specifies that $a_1 = 0$. Thus, the problem reduces to solving 11 simultaneous equations for 11 unknown coefficients,

$$\left[\begin{array}{cccccccccc|c} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b_1 \\ 0 & 0 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & c_1 \\ 0 & 0 & 6.25 & 2.5 & 1 & 0 & 0 & 0 & 0 & 0 & a_2 \\ 0 & 0 & 0 & 0 & 0 & 6.25 & 2.5 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 0 & 0 & 10.24 & 3.2 & 1 & 0 & 0 & c_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10.24 & 3.2 & a_3 \\ 1.6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 4 & c_3 \\ 1 & 0 & -4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & a_4 \\ 0 & 0 & 5 & 1 & 0 & -5 & -1 & 0 & 0 & 0 & b_4 \\ 0 & 0 & 0 & 0 & 0 & 6.4 & 1 & 0 & -6.4 & -1 & c_4 \end{array} \right] = \begin{Bmatrix} 8 \\ 8 \\ 14 \\ 14 \\ 15 \\ 15 \\ 15 \\ 2 \\ 8 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

which can be solved for

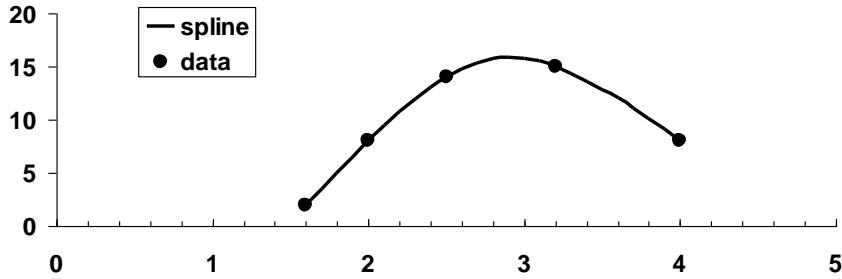
$$\begin{aligned} b_1 &= 15 & c_1 &= -22 \\ a_2 &= -6 & b_2 &= 39 & c_2 &= -46 \\ a_3 &= -10.816327 & b_3 &= 63.081633 & c_3 &= -76.102041 \\ a_4 &= -3.258929 & b_4 &= 14.714286 & c_4 &= 1.285714 \end{aligned}$$

The predictions can be made as

$$f(3.4) = -3.258929(3.4)^2 + 14.714286(3.4) + 1.285714 = 13.64107$$

$$f(2.2) = -6(2.2)^2 + 39(2.2) - 46 = 10.76$$

Finally, here is a plot of the data along with the quadratic spline,



18.11 For the first interior knot

$$\begin{aligned}x_0 &= 1 & f(x_0) &= 3 \\x_1 &= 2 & f(x_1) &= 6 \\x_2 &= 3 & f(x_2) &= 19\end{aligned}$$

$$(2-1)f''(1) + 2(3-1)f''(2) + (3-2)f''(3) = \frac{6}{3-2}(19-6) + \frac{6}{2-1}(3-6)$$

Because of the natural spline condition, $f''(1) = 0$, and the equation reduces to

$$4f''(2) + f''(3) = 60$$

Equations can be written for the remaining interior knots and the results assembled in matrix form as

$$\begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 6 & 2 & 0 \\ 0 & 2 & 8 & 2 \\ 0 & 0 & 2 & 6 \end{bmatrix} \begin{Bmatrix} f''(2) \\ f''(3) \\ f''(5) \\ f''(7) \end{Bmatrix} = \begin{Bmatrix} 60 \\ 162 \\ 336 \\ 342 \end{Bmatrix}$$

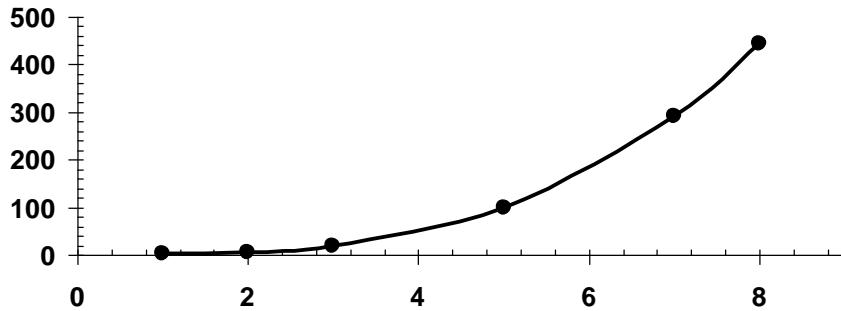
which can be solved for

$$\begin{aligned}f''(2) &= 10.84716 \\f''(3) &= 16.61135 \\f''(5) &= 25.74236 \\f''(7) &= 48.41921\end{aligned}$$

These values can be used in conjunction with Eq. 18.36 to yield the following interpolating splines for each interval,

$$\begin{aligned}f_1(x) &= 1.80786(x-1)^3 + 3(2-x) + 4.19214(x-1) \\f_2(x) &= 1.80786(3-x)^3 + 2.768559(x-2)^3 + 4.19214(3-x) + 16.23144(x-2) \\f_3(x) &= 1.384279(5-x)^3 + 2.145197(x-3)^3 + 3.962882(5-x) + 40.91921(x-3) \\f_4(x) &= 2.145197(7-x)^3 + 4.034934(x-5)^3 + 40.91921(7-x) + 129.3603(x-5) \\f_5(x) &= 8.069869(8-x)^3 + 282.9301(8-x) + 444(x-7)\end{aligned}$$

The interpolating splines can be used to make predictions along the interval. The results are shown in the following plot.



(a) The interpolating equations can be used to determine

$$f_3(4) = 48.41157$$

$$f_2(2.5) = 10.78384$$

(b)

$$f_2(3) = 1.80786(3-3)^3 + 2.768559(3-2)^3 + 4.19214(3-3) + 16.23144(3-2) = 19$$

$$f_3(3) = 1.384279(5-3)^3 + 2.145197(3-3)^3 + 3.962882(5-3) + 40.91921(3-3) = 19$$

18.12 The points to be fit are

$$x_0 = 3.2 \quad f(x_0) = 15$$

$$x_1 = 4 \quad f(x_1) = 8$$

$$x_2 = 4.5 \quad f(x_2) = 2$$

Using Eq. 18.26 the following simultaneous equations can be generated

$$a_0 + 3.2a_1 + 10.24a_2 = 15$$

$$a_0 + 4a_1 + 16a_2 = 8$$

$$a_0 + 4.5a_1 + 20.25a_2 = 2$$

These can be solved for $a_0 = 11$, $a_1 = 9.25$, and $a_2 = -2.5$. Therefore, the interpolating polynomial is

$$f(x) = 11 + 9.25x - 2.5x^2$$

18.13 The points to be fit are

$$x_0 = 1 \quad f(x_0) = 3$$

$$x_1 = 2 \quad f(x_1) = 6$$

$$x_2 = 3 \quad f(x_2) = 19$$

$$x_3 = 5 \quad f(x_3) = 99$$

Using Eq. 18.26 the following simultaneous equations can be generated

$$\begin{aligned}a_0 + a_1 + a_2 + a_3 &= 3 \\a_0 + 2a_1 + 4a_2 + 8a_3 &= 6 \\a_0 + 3a_1 + 9a_2 + 27a_3 &= 19 \\a_0 + 5a_1 + 25a_2 + 125a_3 &= 99\end{aligned}$$

These can be solved for $a_0 = 4$, $a_1 = -1$, $a_2 = -1$, and $a_3 = 1$. Therefore, the interpolating polynomial is

$$f(x) = 4 - x - x^2 + x^3$$

18.14 Here is a VBA/Excel program to implement Newton interpolation.

```
Option Explicit

Sub Newt()
    Dim n As Integer, i As Integer
    Dim yint(10) As Double, x(10) As Double, y(10) As Double
    Dim ea(10) As Double, xi As Double
    Sheets("Sheet1").Select
    Range("a5").Select
    n = ActiveCell.Row
    Selection.End(xlDown).Select
    n = ActiveCell.Row - n
    Range("a5").Select
    For i = 0 To n
        x(i) = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        y(i) = ActiveCell.Value
        ActiveCell.Offset(1, -1).Select
    Next i
    Range("e3").Select
    xi = ActiveCell.Value
    Call Newtint(x, y, n, xi, yint, ea)
    Range("d5:f25").ClearContents
    Range("d5").Select
    For i = 0 To n
        ActiveCell.Value = i
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yint(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = ea(i)
        ActiveCell.Offset(1, -2).Select
    Next i
    Range("a5").Select
End Sub

Sub Newtint(x, y, n, xi, yint, ea)
    Dim i As Integer, j As Integer, order As Integer
    Dim fdd(10, 10) As Double, xterm As Double
    Dim yint2 As Double
    For i = 0 To n
        fdd(i, 0) = y(i)
    Next i
    For j = 1 To n
```

```

For i = 0 To n - j
    fdd(i, j) = (fdd(i + 1, j - 1) - fdd(i, j - 1)) / (x(i + j) - x(i))
Next i
Next j
xterm = 1#
yint(0) = fdd(0, 0)
For order = 1 To n
    xterm = xterm * (xi - x(order - 1))
    yint2 = yint(order - 1) + fdd(0, order) * xterm
    ea(order - 1) = yint2 - yint(order - 1)
    yint(order) = yint2
Next order
End Sub

```

For those who are using MATLAB, here is an M-file that implements Newton interpolation:

```

function yint = Newtint(x,y,xx)
% yint = Newtint(x,y,xx):
%   Newton interpolation. Uses an (n - 1)-order Newton
%   interpolating polynomial based on n data points (x, y)
%   to determine a value of the dependent variable (yint)
%   at a given value of the independent variable, xx.
% input:
%   x = independent variable
%   y = dependent variable
%   xx = value of independent variable at which
%         interpolation is calculated
% output:
%   yint = interpolated value of dependent variable

% compute the finite divided differences in the form of a
% difference table
n = length(x);
if length(y) ~= n, error('x and y must be same length'); end
b = zeros(n,n);
% assign dependent variables to the first column of b.
b(:,1) = y(:,1); % the (:) ensures that y is a column vector.
for j = 2:n
    for i = 1:n-j+1
        b(i,j) = (b(i+1,j-1)-b(i,j-1))/(x(i+j-1)-x(i));
    end
end
% use the finite divided differences to interpolate
xt = 1;
yint = b(1,1);
for j = 1:n-1
    xt = xt*(xx-x(j));
    yint = yint+b(1,j+1)*xt;
end

```

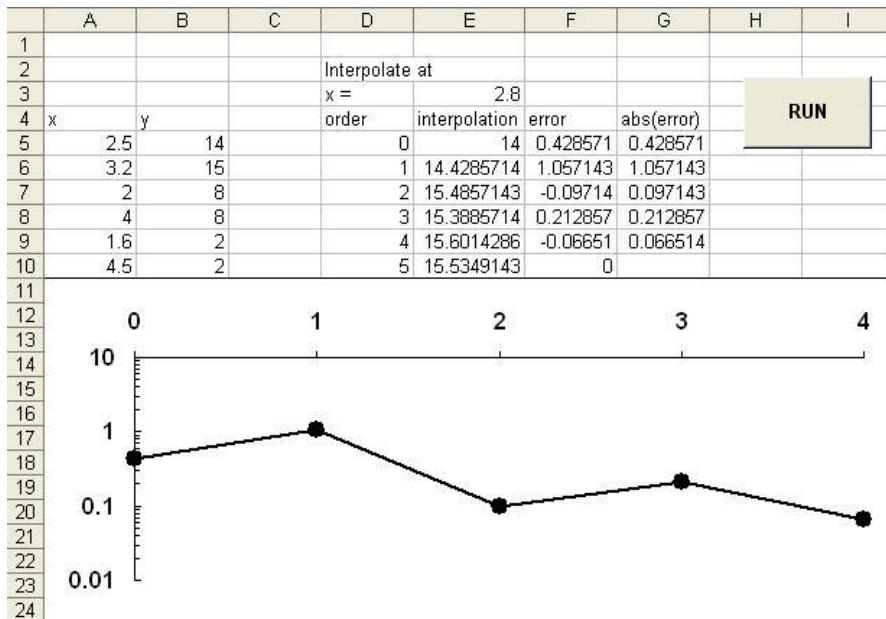
18.15 Here is the solution when the Excel VBA program from Prob. 18.14 is run for Example 18.5.

	A	B	C	D	E	F	G	H	I
1									
2					Interpolate at				
3				x =	2				
4	x	y		order	interpolation	error		RUN	
5	1	0		0	0	0.462098			
6	4	1.386294		1	0.46209812	0.103746			
7	6	1.791759		2	0.565844347	0.062924			
8	5	1.609438		3	0.628768579	0.046953			
9	3	1.098612		4	0.675721874	0.021792			
10	1.5	0.405465		5	0.697514122	-0.00362			
11	2.5	0.916291		6	0.693897701	-0.00046			
12	3.5	1.252763		7	0.693438714	0			

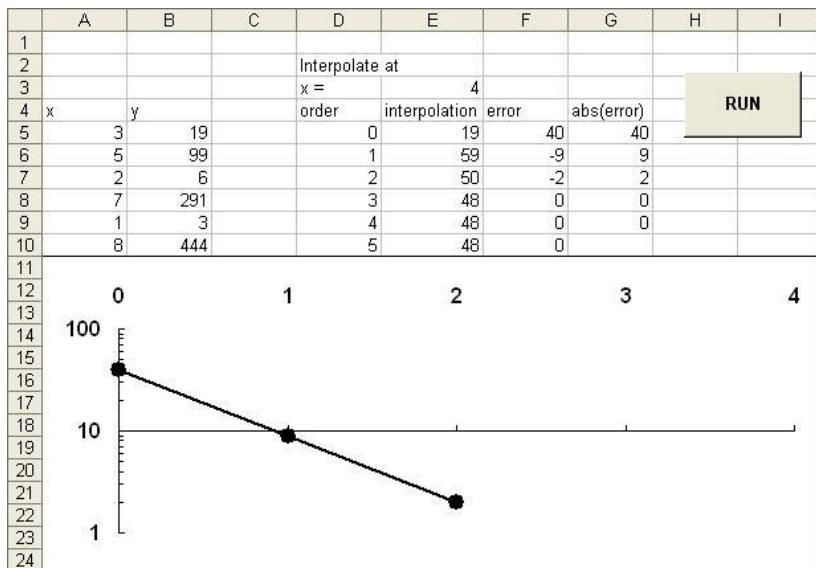
18.16 Here are the solutions when the program from Prob. 18.14 is run for Probs. 18.1 through 18.3.

	A	B	C	D	E	F	G	H	I
1									
2					Interpolate at				
3				x =	10				
4	x	y		order	interpolation	error		RUN	
5	9	0.954243		0	0.954242509	0.043575			
6	11	1.041393		1	0.997817597	0.002526			
7	8	0.90309		2	1.000343409	-0.0003			
8	12	1.079181		3	1.000044924	0			

18.17 A plot of the error can easily be added to the Excel application. Note that we plot the absolute value of the error on a logarithmic scale. The following shows the solution for Prob. 18.4:



The following shows the solution for Prob. 18.5:

**18.18**

Option Explicit

```

Sub LagrInt()
    Dim n As Integer, i As Integer, order As Integer
    Dim x(10) As Double, y(10) As Double, xi As Double
    Range("a5").Select
    n = ActiveCell.Row
    Selection.End(xlDown).Select
    n = ActiveCell.Row - n
    Range("a5").Select
    For i = 0 To n
        x(i) = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        y(i) = ActiveCell.Value
        ActiveCell.Offset(1, -1).Select
    Next i
    Range("e3").Select
    order = ActiveCell.Value
    ActiveCell.Offset(1, 0).Select
    xi = ActiveCell.Value
    ActiveCell.Offset(2, 0).Select
    ActiveCell.Value = Lagrange(x, y, order, xi)
End Sub

Function Lagrange(x, y, order, xi)
    Dim i As Integer, j As Integer
    Dim sum As Double, prod As Double
    sum = 0#
    For i = 0 To order
        prod = y(i)
        For j = 0 To order
            If i <> j Then
                prod = prod * (xi - x(j)) / (x(i) - x(j))
            End If
        Next j
        sum = sum + prod
    Next i
    Lagrange = sum
End Function

```

End Function

Application to Example 18.7:

	A	B	C	D	E	F	G
1	Example 18.7						
2							
3	Data:			Order =	2		
4	x	y		Interpolate at x =	10	RUN	
5	13	4755					
6	7	3940		f(x) =	4672.813		
7	5	3090					
8	3	2310					
9	1	800					

For those who are using MATLAB, here is an M-file that implements Lagrange interpolation:

```
function yint = Lagrange(x,y,xx)
% yint = Lagrange(x,y,xx):
%   Lagrange interpolation. Uses an (n - 1)-order Lagrange
%   interpolating polynomial based on n data points (x, y)
%   to determine a value of the dependent variable (yint)
%   at a given value of the independent variable, xx.
% input:
%   x = independent variable
%   y = dependent variable
%   xx = value of independent variable at which the
%         interpolation is calculated
% output:
%   yint = interpolated value of dependent variable

n = length(x);
if length(y)~==n, error('x and y must be same length'); end
s = 0;
for i = 1:n
    product = y(i);
    for j = 1:n
        if i ~= j
            product = product*(xx-x(j))/(x(i)-x(j));
        end
    end
    s = s+product;
end
yint = s;
```

18.19 The following VBA program uses cubic interpolation for all intervals:

```
Option Explicit

Sub TableLook()
Dim n As Integer, i As Integer
Dim yint(10) As Double, x(10) As Double, y(10) As Double
Dim ea(10) As Double, xi As Double
Range("a5").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
Range("a5").Select
For i = 0 To n
```

```

x(i) = ActiveCell.Value
ActiveCell.Offset(0, 1).Select
y(i) = ActiveCell.Value
ActiveCell.Offset(1, -1).Select
Next i
Range("e4").Select
xi = ActiveCell.Value
ActiveCell.Offset(2, 0).Select
ActiveCell.Value = Interp(x, y, n, xi)
Range("a5").Select
End Sub

Function Interp(x, y, n, xx)
Dim ii As Integer
If xx < x(0) Or xx > x(n) Then
    Interp = "out of range"
Else
    If xx <= x(ii + 1) Then
        Interp = Lagrange(x, y, 0, 3, xx)
    ElseIf xx <= x(n - 1) Then
        For ii = 0 To n - 2
            If xx >= x(ii) And xx <= x(ii + 1) Then
                Interp = Lagrange(x, y, ii - 1, 3, xx)
                Exit For
            End If
        Next ii
    Else
        Interp = Lagrange(x, y, n - 3, 3, xx)
    End If
End If
End Function

Function Lagrange(x, y, i0, order, xi)
Dim i As Integer, j As Integer
Dim sum As Double, prod As Double
sum = 0#
For i = i0 To i0 + order
    prod = y(i)
    For j = i0 To i0 + order
        If i <> j Then
            prod = prod * (xi - x(j)) / (x(i) - x(j))
        End If
    Next j
    sum = sum + prod
Next i
Lagrange = sum
End Function

```

Application to evaluate $\ln(2.5)$:

	A	B	C	D	E	F
1	Problem 18.4					
2						RUN
3	Data:					
4	x	y		Interpolation at x =	2.5	
5		1 0				
6		2 0.693147		f(x) =	0.921221304	
7		3 1.098612				
8		4 1.386294		True value	0.916290732	
9		5 1.609438				
10		6 1.791759		Error	0.53810 %	
11		7 1.94591				
12		8 2.079442				
13		9 2.197225				
14		10 2.302585				

18.20

Option Explicit

```

Sub Splines()
Dim i As Integer, n As Integer
Dim x(7) As Double, y(7) As Double, xu As Double, yu As Double
Dim dy As Double, d2y As Double
Range("a5").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
Range("a5").Select
For i = 0 To n
    x(i) = ActiveCell.Value
    ActiveCell.Offset(0, 1).Select
    y(i) = ActiveCell.Value
    ActiveCell.Offset(1, -1).Select
Next i
Range("e4").Select
xu = ActiveCell.Value
Call Spline(x, y, n, xu, yu, dy, d2y)
ActiveCell.Offset(2, 0).Select
ActiveCell.Value = yu
End Sub

Sub Spline(x, y, n, xu, yu, dy, d2y)
Dim e(10) As Double, f(10) As Double, g(10) As Double, r(10) As Double,
d2x(10) As Double
Call Tridiag(x, y, n, e, f, g, r)
Call Decomp(e, f, g, n - 1)
Call Substit(e, f, g, r, n - 1, d2x)
Call Interpol(x, y, n, d2x, xu, yu, dy, d2y)
End Sub

Sub Tridiag(x, y, n, e, f, g, r)
Dim i As Integer
f(1) = 2 * (x(2) - x(0))
g(1) = x(2) - x(1)
r(1) = 6 / (x(2) - x(1)) * (y(2) - y(1))
r(1) = r(1) + 6 / (x(1) - x(0)) * (y(0) - y(1))
For i = 2 To n - 2
    e(i) = x(i) - x(i - 1)
    f(i) = 2 * (x(i + 1) - x(i - 1))
    g(i) = x(i + 1) - x(i)
    r(i) = 6 / (x(i + 1) - x(i)) * (y(i + 1) - y(i))
    r(i) = r(i) + 6 / (x(i) - x(i - 1)) * (y(i - 1) - y(i))
Next i
e(n - 1) = x(n) - x(n - 1)
f(n - 1) = 2 * (x(n) - x(n - 1))
g(n - 1) = x(n) - x(n - 1)
r(n - 1) = 6 / (x(n) - x(n - 1)) * (y(n) - y(n - 1))
End Sub

```

```

Next i
e(n - 1) = x(n - 1) - x(n - 2)
f(n - 1) = 2 * (x(n) - x(n - 2))
r(n - 1) = 6 / (x(n) - x(n - 1)) * (y(n) - y(n - 1))
r(n - 1) = r(n - 1) + 6 / (x(n - 1) - x(n - 2)) * (y(n - 2) - y(n - 1))
End Sub

Sub Interpol(x, y, n, d2x, xu, yu, dy, d2y)
Dim i As Integer, flag As Integer
Dim c1 As Double, c2 As Double, c3 As Double, c4 As Double
Dim t1 As Double, t2 As Double, t3 As Double, t4 As Double
flag = 0
i = 1
Do
    If xu >= x(i - 1) And xu <= x(i) Then
        c1 = d2x(i - 1) / 6 / (x(i) - x(i - 1))
        c2 = d2x(i) / 6 / (x(i) - x(i - 1))
        c3 = y(i - 1) / (x(i) - x(i - 1)) - d2x(i - 1) * (x(i) - x(i - 1)) / 6
        c4 = y(i) / (x(i) - x(i - 1)) - d2x(i) * (x(i) - x(i - 1)) / 6
        t1 = c1 * (x(i) - xu) ^ 3
        t2 = c2 * (xu - x(i - 1)) ^ 3
        t3 = c3 * (x(i) - xu)
        t4 = c4 * (xu - x(i - 1))
        yu = t1 + t2 + t3 + t4
        t1 = -3 * c1 * (x(i) - xu) ^ 2
        t2 = 3 * c2 * (xu - x(i - 1)) ^ 2
        t3 = -c3
        t4 = c4
        dy = t1 + t2 + t3 + t4
        t1 = 6 * c1 * (x(i) - xu)
        t2 = 6 * c2 * (xu - x(i - 1))
        d2y = t1 + t2
        flag = 1
    Else
        i = i + 1
    End If
    If i = n + 1 Or flag = 1 Then Exit Do
Loop
If flag = 0 Then
    MsgBox "outside range"
End
End If
End Sub

Sub Decomp(e, f, g, n)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
End Sub

Sub Substit(e, f, g, r, n, x)
Dim k As Integer
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

	A	B	C	D	E	F	G
1	Example 18.10						
2							
3	Data:						
4	x	y		Interpolation at x =	5		
5	3	2.5					RUN
6	4.5	1		f(x) =	1.10289		
7	7	2.5					
8	9	0.5					

18.21 The following shows the solution for Prob. 18.4:

	A	B	C	D	E	F	G
1	Problem 18.4						
2							
3	Data:						
4	x	y		Interpolation at x =	2.25		
5	1.6	2					RUN
6	2	8		f(x) =	11.39693		
7	2.5	14					
8	3.2	15					
9	4	8					
10	4.5	2					

The following shows the solution for Prob. 18.5:

	A	B	C	D	E	F	G
1	Problem 18.5						
2							
3	Data:						
4	x	y		Interpolation at x =	2.25		
5	1	3					RUN
6	2	6		f(x) =	8.007915		
7	3	19					
8	5	99					
9	7	291					
10	8	444					

18.22 (a) Linear interpolation:

$$s = 6.4147 + \frac{6.5453 - 6.4147}{0.11144 - 0.10377} (0.108 - 0.10377) = 6.486726$$

(b) Quadratic interpolation:

$$s = 6.486726 + \frac{15.83811 - 17.02738}{0.1254 - 0.10377} (0.108 - 0.10377)(0.108 - 0.11144) = 6.487526$$

(c) Inverse interpolation. First, we fit a quadratic to the data

$$f_2(v) = 4.011945 + 28.860154v - 54.982456v^2$$

The roots problem can then be developed by setting this polynomial equal to the desired value of 6.6 to give

$$f_2(v) = -2.588055 + 28.860154v - 54.982456v^2$$

The quadratic formula can then be used to determine the root as

$$v = \frac{-28.860154 + \sqrt{(28.860154)^2 - 4(-54.982456)(-2.588055)}}{2(-54.982456)} = 0.11477$$

CHAPTER 19

19.1 The angular frequency can be computed as $\omega_0 = 2\pi/24 = 0.261799$. The various summations required for the normal equations can be set up as

t	y	cos($\omega_0 t$)	sin($\omega_0 t$)	sin($\omega_0 t$)cos($\omega_0 t$)	cos²($\omega_0 t$)	sin²($\omega_0 t$)	ycos($\omega_0 t$)	ysin($\omega_0 t$)
0	7.6	1.00000	0.00000	0.00000	1.00000	0.00000	7.60000	0.00000
2	7.2	0.86603	0.50000	0.43301	0.75000	0.25000	6.23538	3.60000
4	7.0	0.50000	0.86603	0.43301	0.25000	0.75000	3.50000	6.06218
5	6.5	0.25882	0.96593	0.25000	0.06699	0.93301	1.68232	6.27852
7	7.5	-0.25882	0.96593	-0.25000	0.06699	0.93301	-1.94114	7.24444
9	7.2	-0.70711	0.70711	-0.50000	0.50000	0.50000	-5.09117	5.09117
12	8.9	-1.00000	0.00000	0.00000	1.00000	0.00000	-8.90000	0.00000
15	9.1	-0.70711	-0.70711	0.50000	0.50000	0.50000	-6.43467	-6.43467
20	8.9	0.50000	-0.86603	-0.43301	0.25000	0.75000	4.45000	-7.70763
22	7.9	0.86603	-0.50000	-0.43301	0.75000	0.25000	6.84160	-3.95000
24	7.0	1.00000	0.00000	0.00000	1.00000	0.00000	7.00000	0.00000
sum→	84.8	2.31784	1.93185	0.00000	6.13397	4.86603	14.94232	10.18401

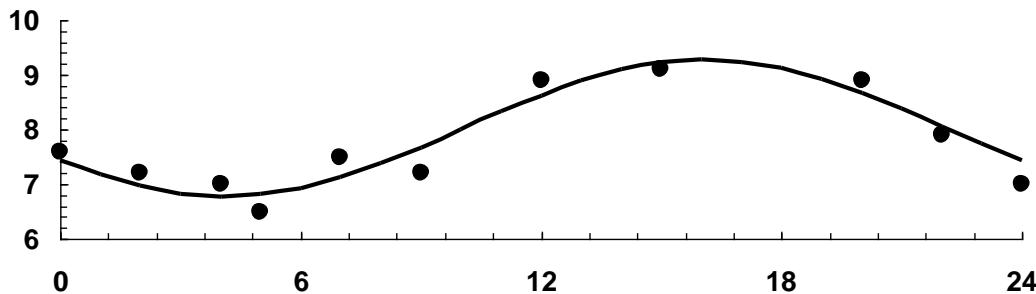
The normal equations can be assembled as

$$\begin{bmatrix} 11 & 2.317837 & 1.931852 \\ 2.3178 & 6.133975 & 0 \\ 1.931852 & 0 & 4.866025 \end{bmatrix} \begin{Bmatrix} A_0 \\ A_1 \\ B_1 \end{Bmatrix} = \begin{Bmatrix} 84.8 \\ 14.94232 \\ 10.18401 \end{Bmatrix}$$

This system can be solved for $A_0 = 8.02704$, $A_1 = -0.59717$, and $B_1 = -1.09392$. Therefore, the best-fit sinusoid is

$$pH = 8.02704 - 0.59717 \cos(\omega_0 t) - 1.09392 \sin(\omega_0 t)$$

The data and the model can be plotted as



19.2 The angular frequency can be computed as $\omega_0 = 2\pi/360 = 0.017453$. Because the data are equispaced, the coefficients can be determined with Eqs. 19.14-19.16. The various summations required to set up the model can be determined as

t	Radiation	$\cos(\omega_0 t)$	$\sin(\omega_0 t)$	$y\cos(\omega_0 t)$	$y\sin(\omega_0 t)$
15	144	0.96593	0.25882	139.093	37.270
45	188	0.70711	0.70711	132.936	132.936
75	245	0.25882	0.96593	63.411	236.652
105	311	-0.25882	0.96593	-80.493	300.403
135	351	-0.70711	0.70711	-248.194	248.194
165	359	-0.96593	0.25882	-346.767	92.916
195	308	-0.96593	-0.25882	-297.505	-79.716
225	287	-0.70711	-0.70711	-202.940	-202.940
255	260	-0.25882	-0.96593	-67.293	-251.141
285	211	0.25882	-0.96593	54.611	-203.810
315	159	0.70711	-0.70711	112.430	-112.430
345	131	0.96593	-0.25882	126.536	-33.905
sum→	2954			-614.175	164.429

The coefficients can be determined as

$$A_0 = \frac{\Sigma y}{N} = \frac{2954}{12} = 246.1667$$

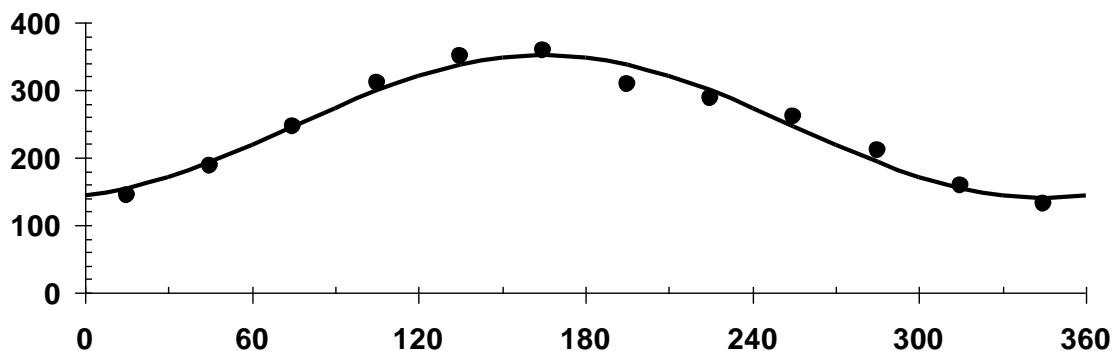
$$A_1 = \frac{2}{N} \Sigma y \cos(\omega_0 t) = \frac{2}{12} (-614.175) = -102.363$$

$$B_1 = \frac{2}{N} \Sigma y \sin(\omega_0 t) = \frac{2}{12} (164.429) = 27.4048$$

Therefore, the best-fit sinusoid is

$$R = 246.1667 - 102.363 \cos(0.017453t) + 27.4048 \sin(0.017453t)$$

The data and the model can be plotted as



The value for mid-August can be computed as

$$R = 246.1667 - 102.363 \cos(0.017453(225)) + 27.4048 \sin(0.017453(225)) = 299.1698$$

19.3 In the following equations, $\omega_0 = 2\pi/T$

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$\frac{\int_0^T \sin(\omega_0 t) dt}{T} = \frac{-\omega_0 [\cos(\omega_0 t)]_0^T}{T} = \frac{-\omega_0}{T} (\cos 2\pi - \cos 0) = 0$$

$$\frac{\int_0^T \cos(\omega_0 t) dt}{T} = \frac{\omega_0 [\sin(\omega_0 t)]_0^T}{T} = \frac{\omega_0}{T} (\sin 2\pi - \sin 0) = 0$$

$$\frac{\int_0^T \sin^2(\omega_0 t) dt}{T} = \frac{\left[\frac{t}{2} - \frac{\sin(2\omega_0 t)}{4\omega_0} \right]_0^T}{T} = \frac{\frac{T}{2} - \frac{\sin 4\pi}{4\omega_0} - 0 + 0}{T} = \frac{1}{2}$$

$$\frac{\int_0^T \cos^2(\omega_0 t) dt}{T} = \frac{\left[\frac{t}{2} + \frac{\sin(2\omega_0 t)}{4\omega_0} \right]_0^T}{T} = \frac{\frac{T}{2} + \frac{\sin 4\pi}{4\omega_0} - 0 - 0}{T} = \frac{1}{2}$$

$$\frac{\int_0^T \cos(\omega_0 t) \sin(\omega_0 t) dt}{T} = \left[\frac{\sin^2(\omega_0 t)}{2T\omega_0} \right]_0^T = \frac{\sin^2 2\pi}{2\omega_0 T} - 0 = 0$$

19.4 $a_0 = 0$

$$\begin{aligned} a_k &= \frac{2}{T} \int_{-T/2}^{T/2} -2t \cos(k\omega_0 t) dt \\ &= -\frac{4}{T} \left[\frac{1}{(k\omega_0)^2} \cos(k\omega_0 t) + \frac{t}{k\omega_0} \sin(k\omega_0 t) \right]_{-T/2}^{T/2} \end{aligned}$$

$$\begin{aligned} b_k &= \frac{2}{T} \int_{-T/2}^{T/2} -2t \sin(k\omega_0 t) dt \\ &= -\frac{4}{T} \left[\frac{1}{(k\omega_0)^2} \sin(k\omega_0 t) - \frac{t}{k\omega_0} \cos(k\omega_0 t) \right]_{-T/2}^{T/2} \end{aligned}$$

On the basis of these, all a 's = 0. For $k = \text{odd}$,

$$b_k = \frac{2}{k\pi}$$

For $k = \text{even}$,

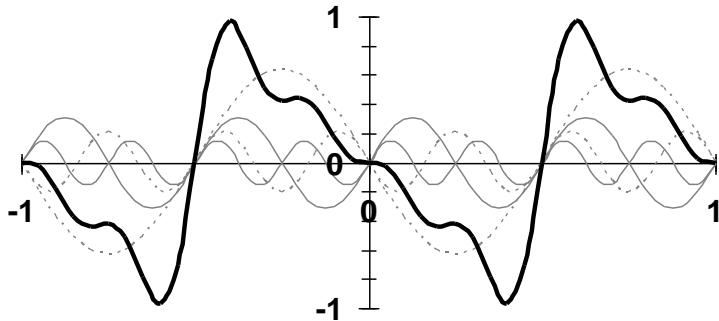
$$b_k = -\frac{2}{k\pi}$$

Therefore, the series is

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$f(t) = -\frac{2}{\pi} \sin(\omega_0 t) + \frac{1}{\pi} \sin(2\omega_0 t) - \frac{2}{3\pi} \sin(3\omega_0 t) + \frac{1}{2\pi} \sin(4\omega_0 t) + \dots$$

The first 4 terms are plotted below along with the summation:



$$19.5 \quad a_0 = 0.5$$

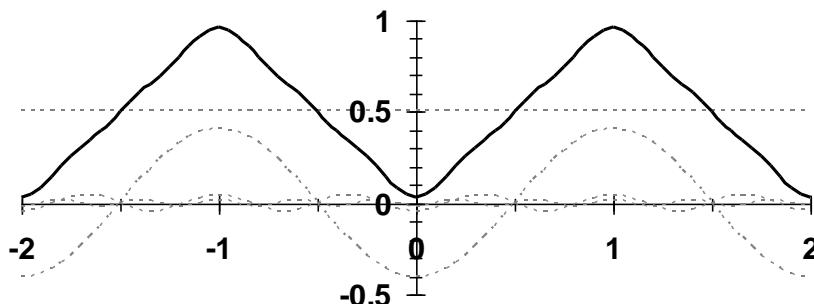
$$\begin{aligned} a_k &= \frac{2}{2} \left[\int_{-1}^0 -t \cos(k\pi t) dt + \int_0^1 t \cos(k\pi t) dt \right] \\ &= 1 \left\{ \left[-\frac{\cos(k\pi t)}{(k\pi)^2} - \frac{t \sin(k\pi t)}{k\pi} \right]_{-1}^0 + \left[\frac{\cos(k\pi t)}{(k\pi)^2} + \frac{t \sin(k\pi t)}{k\pi} \right]_0^1 \right\} \\ &= \frac{2}{(k\pi)^2} (\cos k\pi - 1) \end{aligned}$$

$$b_k = 0$$

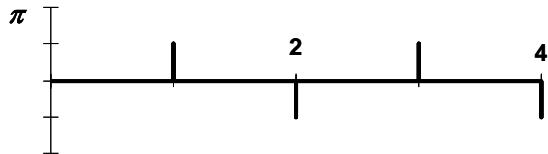
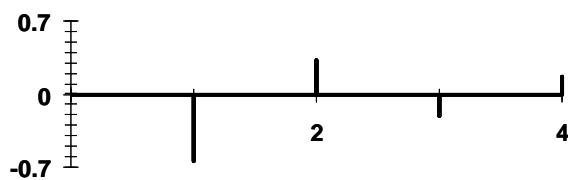
Substituting these coefficients into Eq. (19.17) gives

$$f(t) = \frac{1}{2} - \frac{4}{\pi^2} \cos(\pi t) - \frac{4}{9\pi^2} \cos(3\pi t) - \frac{4}{25\pi^2} \cos(5\pi t) + \dots$$

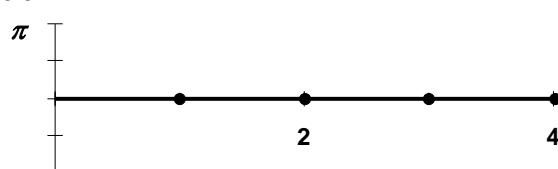
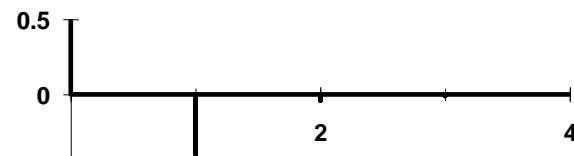
This function for the first 4 terms is displayed below:



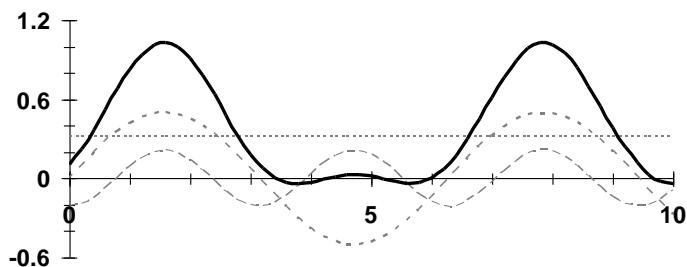
19.6



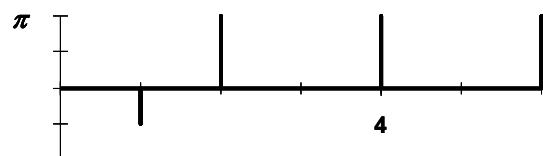
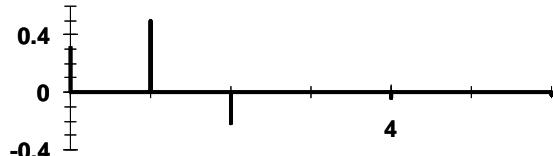
19.7



19.8



19.9



19.10 Here is a VBA code to implement the DFT. It is set up to duplicate Fig. 19.13.

```

Option Explicit

Sub Dfourier()
    Dim i As Integer, N As Integer
    Dim f(127) As Double, re(127) As Double, im(127) As Double
    Dim t As Double, pi As Double, dt As Double, omega As Double
    pi = 4# * Atn(1#)
    N = 16
    t = 0#
    dt = 0.01
    For i = 0 To N - 1
        f(i) = Cos(2 * 12.5 * pi * t)
        t = t + dt
    Next i
    omega = 2 * pi / N
    Call DFT(f, N, re, im, omega)
    Range("A1").Select
    ActiveCell.Value = "INDEX"
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "f(t)"
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "REAL"
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "IMAGINARY"
    Range("A2").Select
    For i = 0 To N - 1
        ActiveCell.Value = i
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = f(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = re(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = im(i)
        ActiveCell.Offset(1, -3).Select
    Next i
    Range("A1").Select
End Sub

Sub DFT(f, N, re, im, omega)
    Dim k As Integer, nn As Integer
    Dim angle As Double
    For k = 0 To N - 1
        re(k) = 0: im(k) = 0
        For nn = 0 To N - 1
            angle = k * omega * nn
            re(k) = re(k) + f(nn) * Cos(angle) / N
            im(k) = im(k) - f(nn) * Sin(angle) / N
        Next nn
    Next k
End Sub

```

When this program is run, the result is

	A	B	C	D
1	INDEX	f(t)	REAL	IMAGINARY
2	0	1.000	0.000	0.000
3	1	0.707	0.000	0.000
4	2	0.000	0.500	0.000
5	3	-0.707	0.000	0.000
6	4	-1.000	0.000	0.000
7	5	-0.707	0.000	0.000
8	6	0.000	0.000	0.000
9	7	0.707	0.000	0.000
10	8	1.000	0.000	0.000
11	9	0.707	0.000	0.000
12	10	0.000	0.000	0.000
13	11	-0.707	0.000	0.000
14	12	-1.000	0.000	0.000
15	13	-0.707	0.000	0.000
16	14	0.000	0.500	0.000
17	15	0.707	0.000	0.000

19.11 The program from Prob. 19.11 can be modified slightly to compute a DFT for the triangular wave from Prob. 19.8. Here is the part that is modified up to the point that the DFT routine is called. The remainder of the program is identical to the one from Prob. 19.11.

Option Explicit

```
Sub Dfourier()
    Dim i As Integer, N As Integer
    Dim f(127) As Double, re(127) As Double, im(127) As Double, t As Double
    Dim pi As Double, Tp As Double, dt As Double, omega As Double
    Dim t1 As Double, t2 As Double, Ni As Integer
    pi = 4# * Atan(1#)
    N = 32
    omega = 2 * pi / N
    t = 0#
    Tp = 2 * pi
    dt = 4 * Tp / N
    For i = 0 To N - 1
        f(i) = Sin(t)
        If f(i) < 0 Then f(i) = 0
        t = t + dt
    Next i
    Call DFT(f, N, re, im, omega)
```

The results for the $n = 32$ case are displayed below:

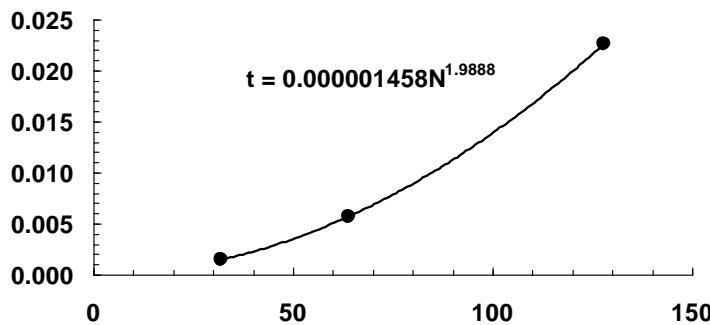
index	f(t)	real	imaginary
0	0	0.3018	0
1	0.7071	0	0
2	1	0	0
3	0.7071	0	0
4	0	0	-0.25
5	0	0	0
6	0	0	0
7	0	0	0
8	0	-0.125	0
9	0.7071	0	0
10	1	0	0
11	0.7071	0	0
12	0	0	0

13	0	0	0
14	0	0	0
15	0	0	0
16	0	-0.0518	0
17	0.7071	0	0
18	1	0	0
19	0.7071	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	-0.125	0
25	0.7071	0	0
26	1	0	0
27	0.7071	0	0
28	0	0	0.25
29	0	0	0
30	0	0	0
31	0	0	0

The runs for $N = 32, 64$ and 128 were performed with the following results obtained. (Note that we had to call the function numerous times to obtain measurable times. These times were then divided by the number of function calls to determine the time per call shown below)

<i>N</i>	<i>time (s)</i>
32	0.001437
64	0.0057
128	0.02264

A power (log-log) model was fit (see plot below) to this data. Thus, the result verifies that the execution time $\propto N^2$.



19.12 Here is a VBA code to implement the FFT. It is set up to duplicate Fig. 19.13.

```

Option Explicit

Sub Ffourier()
Dim i As Integer, N As Integer
Dim f(127) As Double, re(127) As Double, im(127) As Double, t As Double
Dim pi As Double, dt As Double, omega As Double
pi = 4# * Atn(1#)

```

```

N = 16
t = 0#
dt = 0.01
For i = 0 To N - 1
    re(i) = Cos(2 * 12.5 * pi * t)
    im(i) = 0#
    f(i) = re(i)
    t = t + dt
Next i
Call FFT(N, re, im)
Range("A1").Select
ActiveCell.Value = "INDEX"
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "f(t)"
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "REAL"
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "IMAGINARY"
Range("A2:D1026").ClearContents
Range("A2").Select
For i = 0 To N - 1
    ActiveCell.Value = i
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = f(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = re(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = im(i)
    ActiveCell.Offset(1, -3).Select
Next i
Range("A1").Select
End Sub

Sub FFT(N, x, y)
Dim i As Integer, j As Integer, m As Integer
Dim N2 As Integer, N1 As Integer, k As Integer, l As Integer
Dim pi As Double, xN As Double, angle As Double
Dim arg As Double, c As Double, s As Double
Dim xt As Double, yt As Double
xN = N
m = CInt(Log(xN) / Log(2#))
pi = 4# * Atn(1#)
N2 = N
For k = 1 To m
    N1 = N2
    N2 = N2 / 2
    angle = 0#
    arg = 2 * pi / N1
    For j = 0 To N2 - 1
        c = Cos(angle)
        s = -Sin(angle)
        For i = j To N - 1 Step N1
            l = i + N2
            xt = x(i) - x(l)
            x(i) = x(i) + x(l)
            yt = y(i) - y(l)
            y(i) = y(i) + y(l)
            x(l) = xt * c - yt * s
            y(l) = yt * c + xt * s
        Next i
        angle = (j + 1) * arg
    Next j
End Sub

```

```

Next k
j = 0
For i = 0 To N - 2
    If i < j Then
        xt = x(j)
        x(j) = x(i)
        x(i) = xt
        yt = y(j)
        y(j) = y(i)
        y(i) = yt
    End If
    k = N / 2
    Do
        If k >= j + 1 Then Exit Do
        j = j - k
        k = k / 2
    Loop
    j = j + k
Next i
For i = 0 To N - 1
    x(i) = x(i) / N
    y(i) = y(i) / N
Next i
End Sub

```

When this program is run, the result is

	A	B	C	D
1	INDEX	f(t)	REAL	IMAGINARY
2	0	1.0000	0.0000	0.0000
3	1	0.7071	0.0000	0.0000
4	2	0.0000	0.5000	0.0000
5	3	-0.7071	0.0000	0.0000
6	4	-1.0000	0.0000	0.0000
7	5	-0.7071	0.0000	0.0000
8	6	0.0000	0.0000	0.0000
9	7	0.7071	0.0000	0.0000
10	8	1.0000	0.0000	0.0000
11	9	0.7071	0.0000	0.0000
12	10	0.0000	0.0000	0.0000
13	11	-0.7071	0.0000	0.0000
14	12	-1.0000	0.0000	0.0000
15	13	-0.7071	0.0000	0.0000
16	14	0.0000	0.5000	0.0000
17	15	0.7071	0.0000	0.0000

19.13 The program from Prob. 19.12 can be modified slightly to compute a FFT for the triangular wave from Prob. 19.8. Here is the part that is modified up to the point that the FFT routine is called. The remainder of the program is identical to the one from Prob. 19.11.

```

Option Explicit

Sub Ffourier()
    Dim i As Integer, N As Integer
    Dim f(127) As Double, re(127) As Double, im(127) As Double, t As Double
    Dim pi As Double, dt As Double, Tp As Double, omega As Double
    pi = 4# * Atn(1#)
    N = 32
    t = 0#
    Tp = 2 * pi

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

dt = 4 * Tp / N
For i = 0 To N - 1
    re(i) = Sin(t)
    If re(i) < 0 Then re(i) = 0
    im(i) = 0
    f(i) = re(i)
    t = t + dt
Next i
Call FFT(N, re, im)

```

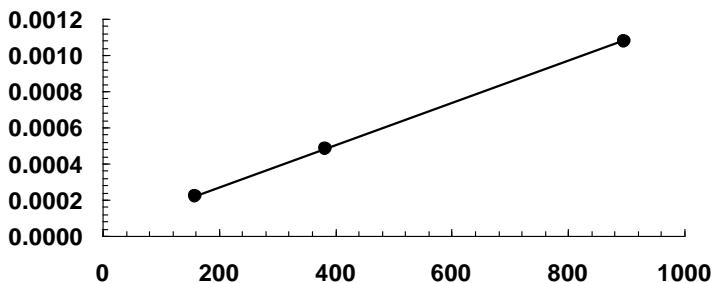
The results for the $n = 32$ case are displayed below:

index	f(t)	real	imaginary
0	0	0.3018	0
1	0.7071	0	0
2	1	0	0
3	0.7071	0	0
4	0	0	-0.25
5	0	0	0
6	0	0	0
7	0	0	0
8	0	-0.125	0
9	0.7071	0	0
10	1	0	0
11	0.7071	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	-0.0518	0
17	0.7071	0	0
18	1	0	0
19	0.7071	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	-0.125	0
25	0.7071	0	0
26	1	0	0
27	0.7071	0	0
28	0	0	0.25
29	0	0	0
30	0	0	0
31	0	0	0

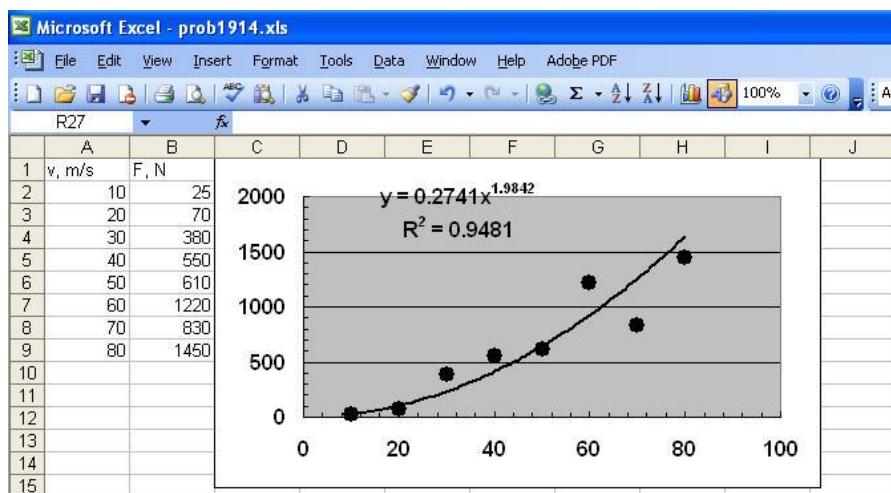
The runs for $N = 32, 64$ and 128 were performed with the following results obtained. (Note that we had to call the function numerous times to obtain measurable times. These times were then divided by the number of function calls to determine the time per call shown below)

N	time (s)
32	0.000219
64	0.000484
128	0.001078

A plot of time versus $N \log_2 N$ yielded a straight line (see plot below). Thus, the result verifies that the execution time $\propto N \log_2 N$.



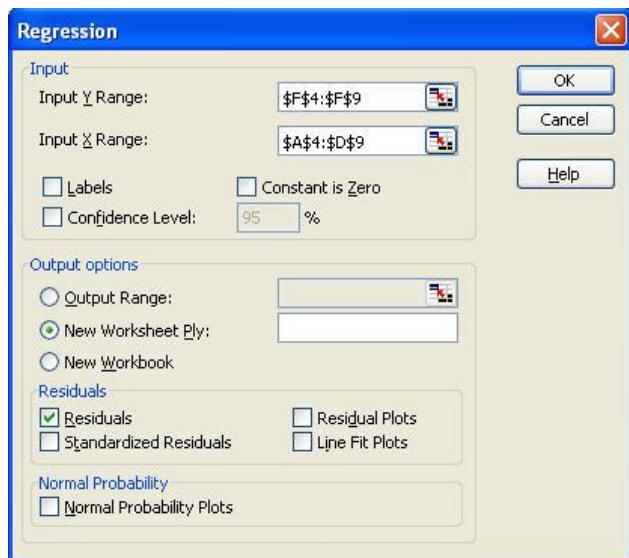
19.14



19.15 An Excel worksheet can be developed with columns holding the dependent variable (o) along with the independent variable (T). In addition, columns can be set up holding progressively higher powers of the independent variable.

	A	B	C	D	E	F
1	Problem 19.15					
2						
3	T	T^2	T^3	T^4	T^5	$o, \text{mg/L}$
4	0	0	0	0	0	14.62
5	8	64	512	4096	32768	11.84
6	16	256	4096	65536	1048576	9.87
7	24	576	13824	331776	7962624	8.42
8	32	1024	32768	1048576	33554432	7.31
9	40	1600	64000	2560000	1.02E+08	6.41

The Data Analysis Toolpack can then be used to develop a regression polynomial as described in Example 19.4. For example, a fourth-order polynomial can be developed as



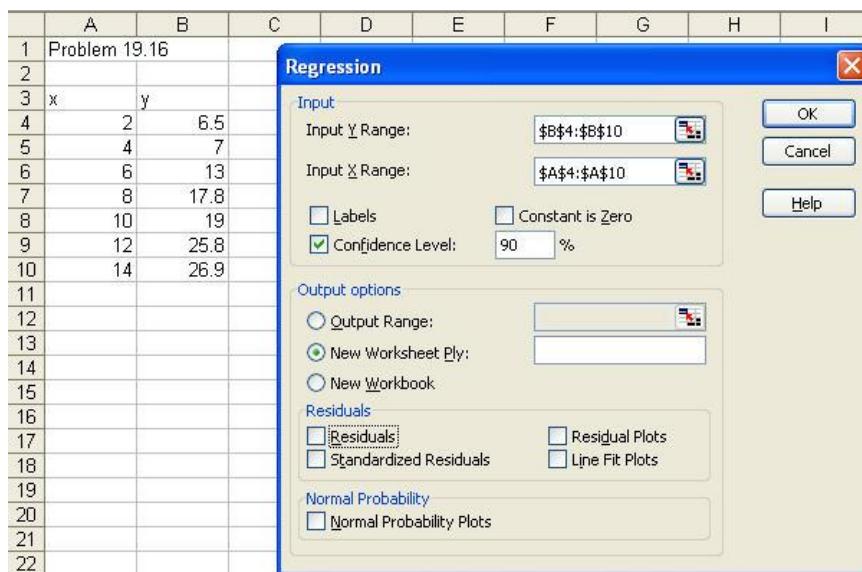
Notice that we have checked off the Residuals box. Therefore, when the regression is implemented, the model predictions (the column labeled Predicted Y) are listed along with the fit statistics as shown below:

	A	B	C	D	E	F	G	H	I
1	SUMMARY OUTPUT								
2									
3	Regression Statistics								
4	Multiple R	0.999999848							
5	R Square	0.999999696							
6	Adjusted R Square	0.99999848							
7	Standard Error	0.003779645							
8	Observations	6							
9									
10	ANOVA								
11		df	SS	MS	F	Significance F			
12	Regression	4	46.97733571	11.74433393	822103.38	0.000827176			
13	Residual	1	1.42857E-05	1.42857E-05					
14	Total	5	46.97735						
15									
16		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%
17	Intercept	14.6197619	0.003772138	3875.722993	0.0001643	14.57183235	14.6676915	14.5718323	14.6676915
18	X Variable 1	-0.411661706	0.001738731	-236.7598567	0.0026889	-0.43375438	-0.38956903	-0.4337544	-0.38956903
19	X Variable 2	0.009023438	0.00020762	43.46130412	0.0146454	0.006385375	0.0116615	0.00638537	0.0116615
20	X Variable 3	-0.000129123	8.18953E-06	-15.76687532	0.040323	-0.00023318	-2.5065E-05	-0.0002332	-2.5065E-05
21	X Variable 4	8.13802E-07	1.01725E-07	8	0.0791668	-4.7874E-07	2.1063E-06	-4.787E-07	2.1063E-06
22									
23									
24									
25	RESIDUAL OUTPUT								
26									
27	Observation	Predicted Y	Residuals						
28	1	14.6197619	0.000238095						
29	2	11.84119048	-0.001190476						
30	3	9.867619048	0.002380952						
31	4	8.422380952	-0.002380952						
32	5	7.308809524	0.001190476						
33	6	6.410238095	-0.000238095						

As can be seen, the results match to the level of precision (second decimal place) of the original data. If a third-order polynomial was used, this would not be the case. Therefore, we conclude that the best-fit equation is

$$o = 14.61976 - 0.4116617T + 9.023438 \times 10^{-3}T^2 - 1.29123 \times 10^{-4}T^3 + 8.13802 \times 10^{-7}T^4$$

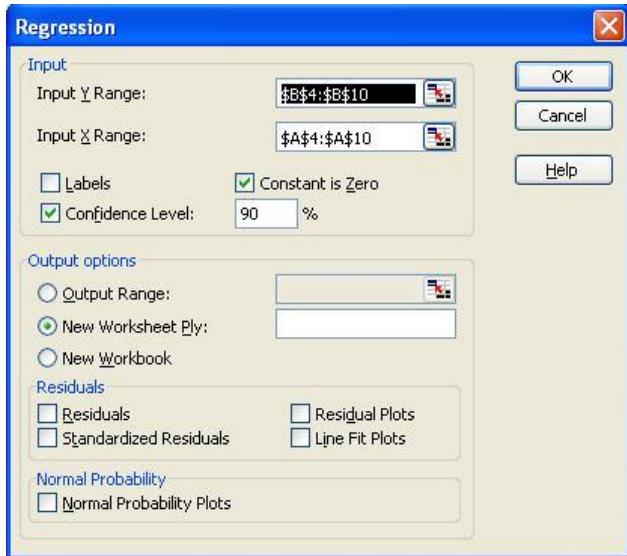
- 19.16** An Excel worksheet can be developed with columns holding the dependent variable (o) along with the independent variable (T). The Data Analysis Toolpack can then be used to develop a linear regression.



Notice that we have checked off the Confidence Level box and entered 90%. Therefore, when the regression is implemented, the 90% confidence intervals for the coefficients will be computed and displayed as shown below:

	A	B	C	D	E	F	G	H	I
1	SUMMARY OUTPUT								
2									
3	Regression Statistics								
4	Multiple R	0.984069227							
5	R Square	0.968392244							
6	Adjusted R Square	0.962070692							
7	Standard Error	1.600178561							
8	Observations	7							
9									
10	ANOVA								
11		df	SS	MS	F	Significance F			
12	Regression	1	392.2514286	392.25143	153.18902	6.09999E-05			
13	Residual	5	12.80285714	2.5605714					
14	Total	6	405.0542857						
15									
16		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 90.0%	Upper 90.0%
17	Intercept		1.6	1.35239772	1.1830839	0.289958668	-1.876449011	5.076449011	-1.125146823
18	X Variable 1	1.871428571	0.151202662	12.376955	6.1E-05	1.482749756	2.260107387	1.566747894	2.176109249

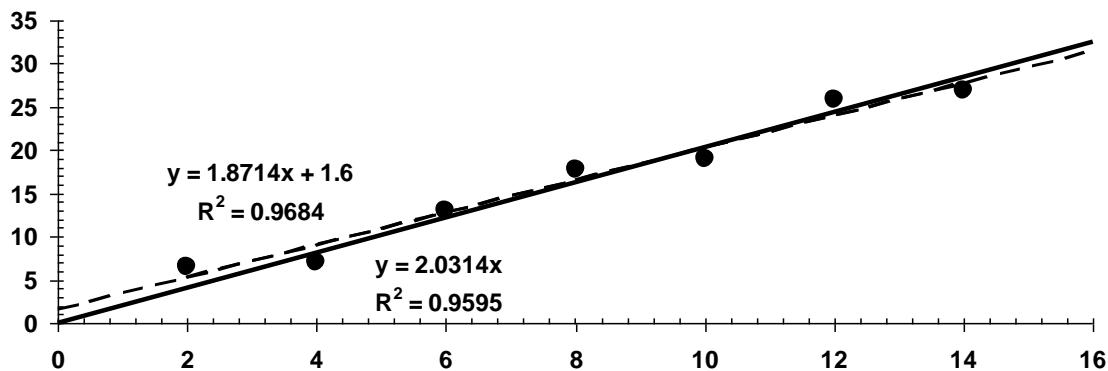
As can be seen, the 90% confidence interval for the intercept (-1.125, 4.325) encompasses zero. Therefore, we can redo the regression, but forcing a zero intercept. As shown below, this is accomplished by checking the Constant is Zero box.



When the regression is implemented, the best-fit equation is

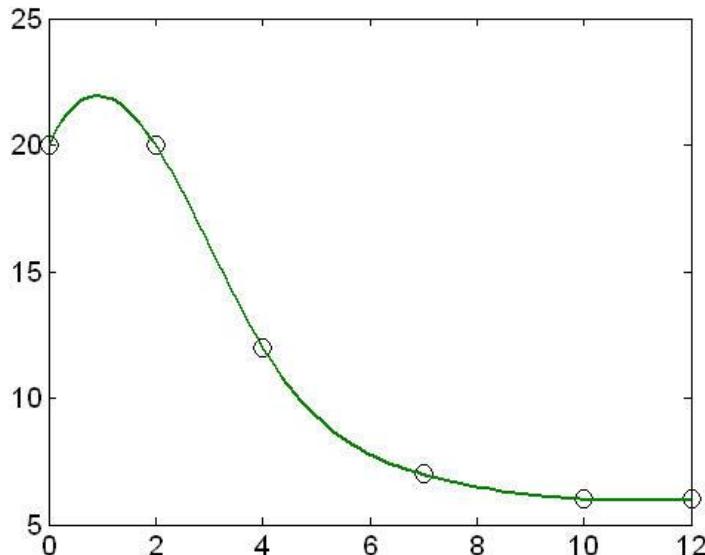
$$y = 2.031429 x$$

This equation, along with the original linear regression, is displayed together with the data on the following plot:



19.17 (a) Here is a MATLAB session to develop the spline and plot it along with the data:

```
>> x=[0 2 4 7 10 12];
>> y=[20 20 12 7 6 6];
>> xx=linspace(0,12);
>> yy=spline(x,y,xx);
>> plot(x,y,'o',xx,yy)
```



Here is the command to make the prediction:

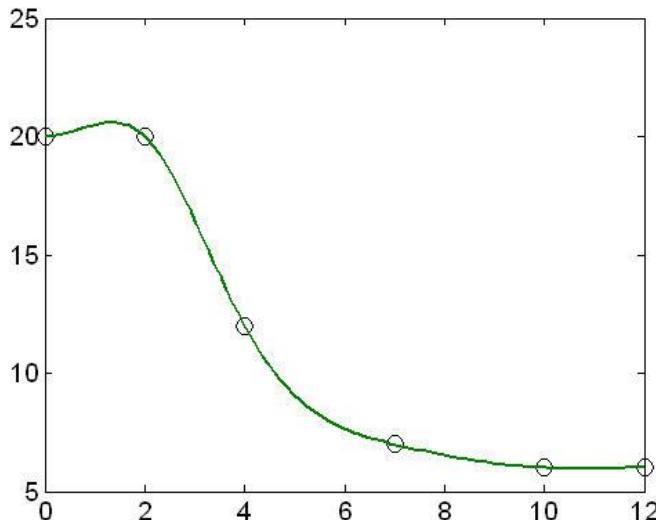
```
>> yp=spline(x,y,1.5)
```

```
yp =
21.3344
```

(b) To prescribe zero first derivatives at the end knots, the y vector is modified so that the first and last elements are set to the desired values of zero. The plot and the prediction both indicate that there is less overshoot between the first two points because of the prescribed zero slopes.

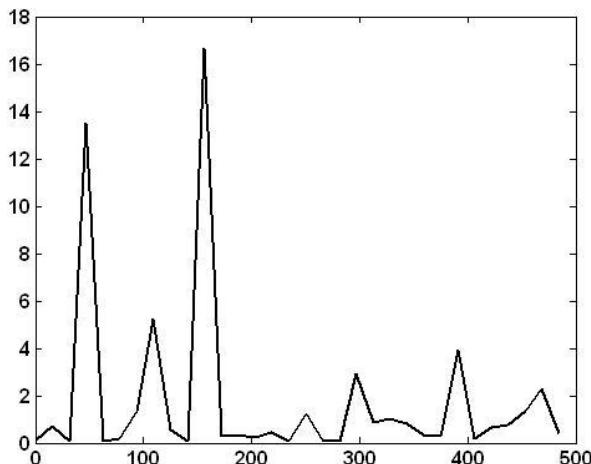
```
>> yd=[0 y 0];
>> yy=spline(x,yd,xx);
>> plot(x,y,'o',xx,yy)
>> yy=spline(x,yd,1.5)
```

```
yy =
20.5701
```



19.18 The following MATLAB session develops the fft along with a plot of the power spectral density versus frequency.

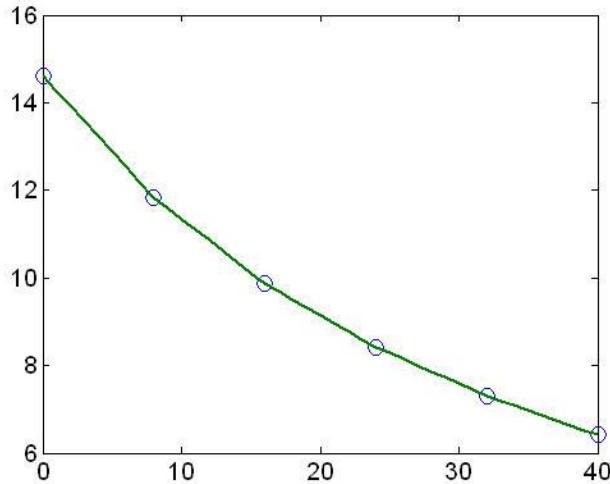
```
>> t=0:63;
>> y=cos(10*2*pi*t/63)+sin(3*2*pi*t/63)+randn(size(t));
>> Y=fft(y,64);
>> Pyy=Y.*conj(Y)/64;
>> f=1000*(0:31)/64;
>> plot(f,Pyy(1:32))
```



19.19 (a) Linear interpolation

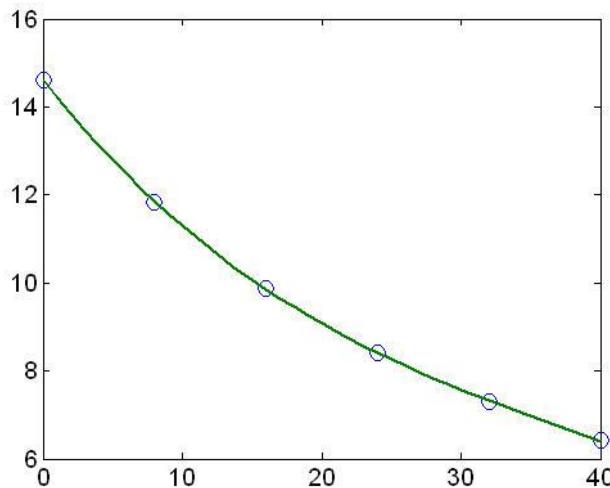
```
>> T=[0 8 16 24 32 40];
>> o=[14.62 11.84 9.87 8.42 7.31 6.41];
>> Ti=0:1:40;
>> oi=interp1(T,o,Ti);
>> plot(T,o,'o',Ti,oi)
```

```
>> ol=interp1(T,o,10)
ol =
    11.3475
```



(b) Third-order regression polynomial

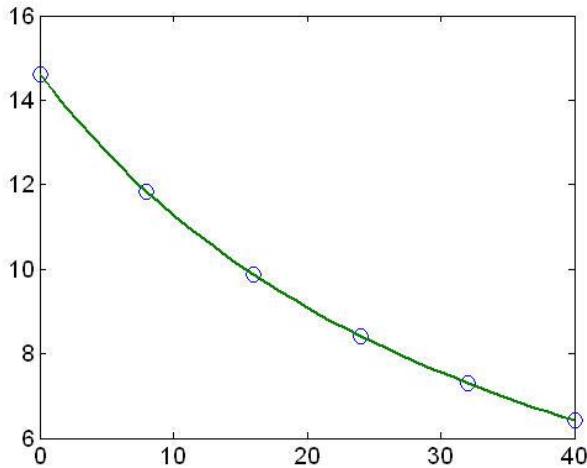
```
>> p=polyfit(T,o,3)
p =
    -0.0001    0.0074   -0.3998    14.6140
>> oi=polyval(p,Ti);
>> plot(T,o,'o',Ti,oi)
>> op=polyval(p,10)
op =
    11.2948
```



(c) Cubic spline

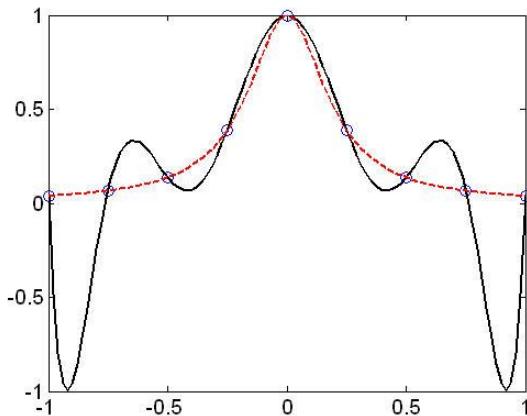
```
>> oi=spline(T,o,Ti);
>> plot(T,o,'o',Ti,oi)
```

```
>> os=spline(T,o,10)
os =
11.2839
```



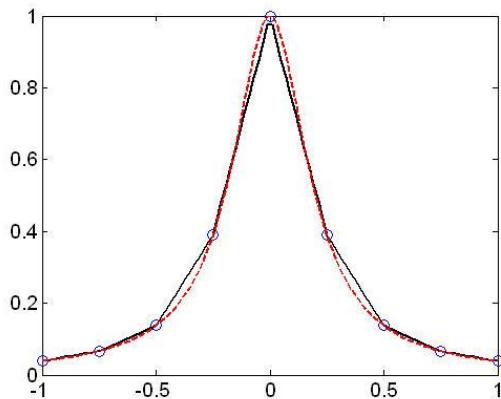
19.20 (a) Eighth-order polynomial:

```
>> x=linspace(-1,1,9);
>> y=1./(1+25*x.^2);
>> p=polyfit(x,y,8);
>> xx=linspace(-1,1);
>> yy=polyval(p,xx);
>> yr=1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```



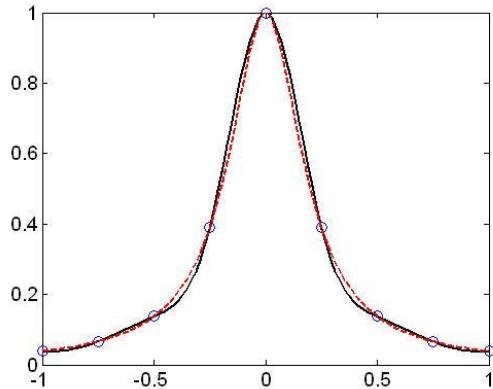
(b) linear spline:

```
>> x=linspace(-1,1,9);
>> y=1./(1+25*x.^2);
>> xx=linspace(-1,1);
>> yy=interp1(x,y,xx);
>> yr=1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```



(b) cubic spline:

```
>> x=linspace(-1,1,9);
>> y=1./(1+25*x.^2);
>> xx=linspace(-1,1);
>> yy=spline(x,y,xx);
>> yr=1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yy,'--')
```

**19.21**

```
PROGRAM Fitpoly
Use IMSL
Implicit NONE
Integer::ndeg,nobs,i,j
Parameter (ndeg=4, nobs=6)
Real:: b (ndeg + 1), sspoly(ndeg + 1), stat(10)
Real:: x(nobs), y(nobs), ycalc(nobs)
Data x/0,8,16,24,32,40/
Data y/14.62,11.84,9.87,8.42,7.31,6.41/
Call Rcurv(nobs, x, y, ndeg, b, sspoly, stat)
Print *, 'Fitted polynomial is'
Do i = 1,ndeg+1
  Print 10, i - 1, b(i)
```

```

End Do
Print *
Print 20, stat(5)
Print *
Print *, '      No.          X          Y          YCALC'
Do i = 1,nobs
  ycalc = 0
  Do j = 1,ndeg+1
    ycalc(i) = ycalc(i) + b(j)*x(i)**(j-1)
  End Do
  Print 30, i, x(i), y(i), ycalc(i)
End Do
10 Format(1X, 'X^',I1,' TERM: ',F12.7)
20 Format(1X,'R^2: ',F8.3,'%')
30 Format(1X,I8,3(5X,F8.4))
End

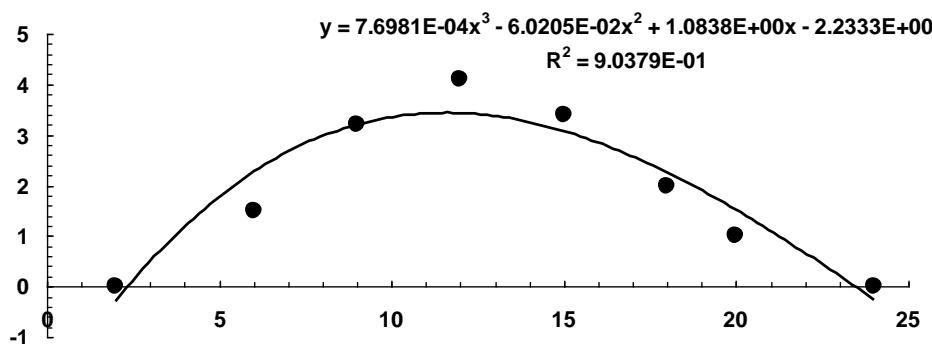
Fitted polynomial is
X^0 TERM: 0.146198E+02
X^1 TERM: -.411661E+00
X^2 TERM: 0.902330E-02
X^3 TERM: -.129118E-03
X^4 TERM: 0.813734E-06

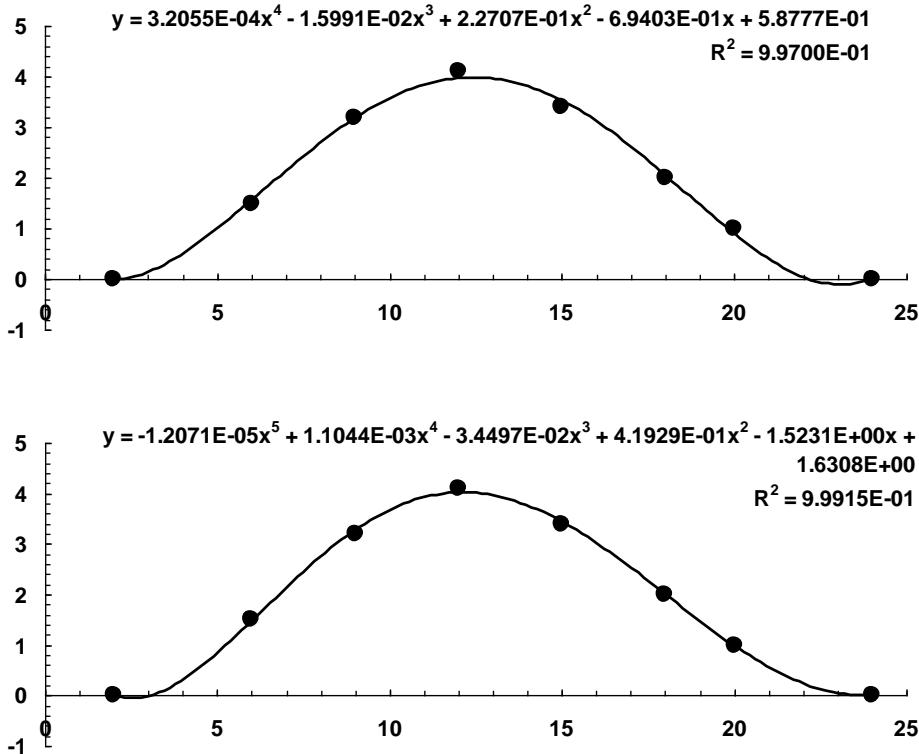
R^2: 100.000%

```

No.	X	Y	YCALC
1	0.0000	14.6200	14.6198
2	8.0000	11.8400	11.8412
3	16.0000	9.8700	9.8676
4	24.0000	8.4200	8.4224
5	32.0000	7.3100	7.3088
6	40.0000	6.4100	6.4102

- 19.22** Using Excel, plot the data and use the trend line function to fit a polynomial of specific order. Obtain the r^2 value to determine the goodness of fit.





Use the 5th order polynomial:

$$C = -1.2071 \times 10^{-5} t^5 + 1.1044 \times 10^{-3} t^4 - 3.4497 \times 10^{-2} t^3 + 0.41929 t^2 - 1.5231 t + 1.6308$$

Integrate to find the area under the curve:

$$\int_2^{24} -1.2071 \times 10^{-5} t^5 + 1.1044 \times 10^{-3} t^4 - 3.4497 \times 10^{-2} t^3 + 0.41929 t^2 - 1.5231 t + 1.6308 dt = 44.37$$

Area under curve: 44.37 mg sec/L

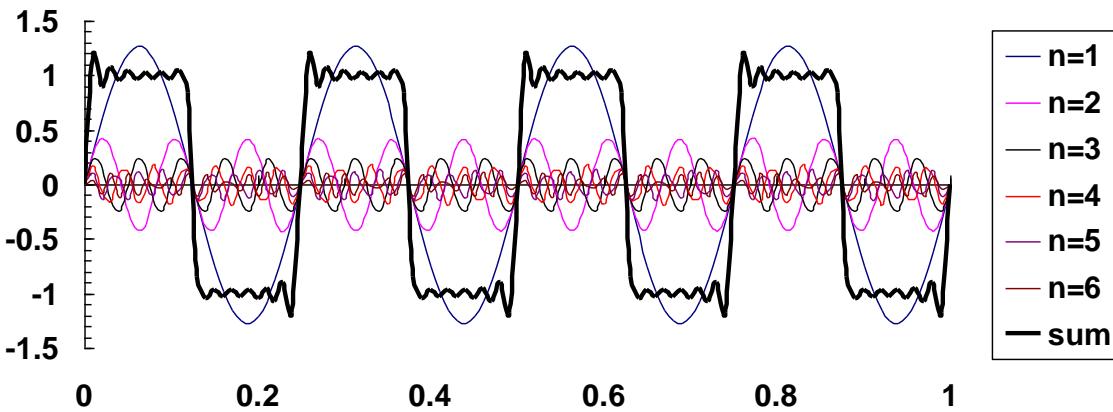
$$\text{Cardiac output} = \frac{5 \text{ mg}}{44.37 \text{ mg sec/L}} = 0.11269 \text{ L/sec} = 6.76 \text{ L/min}$$

19.23 Plug in $A_0 = 1$ and $T = 0.25$,

$$f(t) = \sum_{n=1}^{\infty} \left(\frac{4}{(2n-1)\pi} \right) \sin(8\pi(2n-1)t)$$

Make a table and plot in Excel. The following shows the first several entries from the table along with the plot.

	A	B	C	D	E	F	G	H
6	t	n=1	n=2	n=3	n=4	n=5	n=6	sum
7	0	0	0	0	0	0	0	0
8	0.01	0.316642	0.290531	0.242185	0.17867	0.109005	0.04261	1.179642
9	0.02	0.613388	0.423576	0.149678	-0.06696	-0.13897	-0.07924	0.901482
10	0.03	0.871592	0.327016	-0.14968	-0.15358	0.068154	0.104733	1.068241
11	0.04	1.075032	0.053193	-0.24218	0.124513	0.052079	-0.11552	0.947112
12	0.05	1.210923	-0.24946	-6.2E-17	0.106913	-0.13455	0.110084	1.043909
13	0.06	1.270727	-0.4169	0.242185	-0.16458	0.119448	-0.08919	0.961698
14	0.07	1.250687	-0.35834	0.149678	-0.04523	-0.01773	0.055763	1.034818
15	0.08	1.152062	-0.10555	-0.14968	0.181532	-0.09684	-0.01451	0.967018
16	0.09	0.981048	0.204463	-0.24218	-0.0228	0.141192	-0.02879	1.032935
17	0.1	0.748391	0.403641	-1.2E-16	-0.17299	-0.08315	0.068036	0.963924



19.24 This problem is convenient to solve with MATLAB

- (a) When we first try to fit a sixth-order interpolating polynomial, MATLAB displays the following error message

```
>> x=[0 100 200 400 600 800 1000];
>> y=[0 0.82436 1 .73576 .40601 .19915 .09158];
>> p=polyfit(x,y,6);
Warning: Polynomial is badly conditioned. Remove repeated data points
or try centering and scaling as described in HELP POLYFIT.
> In polyfit at 79
```

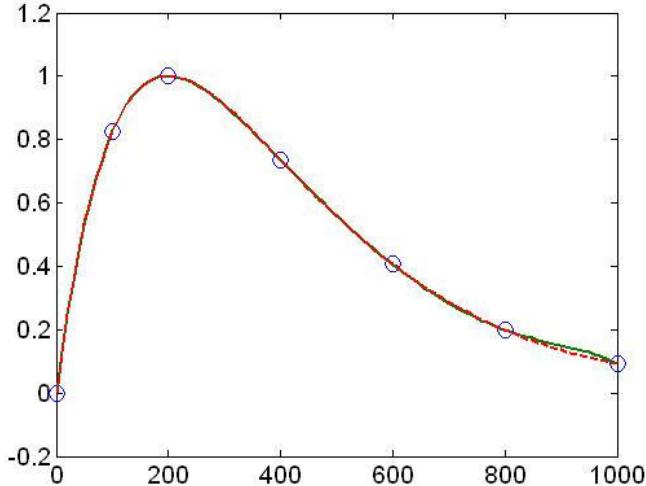
Therefore, we redo the calculation but centering and scaling the x values as shown,

```
>> xs=(x-500)/100;
>> p=polyfit(xs,y,6);
```

Now, there is no error message so we can proceed.

```
>> xx=linspace(0,1000);
>> yy=polyval(p, (xx-500)/100);
>> yc=xx/200.*exp(-xx/200+1);
>> plot(x,y, 'o', xx, yy, xx, yc, '--')
```

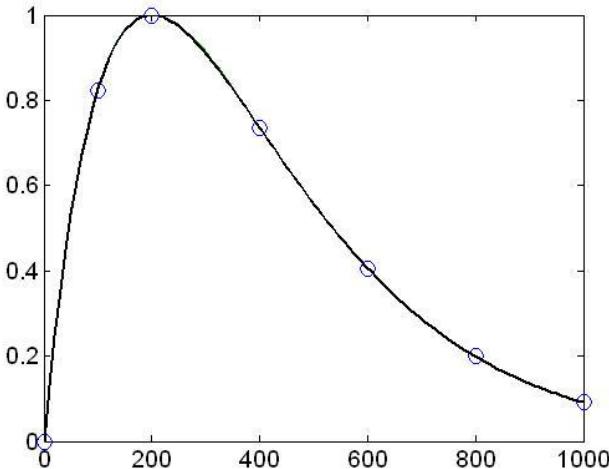
This results in a decent plot. Note that the interpolating polynomial (solid) and the original function (dashed) almost plot on top of each other:



(b) Cubic spline:

```
>> x=[0 100 200 400 600 800 1000];
>> y=[0 0.82436 1 .73576 .40601 .19915 .09158];
>> xx=linspace(0,1000);
>> yc=xx/200.*exp(-xx/200+1);
>> yy=spline(x,y,xx);
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```

For this case, the fit is so good that the spline and the original function are indistinguishable.

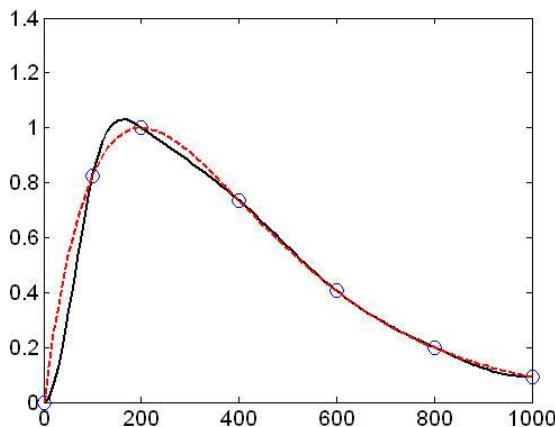


(c) Cubic spline with clamped end conditions (zero slope):

```
>> x=[0 100 200 400 600 800 1000];
>> y=[0 0.82436 1 .73576 .40601 .19915 .09158];
>> ys=[0 y 0];
```

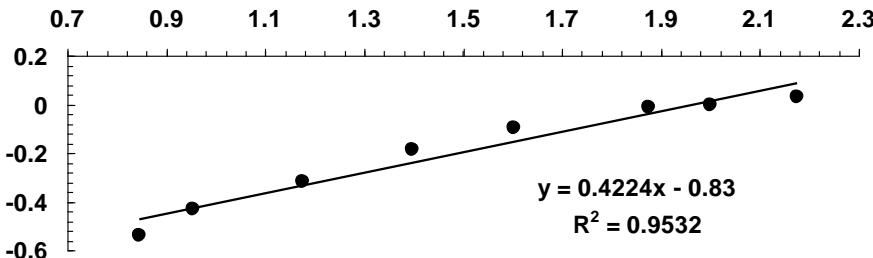
```
>> xx=linspace(0,1000);
>> yc=xx/200.*exp(-xx/200+1);
>> yy=spline(x,ys,xx);
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```

For this case, the spline differs from the original function because the latter does not have zero end derivatives.



CHAPTER 20

20.1 A plot of $\log_{10}k$ versus $\log_{10}f$ can be developed as



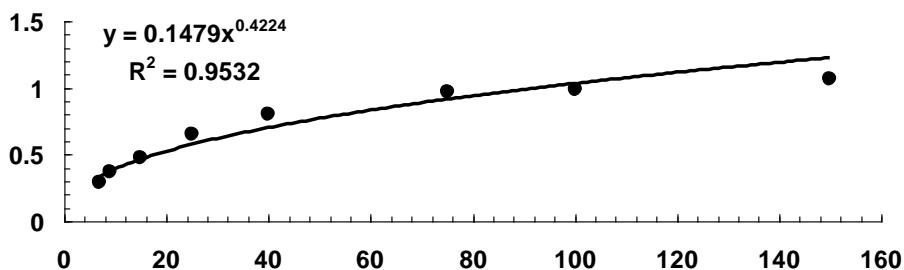
As shown, the best fit line is

$$\log_{10} f = 0.422363 \log_{10} k - 0.83$$

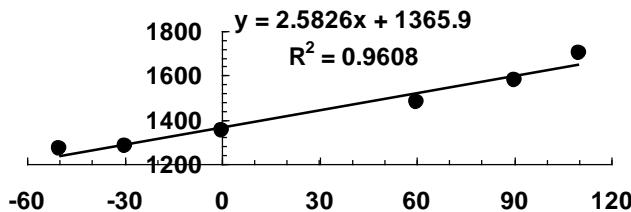
Therefore, $\alpha_2 = 10^{-0.83} = 0.147913$ and $\beta_2 = 0.422363$, and the power model is

$$y = 0.147913 x^{0.422363}$$

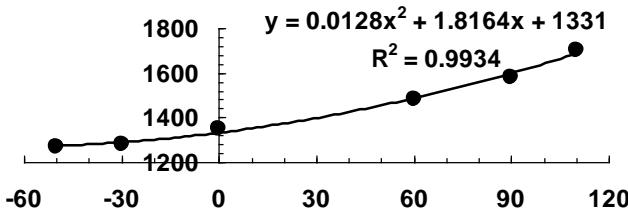
The model and the data can be plotted as



20.2 We can first try a linear fit



As shown, the fit line is somewhat lacking. Therefore, we can use polynomial regression to fit a parabola



This fit seems adequate in that it captures the general trend of the data. Note that a slightly better fit can be attained with a cubic polynomial, but the improvement is marginal.

20.3 This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$$\begin{array}{ll} x_0 = 20 & f(x_0) = 8.17 \\ x_1 = 15 & f(x_1) = 9.03 \\ x_2 = 25 & f(x_2) = 7.46 \\ x_3 = 10 & f(x_3) = 10.1 \\ x_4 = 30 & f(x_4) = 6.85 \\ x_5 = 5 & f(x_5) = 11.3 \\ x_6 = 0 & f(x_6) = 12.9 \end{array}$$

The results of applying Newton's polynomial at $T = 18$ are

Order	f(x)	Error
0	8.17000	0.344
1	8.51400	-0.018
2	8.49600	-0.00336
3	8.49264	0.00022
4	8.49286	-0.00161
5	8.49125	-0.00298
6	8.48827	

The minimum error occurs for the third-order version so we conclude that the interpolation is 8.4926.

20.4 We can use MATLAB to solve this problem,

```
>> format long
>> T = [0 5 10 15 20 25 30];
>> c = [12.9 11.3 10.1 9.03 8.17 7.46 6.85];
>> p = polyfit(T,c,3)

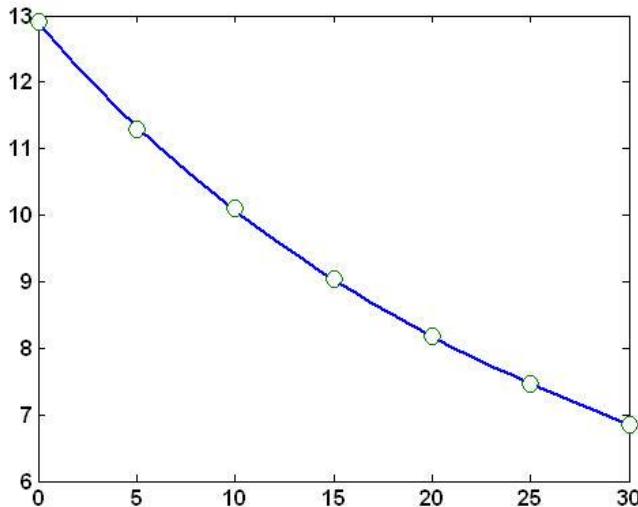
p =
-0.0000622222222222 0.00652380952381 -0.3411111111111111
12.88785714285715
```

Thus, the best-fit cubic would be

$$c = 12.8878571 - 0.34111111T + 0.00652381T^2 - 0.000062222T^3$$

We can generate a plot of the data along with the best-fit line as

```
>> Tp=[0:30];
>> cp=polyval(p,Tp);
>> plot(Tp,cp,T,c,'o')
```



We can use the best-fit equation to generate a prediction at $T = 8$ as

```
>> y=polyval(p,8)
y =
10.54463428571429
```

20.5 The multiple linear regression model to evaluate is

$$o = a_0 + a_1 T + a_2 c$$

As described in Section 17.1, we can use general linear least squares to generate the best-fit equation. The $[Z]$ and y matrices can be set up using MATLAB commands as

```
>> format long
>> t = [0 5 10 15 20 25 30];
>> T = [t t t]';
>> c = [zeros(size(t)) 10*ones(size(t)) 20*ones(size(t))]';
>> Z = [ones(size(T)) T c];
>> y = [14.6 12.8 11.3 10.1 9.09 8.26 7.56 12.9 11.3 10.1 9.03 8.17
7.46 6.85 11.4 10.3 8.96 8.08 7.35 6.73 6.2]';
```

The coefficients can be evaluated as

```
>> a=inv(Z'*Z)*(Z'*y)
```

```
a =
13.52214285714285
-0.20123809523809
-0.10492857142857
```

Thus, the best-fit multiple regression model is

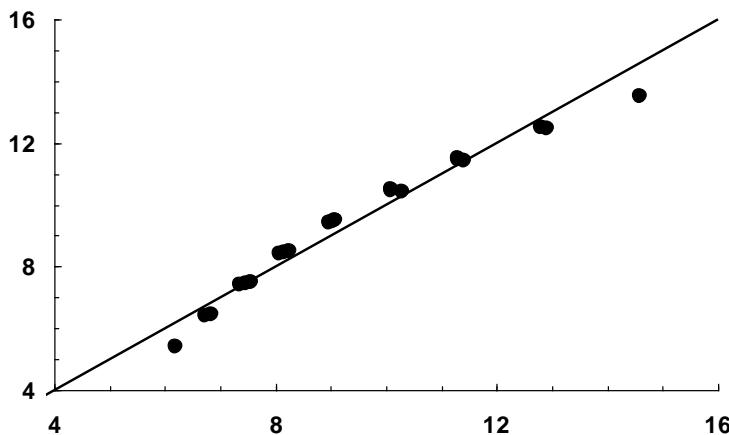
$$o = 13.52214285714285 - 0.20123809523809T - 0.10492857142857c$$

We can evaluate the prediction at $T = 17$ and $c = 5$ and evaluate the percent relative error as

```
>> op = a(1)+a(2)*17+a(3)*5
```

```
op =
9.57645238095238
```

We can also assess the fit by plotting the model predictions versus the data. A one-to-one line is included to show how the predictions diverge from a perfect fit.

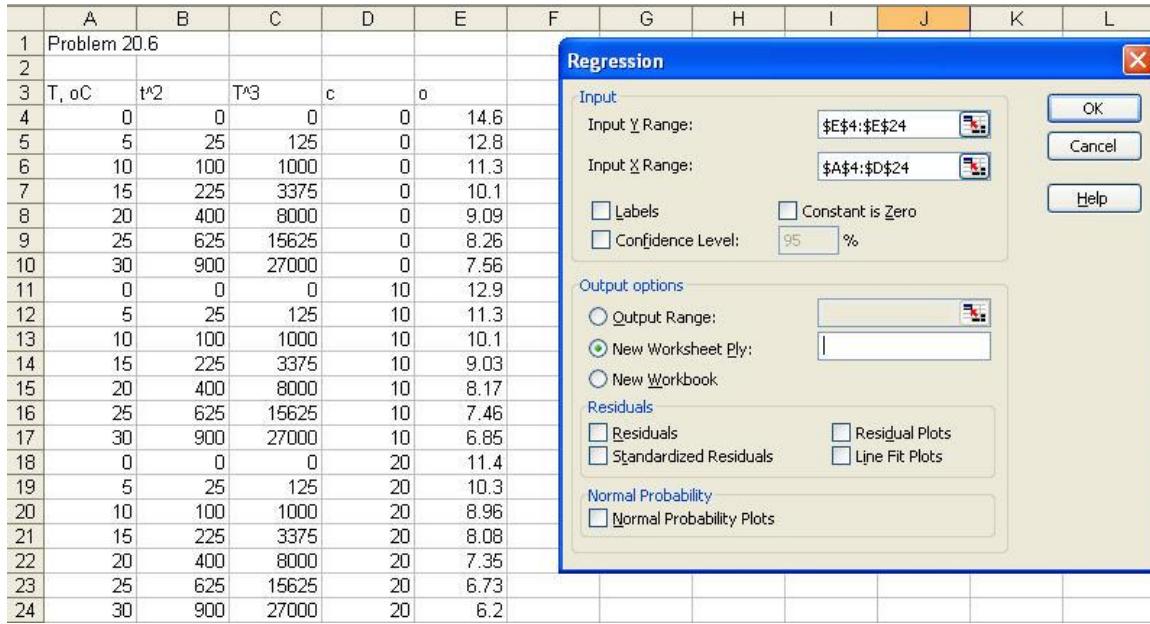


The cause for the discrepancy is because the dependence of oxygen concentration on the unknowns is significantly nonlinear. It should be noted that this is particularly the case for the dependency on temperature.

20.6 The multiple linear regression model to evaluate is

$$o = a_0 + a_1T + a_2T^2 + a_3T^3 + a_4c$$

The Excel Data Analysis toolpack provides a convenient tool to fit this model. We can set up a worksheet and implement the Data Analysis Regression tool as



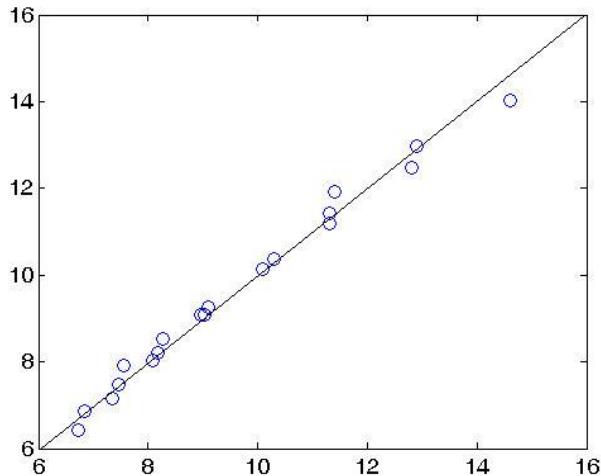
When the tool is run, the following worksheet is generated

	A	B	C	D	E	F	G	H	I
1	SUMMARY OUTPUT								
2									
3	Regression Statistics								
4	Multiple R	0.993835141							
5	R Square	0.987708288							
6	Adjusted R Square	0.98463536							
7	Standard Error	0.282662722							
8	Observations	21							
9									
10	ANOVA								
11		df	SS	MS	F	Significance F			
12	Regression	4	102.7243429	25.68109	321.4225	4.63842E-15			
13	Residual	16	1.278371429	0.079898					
14	Total	20	104.0027143						
15									
16		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%
17	Intercept	14.02714286	0.17446322	80.40172	2.71E-22	13.65729736	14.39699	13.6573	14.39699
18	X Variable 1	-0.33642328	0.049672965	-6.77276	4.48E-06	-0.441725261	-0.23112	-0.44173	-0.23112
19	X Variable 2	0.005744444	0.00406041	1.414745	0.17631	-0.002863241	0.014352	-0.00286	0.014352
20	X Variable 3	-4.37037E-05	8.88323E-05	-0.49198	0.629415	-0.00023202	0.000145	-0.00023	0.000145
21	X Variable 4	-0.104928571	0.007554479	-13.8896	2.41E-10	-0.120943351	-0.08891	-0.12094	-0.08891

Thus, the best-fit model is

$$o_s = 14.027143 - 0.336423T + 0.00574444T^2 - 0.000043704T^3 - 0.10492857c$$

The model can then be used to predict values of oxygen at the same values as the data. These predictions can be plotted against the data to depict the goodness of fit.

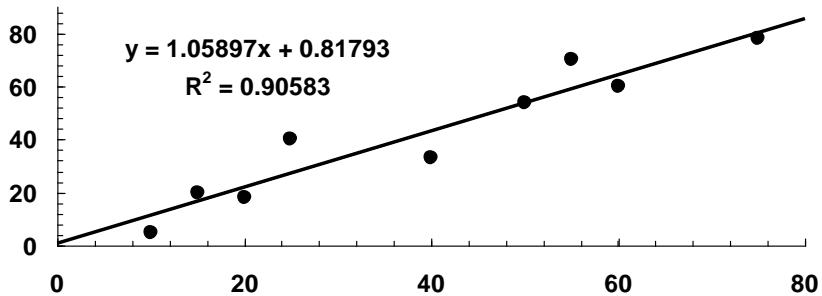


Thus, although there are some discrepancies, the fit is generally adequate. Finally, the prediction can be made at $T = 20$ and $c = 10$,

$$o_s = 14.027143 - 0.336423(20) + 0.00574444(20)^2 - 0.000043704(20)^3 - 0.10492857(10) = 8.19754$$

which compares favorably with the true value of 8.17 mg/L.

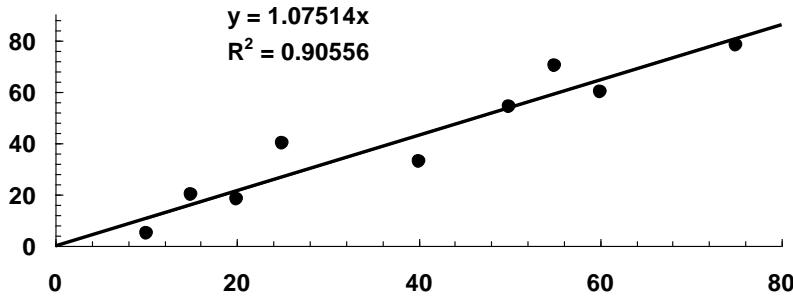
20.7 (a) The linear fit is



The tensile strength at $t = 32$ can be computed as

$$y = 1.05897(32) + 0.81793 = 34.7048913$$

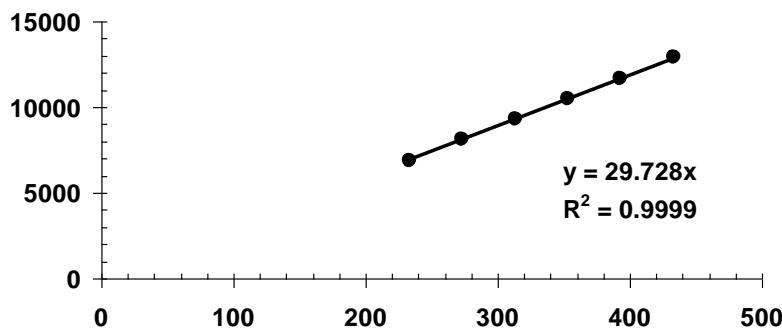
(b) A straight line with zero intercept can be fit as



For this case, the tensile strength at $t = 32$ can be computed as

$$y = 1.07514(32) = 34.40452$$

20.8 Linear regression with a zero intercept gives [note that $T(K) = T(^{\circ}\text{C}) + 273.15$].



Thus, the fit is

$$p = 29.728T$$

Using the ideal gas law

$$R = \left(\frac{p}{T}\right)\frac{V}{n}$$

For our fit

$$\frac{p}{T} = 29.728$$

For nitrogen,

$$n = \frac{1 \text{ kg}}{28 \text{ g/mole}}$$

Therefore,

$$R = 29.728 \left(\frac{10}{10^3 / 28} \right) = 8.324$$

This is close to the standard value of 8.314 J/gmole.

20.9 This problem is ideally suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$$\begin{array}{ll} x_0 = 740 & f(x_0) = 0.1406 \\ x_1 = 760 & f(x_1) = 0.15509 \\ x_2 = 720 & f(x_2) = 0.12184 \\ x_3 = 780 & f(x_3) = 0.16643 \\ x_4 = 700 & f(x_4) = 0.0977 \end{array}$$

The results of applying Newton's polynomial at $T = 750$ are

Order	f(x)	Error
0	0.14060	0.007245
1	0.14785	0.000534
2	0.14838	-7E-05
3	0.14831	0.00000
4	0.14831	

Note that the third-order polynomial yields an exact result, and so we conclude that the interpolation is 0.14831.

20.10 A program can be written to fit a natural cubic spline to this data and also generate the first and second derivatives at each knot.

Option Explicit

```
Sub Splines()
Dim i As Integer, n As Integer
Dim x(100) As Double, y(100) As Double, xu As Double, yu As Double
Dim dy As Double, d2y As Double
Dim resp As Variant
Range("a5").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
Range("a5").Select
For i = 0 To n
    x(i) = ActiveCell.Value
    ActiveCell.Offset(0, 1).Select
    y(i) = ActiveCell.Value
    ActiveCell.Offset(1, -1).Select
Next i
Range("c5").Select
Range("c5:d1005").ClearContents
For i = 0 To n
    Call Spline(x(), y(), n, x(i), yu, dy, d2y)
    ActiveCell.Value = dy
End Sub
```

```

ActiveCell.Offset(0, 1).Select
ActiveCell.Value = d2y
ActiveCell.Offset(1, -1).Select
Next i
Do
    resp = MsgBox("Do you want to interpolate?", vbYesNo)
    If resp = vbNo Then Exit Do
    xu = InputBox("z = ")
    Call Spline(x(), y(), n, xu, yu, dy, d2y)
    MsgBox "For z = " & xu & Chr(13) & "T = " & yu & Chr(13) &
           "dT/dz = " & dy & Chr(13) & "d2T/dz2 = " & d2y
Loop
End Sub

Sub Spline(x, y, n, xu, yu, dy, d2y)
Dim e(100) As Double, f(100) As Double, g(100) As Double, r(100) As Double,
d2x(100) As Double
Call Tridiag(x, y, n, e, f, g, r)
Call Decomp(e(), f(), g(), n - 1)
Call Substit(e(), f(), g(), r(), n - 1, d2x())
Call Interpol(x, y, n, d2x(), xu, yu, dy, d2y)
End Sub

Sub Tridiag(x, y, n, e, f, g, r)
Dim i As Integer
f(1) = 2 * (x(2) - x(0))
g(1) = x(2) - x(1)
r(1) = 6 / (x(2) - x(1)) * (y(2) - y(1))
r(1) = r(1) + 6 / (x(1) - x(0)) * (y(0) - y(1))
For i = 2 To n - 2
    e(i) = x(i) - x(i - 1)
    f(i) = 2 * (x(i + 1) - x(i - 1))
    g(i) = x(i + 1) - x(i)
    r(i) = 6 / (x(i + 1) - x(i)) * (y(i + 1) - y(i))
    r(i) = r(i) + 6 / (x(i) - x(i - 1)) * (y(i - 1) - y(i))
Next i
e(n - 1) = x(n - 1) - x(n - 2)
f(n - 1) = 2 * (x(n) - x(n - 2))
r(n - 1) = 6 / (x(n) - x(n - 1)) * (y(n) - y(n - 1))
r(n - 1) = r(n - 1) + 6 / (x(n - 1) - x(n - 2)) * (y(n - 2) - y(n - 1))
End Sub

Sub Interpol(x, y, n, d2x, xu, yu, dy, d2y)
Dim i As Integer, flag As Integer
Dim c1 As Double, c2 As Double, c3 As Double, c4 As Double
Dim t1 As Double, t2 As Double, t3 As Double, t4 As Double
flag = 0
i = 1
Do
    If xu >= x(i - 1) And xu <= x(i) Then
        c1 = d2x(i - 1) / 6 / (x(i) - x(i - 1))
        c2 = d2x(i) / 6 / (x(i) - x(i - 1))
        c3 = y(i - 1) / (x(i) - x(i - 1)) - d2x(i - 1) * (x(i) - x(i - 1)) / 6
        c4 = y(i) / (x(i) - x(i - 1)) - d2x(i) * (x(i) - x(i - 1)) / 6
        t1 = c1 * (x(i) - xu) ^ 3
        t2 = c2 * (x(i) - xu) ^ 3
        t3 = c3 * (x(i) - xu)
        t4 = c4 * (x(i) - xu)
        yu = t1 + t2 + t3 + t4
        t1 = -3 * c1 * (x(i) - xu) ^ 2
        t2 = 3 * c2 * (x(i) - xu) ^ 2
        t3 = -c3
    End If
Loop
End Sub

```

```

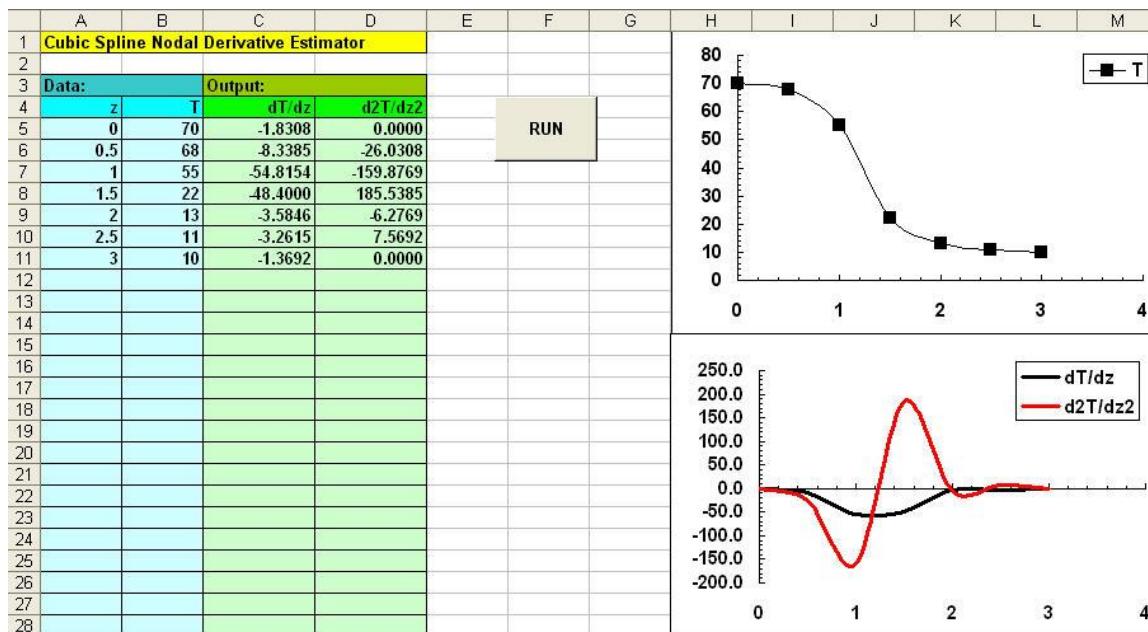
t4 = c4
dy = t1 + t2 + t3 + t4
t1 = 6 * c1 * (x(i) - xu)
t2 = 6 * c2 * (xu - x(i - 1))
d2y = t1 + t2
flag = 1
Else
    i = i + 1
End If
If i = n + 1 Or flag = 1 Then Exit Do
Loop
If flag = 0 Then
    MsgBox "outside range"
End
End If
End Sub

Sub Decomp(e, f, g, n)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
End Sub

Sub Substit(e, f, g, r, n, x)
Dim k As Integer
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

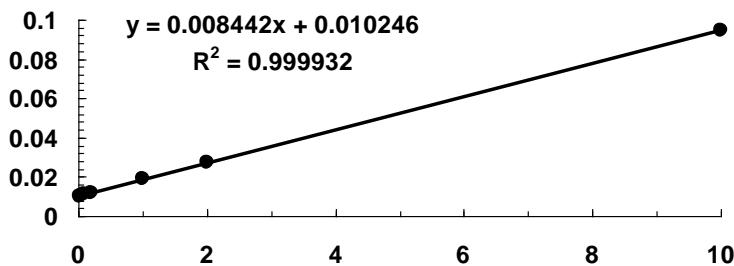
Here is the output when it is applied to the data for this problem:



The plot suggests a zero second derivative at a little above $z = 1.2$ m. The program is set up to allow you to evaluate the spline prediction and the associated derivatives for as many cases as you desire. This can be done by trial-and-error to determine that the zero second derivative occurs at about $z = 1.2315$ m. At this point, the first derivative is $-73.315 \text{ } ^\circ\text{C/m}$. This can be substituted into Fourier's law to compute the flux as

$$J = -0.02 \frac{\text{cal}}{\text{s cm}^\circ\text{C}} \left(-73.315 \frac{\text{}}{\text{m}} \right) \times \frac{\text{m}}{100 \text{ cm}} = 0.01466 \frac{\text{cal}}{\text{cm}^2 \text{ s}}$$

20.11 This is an example of the saturated-growth-rate model. We can plot $1/[F]$ versus $1/[B]$ and fit a straight line as shown below.



The model coefficients can then be computed as

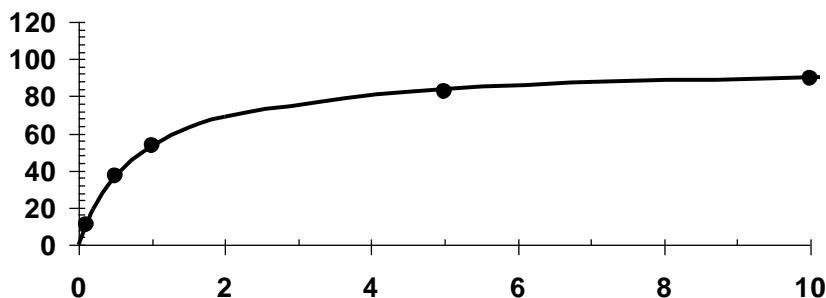
$$\alpha_3 = \frac{1}{0.010246} = 97.60128$$

$$\beta_3 = 97.60128(0.008442) = 0.823968$$

Therefore, the best-fit model is

$$[B] = 97.60128 \frac{[F]}{0.823968 + [F]}$$

A plot of the original data along with the best-fit curve can be developed as



20.12 Nonlinear regression can be used to estimate the model parameters. This can be done using the Excel Solver

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Prob 20.12												
2													
3	R	0.00198											
4													
5	k ₀₁	7526.842											
6	E ₁	4.478896		SSR	1547.576								
7													
8	-dA/dt data	A	T	-dA/dt calc	SR								
9	460	200	280	=466.7210343	=45.1723								
10	960	150	320	=960.9301461	=0.865172								
11	2485	50	450	=2468.669749	=266.6771								
12	1600	20	500	=1632.431512	=1051.803								
13	1245	10	550	=1231.470094	=183.0584								
14													
15													

Solver Parameters

Set Target Cell: \$E\$6

Equal To: Max Min Value of: 0

By Changing Cells: \$B\$5:\$B\$6

Subject to the Constraints:

Solve **Close** **Guess** **Options** **Add** **Change** **Delete** **Reset All** **Help**

Here are the formulas:

	A	B	C	D	E
1	Prob 20.12				
2					
3	R	0.00198			
4					
5	k ₀₁	7526.84240252			
6	E ₁	4.47889589058	SSR	=SUM(E9:E16)	
7					
8	-dA/dt data	A	T	-dA/dt calc	SR
9	460	200	280	=k01 *EXP(-E1_/(R_*C9))*B9	=(A9-D9)^2
10	960	150	320	=k01 *EXP(-E1_/(R_*C10))*B10	=(A10-D10)^2
11	2485	50	450	=k01 *EXP(-E1_/(R_*C11))*B11	=(A11-D11)^2
12	1600	20	500	=k01 *EXP(-E1_/(R_*C12))*B12	=(A12-D12)^2
13	1245	10	550	=k01 *EXP(-E1_/(R_*C13))*B13	=(A13-D13)^2

Therefore, we estimate $k_{01} = 7,526.842$ and $E_1 = 4.478896$.

20.13 Nonlinear regression can be used to estimate the model parameters. This can be done using the Excel Solver. To do this, we set up columns holding each side of the equation: $PV/(RT)$ and $1 + A_1/V + A_2/V^2$. We then form the square of the difference between the two sides and minimize this difference by adjusting the parameters.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Prob. 20.13													
2														
3	R	82.05												
4														
5	A ₁	-237.531												
6	A ₂	1.192283		SSR	2.09474E-08									
7														
8	P (atm)	T(K)	Vol(mL)	PV/(RT)	1 + A ₁ /V + A ₂ /V ²	SR								
9	0.985	303	25000	0.990501	0.990498757	6.18312E-12								
10	1.108	303	22200	0.989399	0.989300403	9.74565E-09								
11	1.363	303	18000	0.986841	0.986803831	1.37521E-09								
12	1.631	303	15000	0.984065	0.984164598	9.82039E-09								
13														
14														
15														

Solver Parameters

Set Target Cell: \$F\$6

Equal To: Max Min Value of: 0

By Changing Cells: \$B\$5:\$B\$6

Subject to the Constraints:

Solve **Close** **Guess** **Options** **Add** **Change** **Delete** **Reset All** **Help**

Here are the formulas that underly the worksheet cells:

	A	B	C	D	E	F
1	Prob. 20.13					
2						
3	R	82.05				
4						
5	A1	-237.53111469				
6	A2	1.19228253686		SSR	=SUM(F9:F12)	
7						
8	P (atm)	T(K)	Vol(mL)	PV/(RT)	1 + A1/V + A2/V^2	SR
9	0.985	303	25000	=A9*C9/(R_*B9)	=1+A1/_C9+A2/_C9^2	=(D9-E9)^2
10	1.108	303	22200	=A10*C10/(R_*B10)	=1+A1/_C10+A2/_C10^2	=(D10-E10)^2
11	1.363	303	18000	=A11*C11/(R_*B11)	=1+A1/_C11+A2/_C11^2	=(D11-E11)^2
12	1.631	303	15000	=A12*C12/(R_*B12)	=1+A1/_C12+A2/_C12^2	=(D12-E12)^2

Therefore, we estimate $A_1 = -237.531$ and $A_2 = 1.192283$.

20.14 The standard errors can be computed via Eq. 17.9

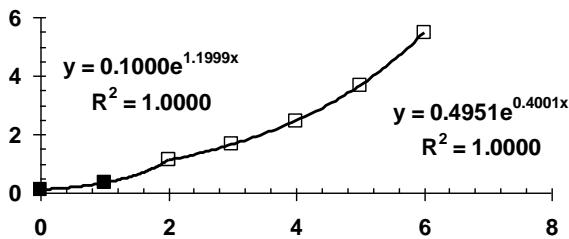
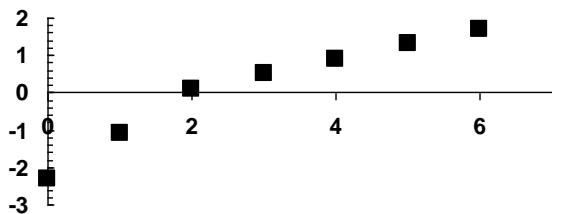
$$s_{y/x} = \sqrt{\frac{S_r}{n-p}}$$

$$n = 15$$

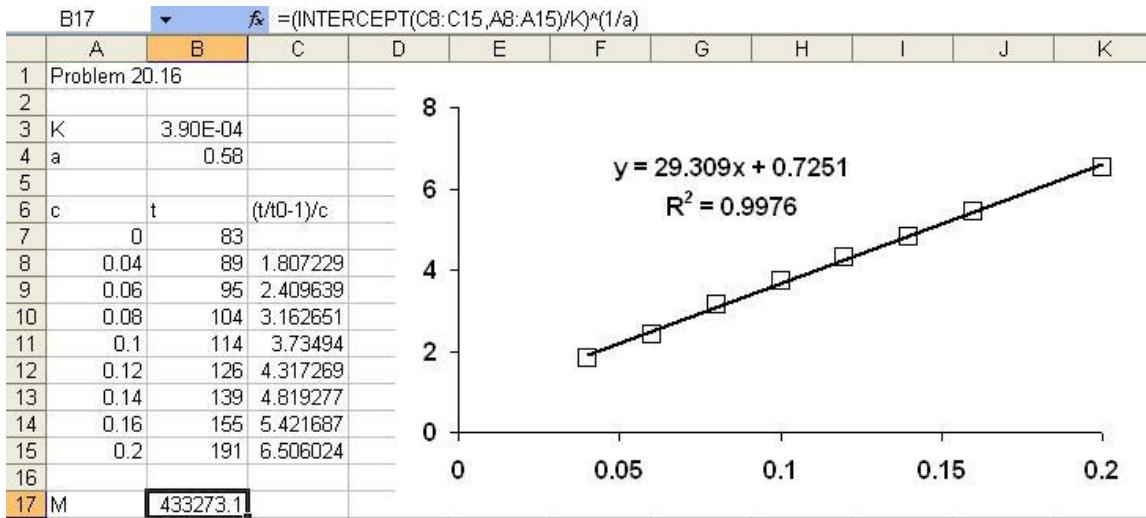
	Model A	Model B	Model C
S_r	135	105	100
Number of model parameters fit (p)	2	3	5
$s_{y/x}$	3.222517	2.95804	3.162278

Thus, Model B seems best because its standard error is lower.

20.15 A plot of the natural log of cells versus time indicates two straight lines with a sharp break at 2. The Excel Trendline tool can be used to fit each range separately with the exponential model as shown in the second plot.



20.16 This problem can be solved with Excel,



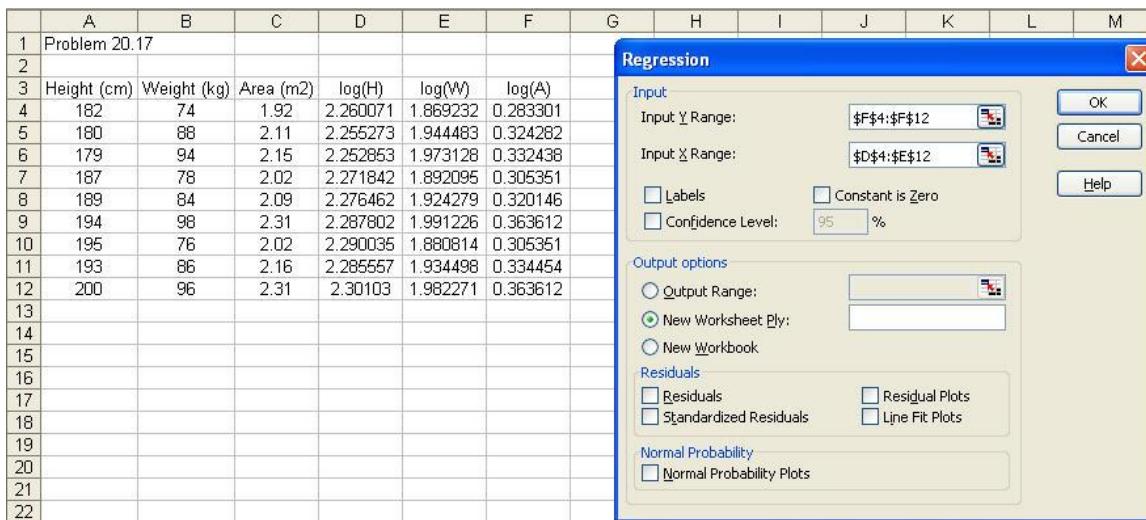
20.17 This problem can be solved with a power model,

$$A = a_0 H^{a_1} W^{a_2}$$

This equation can be linearized by taking the logarithm

$$\log_{10} A = \log_{10} a_0 + a_1 \log_{10} H + a_2 \log_{10} W$$

This model can be fit with the Excel Data Analysis Regression tool,



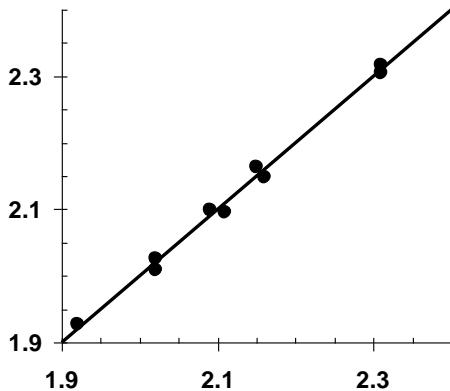
The results are

	A	B	C	D	E	F	G	H	I
1	SUMMARY OUTPUT								
2									
3	Regression Statistics								
4	Multiple R	0.996755676							
5	R Square	0.993521878							
6	Adjusted R Square	0.991362504							
7	Standard Error	0.002470988							
8	Observations	9							
9									
10	ANOVA								
11		df	SS	MS	F	Significance F			
12	Regression	2	0.005618506	0.002809	460.0972	2.71861E-07			
13	Residual	6	3.66347E-05	6.11E-06					
14	Total	8	0.005655141						
15									
16		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95%	Upper 95%
17	Intercept	-1.864582132	0.118395673	-15.7487	4.16E-06	-2.154285906	-1.57488	-2.15429	-1.57488
18	X Variable 1	0.521803441	0.052706914	9.900095	6.13E-05	0.392834268	0.650773	0.392834	0.650773
19	X Variable 2	0.519017838	0.019877438	26.1109	2.08E-07	0.4703795	0.567656	0.470379	0.567656

Therefore the best-fit coefficients are $a_0 = 10^{-1.86458} = 0.013659$, $a_1 = 0.5218$, and $a_2 = 0.5190$. Therefore, the model is

$$A = 0.013659 H^{0.5218} W^{0.5190}$$

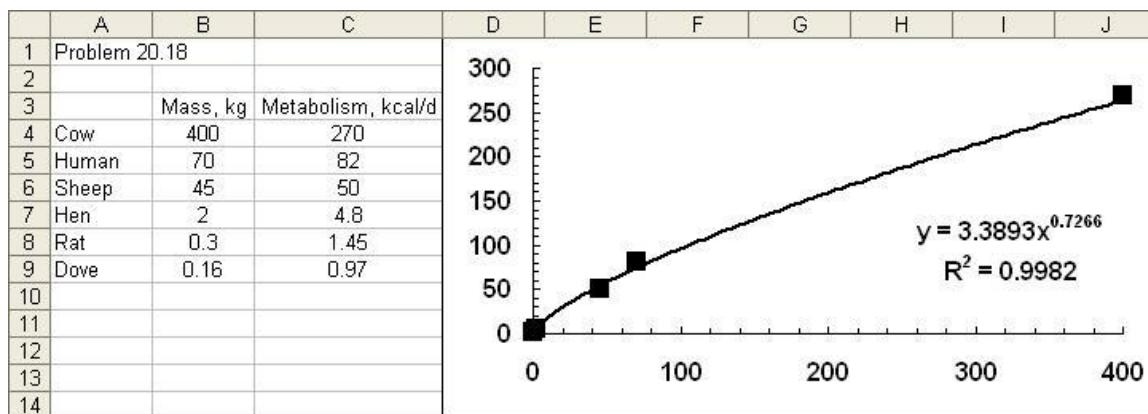
We can assess the fit by plotting the model predictions versus the data. We have included the perfect 1:1 line for comparison.



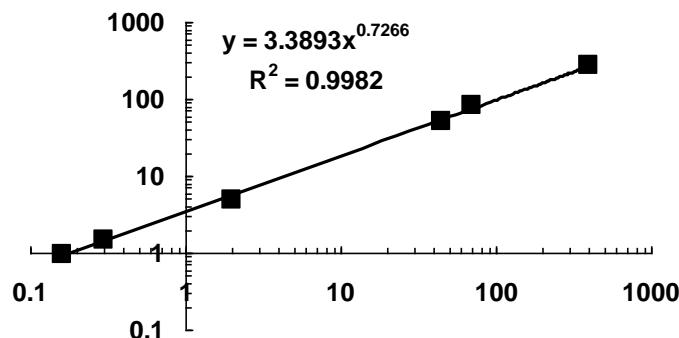
The prediction at $H = 187$ cm and $W = 78$ kg can be computed as

$$A = 0.013659(187)^{0.5218}(78)^{0.5190} = 2.008646$$

20.18 The Excel Trend Line tool can be used to fit a power law to the data:



Note that although the model seems to represent a good fit, the performance of the lower-mass animals is not evident because of the scale. A log-log plot provides a better perspective to assess the fit:



20.19 For the Casson Region, linear regression yields

$$\sqrt{\tau} = 0.065818 + 0.180922 \sqrt{\dot{\gamma}} \quad r^2 = 0.99985$$

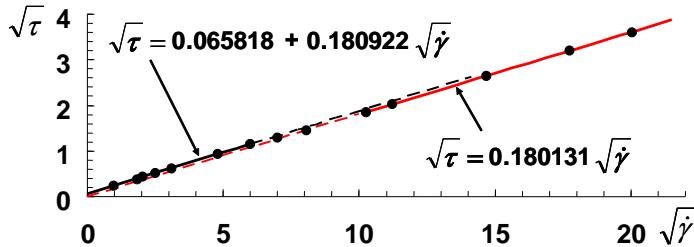
For the Newton Region, linear regression with zero intercept gives

$$\tau = 0.032447 \dot{\gamma} \quad r^2 = 0.99992$$

On a Casson plot, this function becomes

$$\sqrt{\tau} = 0.180131 \sqrt{\dot{\gamma}}$$

We can plot both functions along with the data on a Casson plot



20.20 Recall from Sec. 4.1.3, that finite divided differences can be used to estimate the derivatives. For the first point, we can use a forward difference (Eq. 4.17)

$$\frac{d\sigma}{d\varepsilon} = \frac{96.6 - 87.8}{204 - 153} = 0.1725$$

For the intermediate points, we can use centered differences. For example, for the second point

$$\frac{d\sigma}{d\varepsilon} = \frac{176 - 87.8}{255 - 153} = 0.8647$$

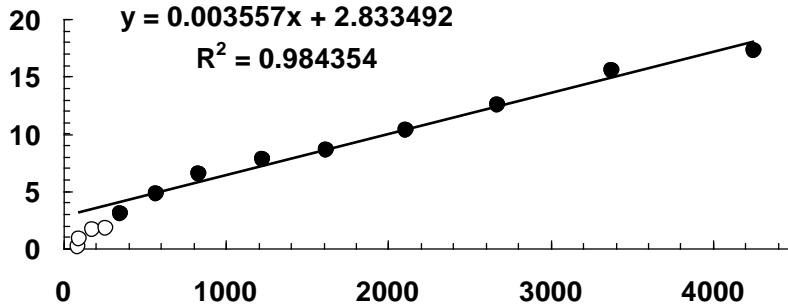
For the last point, we can use a backward difference

$$\frac{d\sigma}{d\varepsilon} = \frac{4258 - 3380}{765 - 714} = 17.2157$$

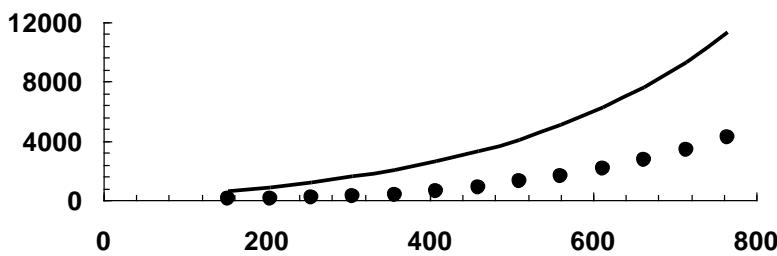
All the values can be tabulated as

σ	ε	$d\sigma/d\varepsilon$
87.8	153	0.1725
96.6	204	0.8647
176	255	1.6314
263	306	1.7157
351	357	3.0196
571	408	4.7353
834	459	6.4510
1229	510	7.7451
1624	561	8.6078
2107	612	10.3333
2678	663	12.4804
3380	714	15.4902
4258	765	17.2157

We can plot these results and after discarding the first few points, we can fit a straight line as shown (note that the discarded points are displayed as open circles).



Therefore, the parameter estimates are $E_o = 2.833492$ and $a = 0.003557$. The data along with the first equation can be plotted as



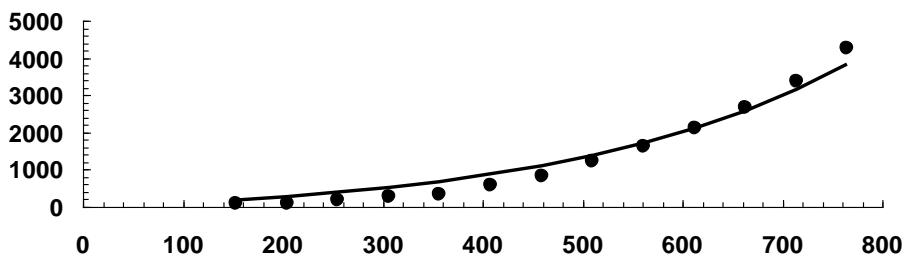
As described in the problem statement, the fit is not very good. We therefore pick a point near the midpoint of the data interval

$$(\bar{\varepsilon} = 612, \bar{\sigma} = 2107)$$

We can then plot the second model

$$\sigma = \left(\frac{2107}{e^{0.003557(612)} - 1} \right) (e^{0.003557\varepsilon} - 1) = 269.5(e^{0.003557\varepsilon} - 1)$$

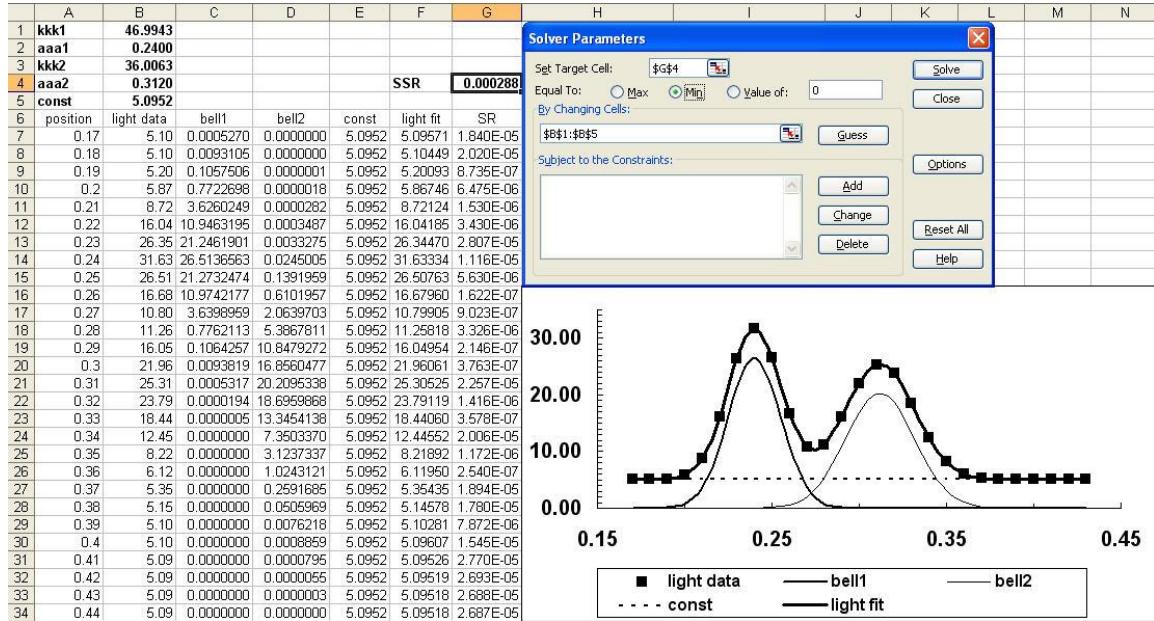
The result is a much better fit



20.21 The problem is set up as the following Excel Solver application. Notice that we have assumed that the model consists of a constant plus two bell-shaped curves:

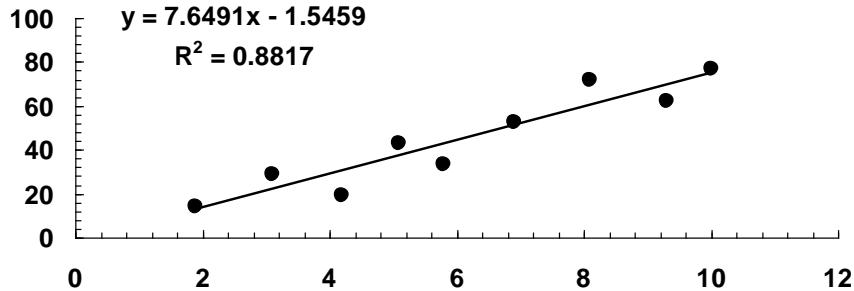
$$f(x) = c + \frac{k_1 e^{-k_1^2(x-a_1)^2}}{\sqrt{\pi}} + \frac{k_2 e^{-k_2^2(x-a_2)^2}}{\sqrt{\pi}}$$

The resulting solution is



Thus, the retina thickness is estimated as $0.312 - 0.24 = 0.072$.

20.22 Simple linear regression can be applied to yield the following fit

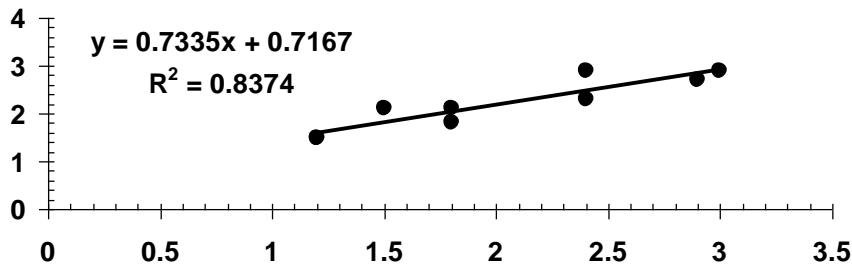


The shear stress at the depth of 4.5 m can be computed as

$$\sigma = 7.6491(4.5) - 1.5459 = 32.875$$

20.23 (a) and (b) Simple linear regression can be applied to yield the following fit

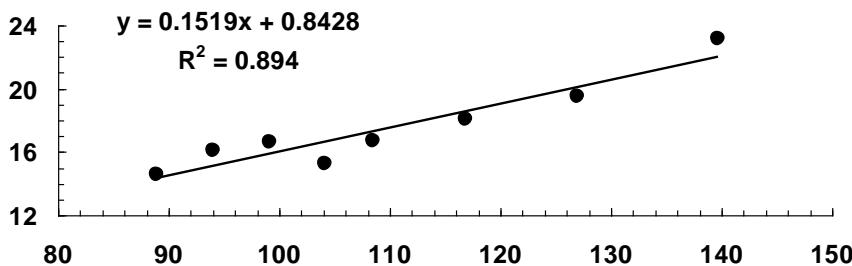
PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.



(c) The minimum lane width corresponding to a bike-car distance of 2 m can be computed as

$$y = 0.7335(2) + 0.7167 = 2.1837 \text{ m}$$

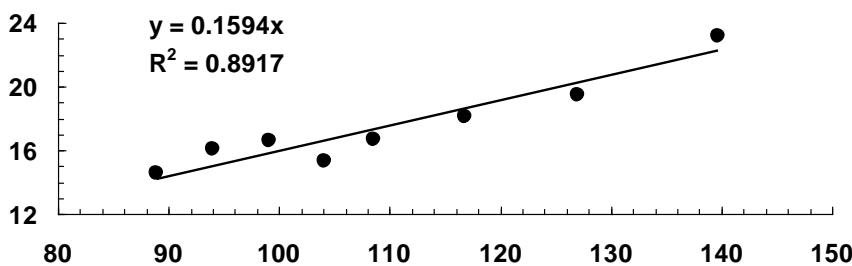
20.24 (a) and (b) Simple linear regression can be applied to yield the following fit



(c) The flow corresponding to the precipitation of 120 cm can be computed as

$$Q = 0.1519(120) + 0.8428 = 19.067$$

(d) We can redo the regression, but with a zero intercept



Thus, the model is

$$Q = 0.1594P$$

where Q = flow and P = precipitation. Now, if there are no water losses, the maximum flow, Q_m , that could occur for a level of precipitation should be equal to the product of the annual precipitation and the drainage area. This is expressed by the following equation.

$$Q_m = A(\text{km}^2)P\left(\frac{\text{cm}}{\text{yr}}\right)$$

For an area of 1100 km^2 and applying conversions so that the flow has units of m^3/s

$$Q_m = 1,100 \text{ km}^2 P\left(\frac{\text{cm}}{\text{yr}}\right) \frac{10^6 \text{ m}^2}{\text{km}^2} \frac{1 \text{ m}}{100 \text{ cm}} \frac{1 \text{ d}}{86,400 \text{ s}} \frac{1 \text{ yr}}{365 \text{ d}}$$

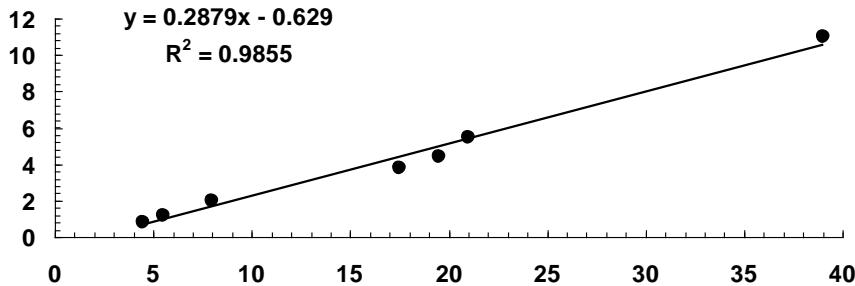
Collecting terms gives

$$Q_m = 0.348808 P$$

Using the slope from the linear regression with zero intercept, we can compute the fraction of the total flow that is lost to evaporation and other consumptive uses can be computed as

$$F = \frac{0.348808 - 0.1594}{0.348808} = 0.543$$

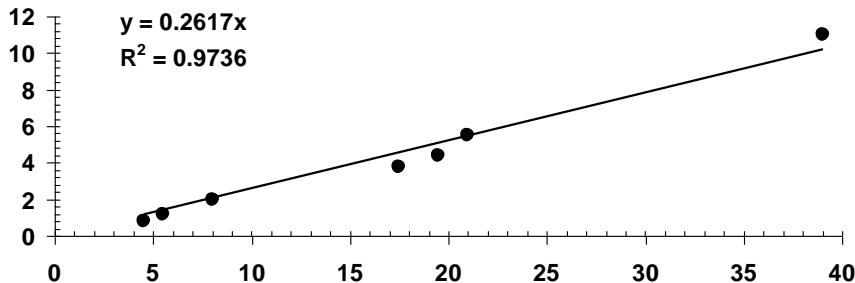
20.25 The data can be computed in a number of ways. First, we can use linear regression.



For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m^3 is

$$c = 0.2879(10) - 0.629 = 2.2495$$

One problem with this result is that the model yields a physically unrealistic negative intercept. In fact, because chlorophyll cannot exist without phosphorus, a fit with a zero intercept would be preferable. Such a fit can be developed as

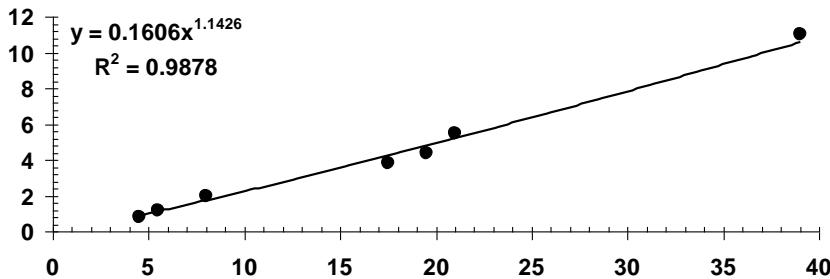


For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m^3 is

$$c = 0.2617(10) = 2.617$$

Thus, as expected, the result differs from that obtained with a nonzero intercept.

Finally, it should be noted that in practice, such data is usually fit with a power model. This model is often adopted because it has a zero intercept while not constraining the model to be linear. When this is done, the result is



For this model, the chlorophyll level in western Lake Erie corresponding to a phosphorus concentration of 10 mg/m^3 is

$$c = 0.1606(10)^{1.1426} = 2.2302$$

20.26

$$m = \frac{4.6}{10} = 0.46 \quad n = \frac{14}{10} = 1.4$$

Use the following values

$$\begin{array}{ll} x_0 = 0.3 & f(x_0) = 0.08561 \\ x_1 = 0.4 & f(x_1) = 0.10941 \\ x_2 = 0.5 & f(x_2) = 0.13003 \\ x_3 = 0.6 & f(x_3) = 0.14749 \end{array}$$

MATLAB can be used to perform the interpolation

```
>> format long
>> x=[0.3 0.4 0.5 0.6];
>> y=[0.08561 0.10941 0.13003 0.14749];
>> a=polyfit(x,y,3)

a =

Columns 1 through 3

0.0033333333331 -0.16299999999997 0.35086666666665

Column 4

-0.00507000000000

>> polyval(a,0.46)

ans =

0.12216232000000
```

This result can be used to compute the vertical stress

$$q = \frac{100}{4.6(14)} = 1.552795$$

$$\sigma_z = 1.552795(0.1221623) = 0.189693$$

20.27 This is an ideal problem for general linear least squares. The problem can be solved with MATLAB:

```
>> t=[0.5 1 2 3 4 5 6 7 9]';
>> p=[6 4.4 3.2 2.7 2.2 1.9 1.7 1.4 1.1]';
>> Z=[exp(-1.5*t) exp(-0.3*t) exp(-0.05*t)];
>> a=inv(Z'*Z)*(Z'*p)

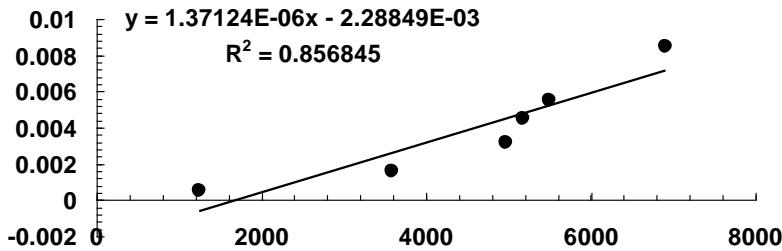
a =
    4.0046
    2.9213
    1.5647
```

Therefore, $A = 4.0046$, $B = 2.9213$, and $C = 1.5647$.

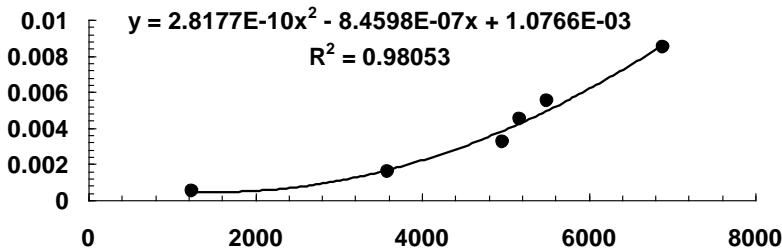
20.28 First, we can determine the stress

$$\sigma = \frac{25000}{10.65} = 2,347.418$$

We can then try to fit the data to obtain a mathematical relationship between strain and stress. First, we can try linear regression:



This is not a particularly good fit as the r^2 is relatively low. We therefore try a best-fit parabola,



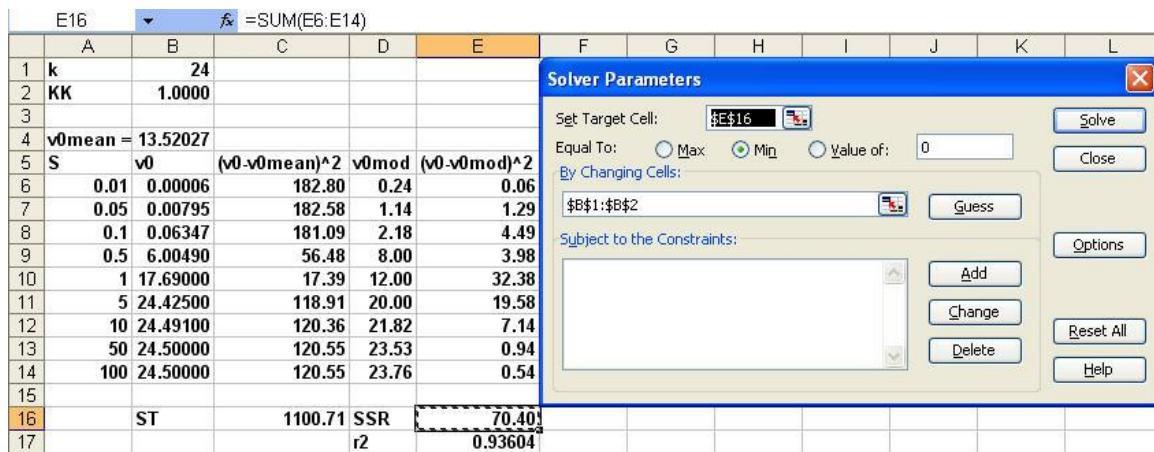
We can use this model to compute the strain as

$$\varepsilon = 2.8177 \times 10^{-10} (2347.4178)^2 - 8.4598 \times 10^{-7} (2347.4178) + 1.0766 \times 10^{-3} = 6.4341 \times 10^{-4}$$

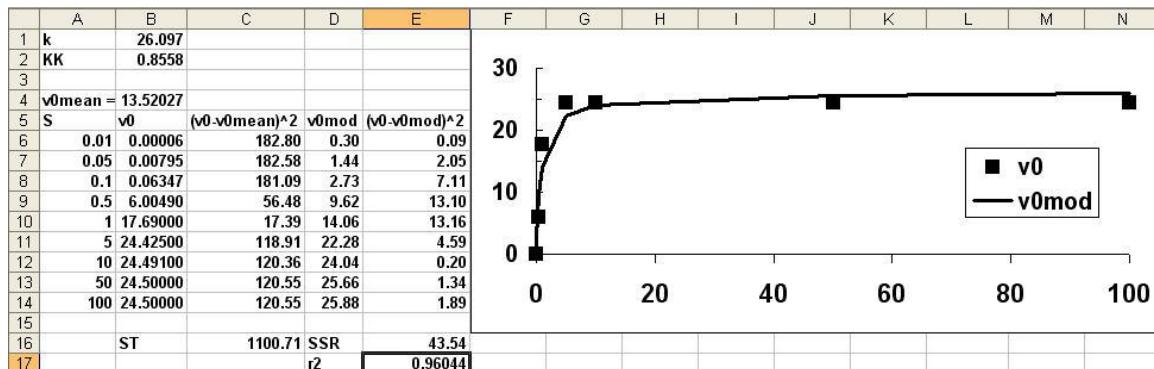
The deflection can be computed as

$$\Delta L = 6.4341 \times 10^{-4} (9) = 0.0057907 \text{ m}$$

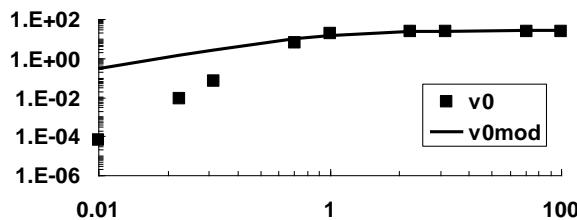
20.29 Clearly the linear model is not adequate. The second model can be fit with the Excel Solver:



Notice that we have reexpressed the initial rates by multiplying them by 1×10^6 . We did this so that the sum of the squares of the residuals would not be minuscule. Sometimes this will lead the Solver to conclude that it is at the minimum, even though the fit is poor. The solution is:

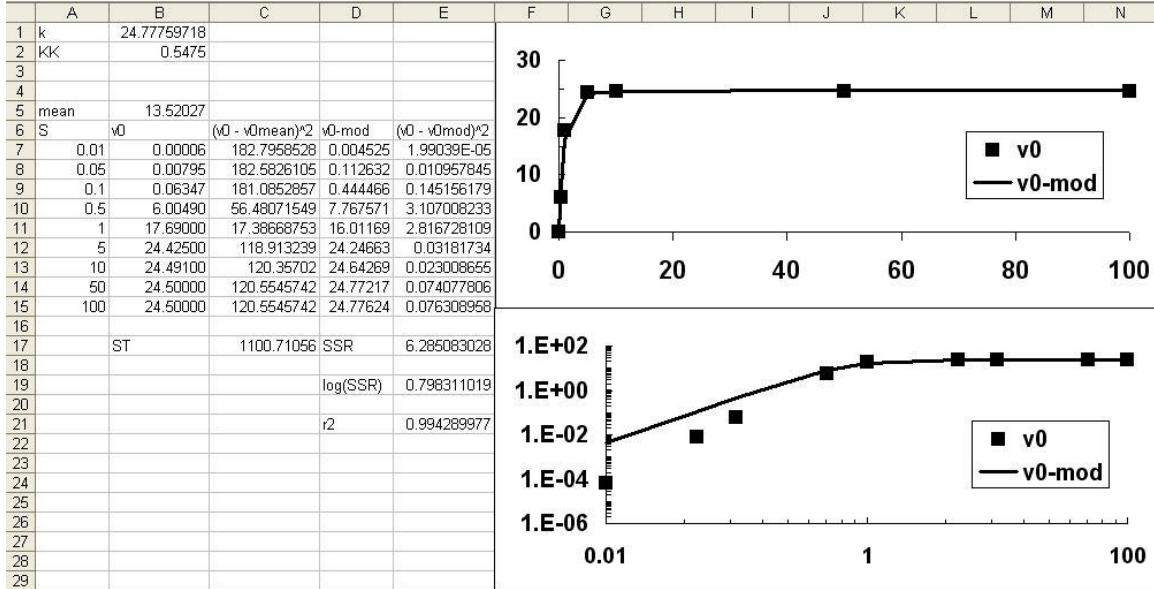


Although the fit might appear to be OK, it is biased in that it underestimates the low values and overestimates the high ones. The poorness of the fit is really obvious if we display the results as a log-log plot:

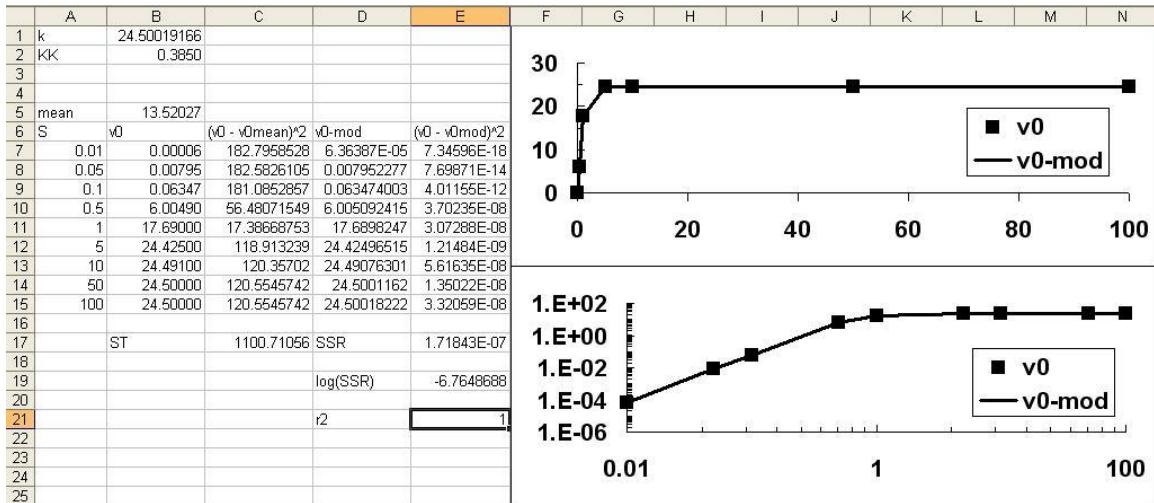


Notice that this view illustrates that the model actually overpredicts the very lowest values.

The third and fourth models provide a means to rectify this problem. Because they raise [S] to powers, they have more degrees of freedom to follow the underlying pattern of the data. For example, the third model gives:



Finally, the cubic model results in a perfect fit:



Thus, the best fit is

$$v_0 = \frac{2.45 \times 10^{-5} [S]^3}{0.385 + [S]^3}$$

20.30 As described in Section 17.1, we can use general linear least squares to generate the best-fit equation. We can use a number of different software tools to do this. For example, the $[Z]$ and y matrices can be set up using MATLAB commands as

```
>> format long
>> x = [273.15 283.15 293.15 303.15 313.15]';
>> Kw = [1.164e-15 2.950e-15 6.846e-15 1.467e-14 2.929e-14]';
>> y=-log10 (Kw);
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```
>> Z = [ (1./x) log10(x) x ones(size(x))];
```

The coefficients can be evaluated as

```
>> a=inv(Z'*Z)*(Z'*y)
Warning: Matrix is close to singular or badly scaled.
          Results may be inaccurate. RCOND = 6.457873e-020.

a =
1.0e+003 *
5.18067187500000
0.01342456054688
0.00000562858582
-0.03827636718750
```

Note the warning that the results are ill-conditioned. According to this calculation, the best-fit model is

$$-\log_{10} K_w = \frac{5180.67}{T_a} + 13.42456 \log_{10} T_a + 0.00562859 T_a - 38.276367$$

We can check the results by using the model to make predictions at the values of the original data

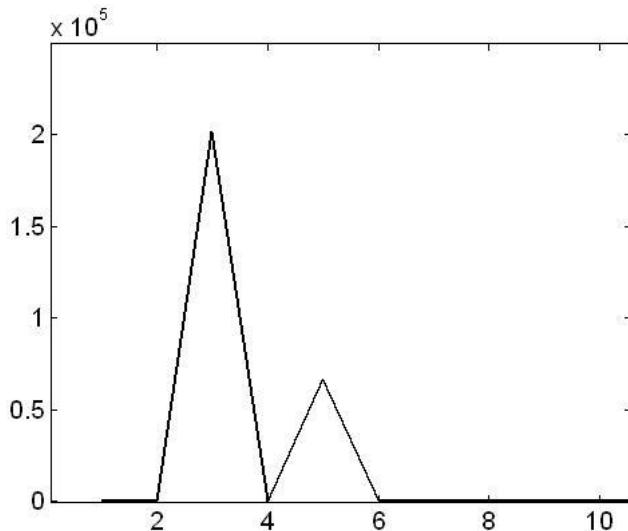
```
>> yp=10.^-(a(1)./x+a(2)*log10(x)+a(3)*x+a(4))

yp =
1.0e-013 *
0.01161193308242
0.02943235714551
0.06828461729494
0.14636330575049
0.29218444886852
```

These results agree to about 2 or 3 significant digits with the original data.

20.31 Using MATLAB:

```
>> t=0:2*pi/128:2*pi;
>> f=4*cos(5*t)-7*sin(3*t)+6;
>> y=fft(f);
>> y(1)=[];
>> n=length(y);
>> power=abs(y(1:n/2)).^2;
>> nyquist=1/2;
>> freq=n*(1:n/2)/(n/2)*nyquist;
>> plot(freq,power);
```



20.32 Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

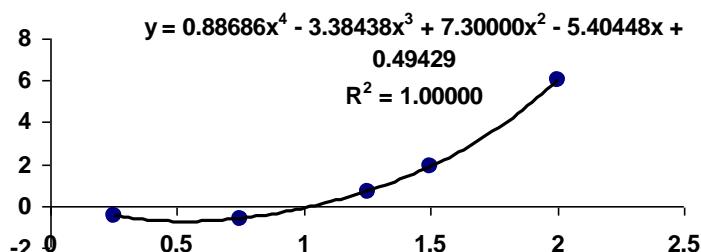
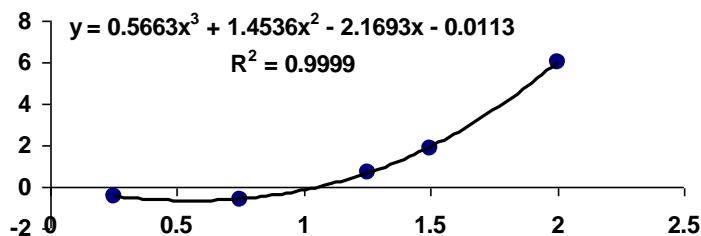
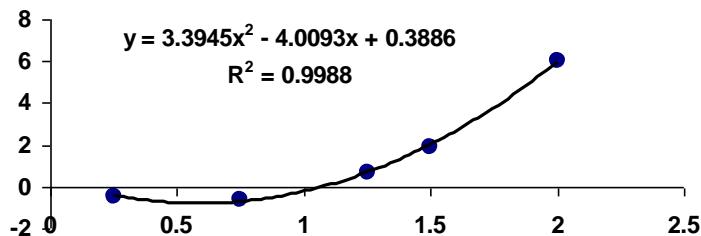
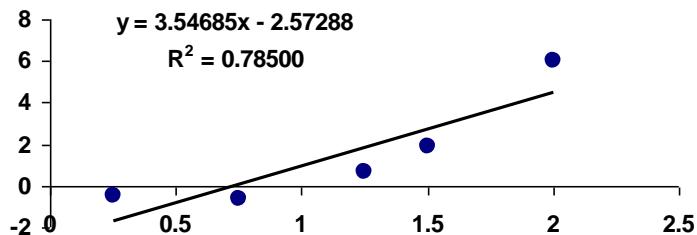
$$\begin{aligned}
 x_0 &= 1.25 & f(x_0) &= 0.7 \\
 x_1 &= 0.75 & f(x_1) &= -0.6 \\
 x_2 &= 1.5 & f(x_2) &= 1.88 \\
 x_3 &= 0.25 & f(x_3) &= -0.45 \\
 x_4 &= 2 & f(x_4) &= 6
 \end{aligned}$$

The results of applying Newton's polynomial at $i = 1.15$ are

Order	$f(x)$	Error
0	0.7	-0.26
1	0.44	-0.11307
2	0.326933	-0.00082
3	0.326112	0.011174
4	0.337286	

The minimum error occurs for the second-order version so we conclude that the interpolation is 0.3269.

20.33 Here are the results of first through fourth-order regression:



The 2nd through 4th-order polynomials all seem to capture the general trend of the data. Each of the polynomials can be used to make the prediction at $i = 1.15$ with the results tabulated below:

Order	Prediction
1	1.5060
2	0.2670
3	0.2776
4	0.3373

Thus, although the 2nd through 4th-order polynomials all seem to follow a similar trend, they yield quite different predictions. The results are so sensitive because there are few data points.

20.34 Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$$x_0 = 0.25 \quad f(x_0) = 7.75$$

$$x_1 = 0.125 \quad f(x_1) = 6.24$$

$$x_2 = 0.375 \quad f(x_2) = 4.85$$

$$x_3 = 0 \quad f(x_3) = 0$$

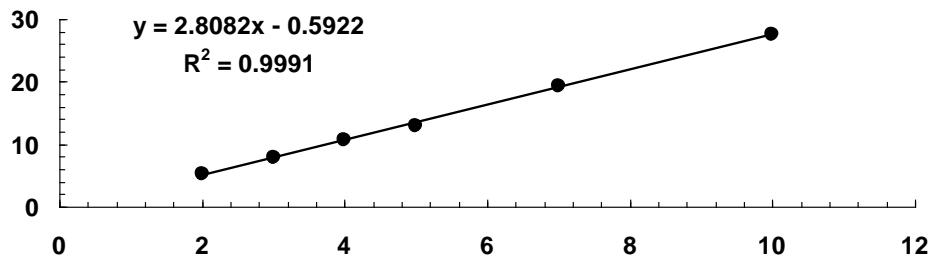
$$x_4 = 0.5 \quad f(x_4) = 0$$

The results of applying Newton's polynomial at $t = 0.23$ are

Order	f(x)	Error
0	7.75	-0.2416
1	7.5084	0.296352
2	7.804752	0.008315
3	7.813067	0.025579
4	7.838646	

The minimum error occurs for the second-order version so we conclude that the interpolation is 7.805.

20.35(a) The linear fit is

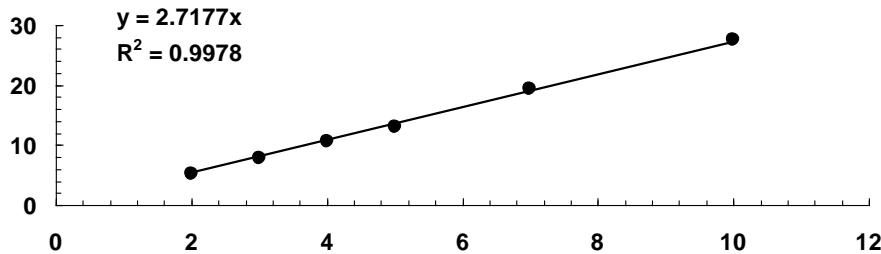


The current for a voltage of 3.5 V can be computed as

$$y = 2.8082(3.5) - 0.5922 = 9.2364$$

Both the graph and the r^2 indicate that the fit is good.

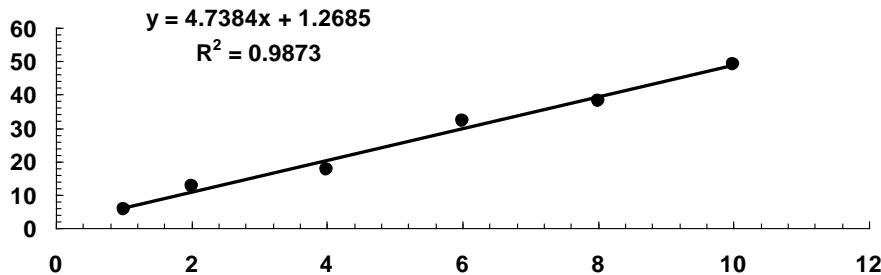
(b) A straight line with zero intercept can be fit as



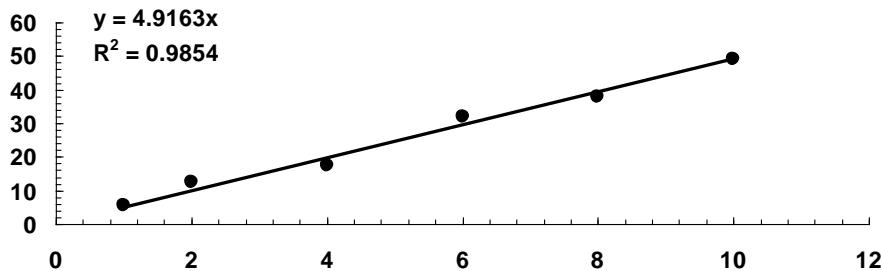
For this case, the current at $V = 3.5$ can be computed as

$$y = 2.7177(3.5) = 9.512$$

20.36 The linear fit is



Therefore, an estimate for L is 4.7384. However, because there is a non-zero intercept, a better approach would be to fit the data with a linear model with a zero intercept



This fit is almost as good as the first case, but has the advantage that it has the physically more realistic zero intercept. Thus, a superior estimate of L is 4.9163.

20.37 Since we do not know the proper order of the interpolating polynomial, this problem is suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$$\begin{aligned} x_0 &= 0.5 & f(x_0) &= 20.5 \\ x_1 &= -0.5 & f(x_1) &= -20.5 \\ x_2 &= 1 & f(x_2) &= 96.5 \end{aligned}$$

$$\begin{array}{ll} x_3 = -1 & f(x_3) = -96.5 \\ x_4 = 2 & f(x_4) = 637 \\ x_5 = -2 & f(x_5) = -637 \end{array}$$

The results of applying Newton's polynomial at $i = 0.1$ are

Order	$f(x)$	Error
0	20.5	-16.4
1	4.1	-17.76
2	-13.66	15.984
3	2.324	0
4	2.324	0
5	2.324	

Thus, we can see that the data was generated with a cubic polynomial.

20.38 Because there are 6 points, we can fit a 5th-order polynomial. This can be done with Eq. 18.26 or with a software package like Excel or MATLAB that is capable of evaluating the coefficients. For example, using MATLAB,

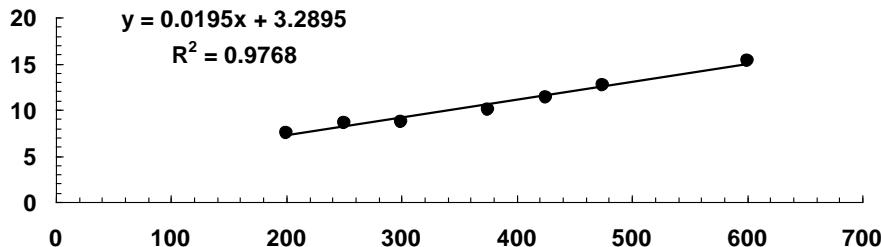
```
>> x=[-2 -1 -0.5 0.5 1 2];
>> y=[-637 -96.5 -20.5 20.5 96.5 637];
>> a=polyfit(x,y,5)

a =
    0.0000    0.0000    74.0000   -0.0000    22.5000    0.0000
```

Thus, we see that the polynomial is

$$V = 74i^3 + 22.5i$$

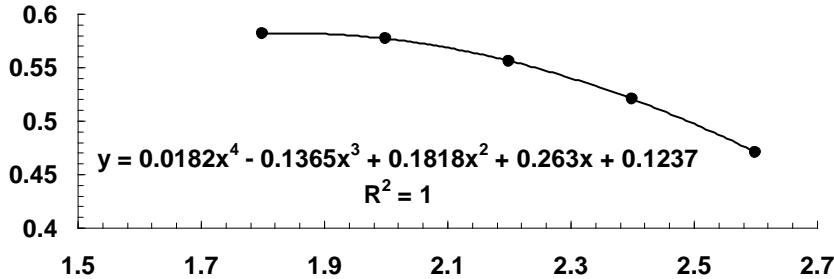
20.39 Linear regression yields



The percent elongation for a temperature of 400 can be computed as

$$\% \text{ elongation} = 0.0195(400) + 3.2895 = 11.072$$

20.40 (a) A 4th-order interpolating polynomial can be generated as



The polynomial can be used to compute

$$J_1(2.1) = 0.018229(2.1)^4 - 0.13646(2.1)^3 + 0.181771(2.1)^2 + 0.262958(2.1) + 0.1237 = 0.568304$$

The relative error is

$$\varepsilon_t = \left| \frac{0.568292 - 0.568304}{0.568292} \right| \times 100\% = 0.0021\%$$

Thus, the interpolating polynomial yields an excellent result.

(b) A program can be developed to fit natural cubic splines through data based on Fig. 18.18. If this program is run with the data for this problem, the interpolation at 2.1 is 0.56846 which has a relative error of $\varepsilon_t = 0.0295\%$.

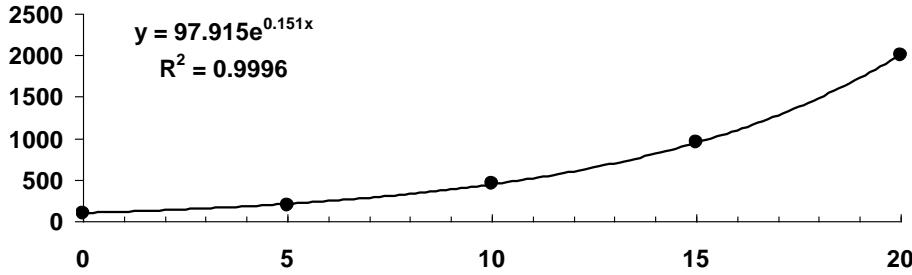
A spline can also be fit with MATLAB. It should be noted that MATLAB does not use a natural spline. Rather, it uses a so-called “not-a-knot” spline. Thus, as shown below, although it also yields a very good prediction, the result differs from the one generated with the natural spline,

```
>> format long
>> x=[1.8 2 2.2 2.4 2.6];
>> y=[0.5815 0.5767 0.556 0.5202 0.4708];
>> spline(x,y,2.1)

ans =
    0.56829843750000
```

This result has a relative error of $\varepsilon_t = 0.0011\%$.

20.41 The fit of the exponential model is



The model can be used to predict the population 5 years in the future as

$$p = 97.91484 e^{0.150992(25)} = 4268$$

20.42 The prediction using the model from Sec. 20.4 is

$$Q = 55.9(1.23)^{2.62}(0.001)^{0.54} = 2.30655$$

The linear prediction is

$$\begin{array}{ll} x_0 = 1 & f(x_0) = 1.4 \\ x_1 = 2 & f(x_1) = 8.3 \end{array}$$

$$Q = 1.4 + \frac{8.3 - 1.4}{2 - 1}(1.23 - 1) = 2.987$$

The quadratic prediction is

$$\begin{array}{ll} x_0 = 1 & f(x_0) = 1.4 \\ x_1 = 2 & f(x_1) = 8.3 \\ x_2 = 3 & f(x_2) = 24.2 \end{array}$$

$$Q = 2.987 + \frac{\frac{24.2 - 8.3}{3 - 2} - \frac{8.3 - 1.4}{2 - 1}}{3 - 1}(1.23 - 1)(1.23 - 2) = 2.19005$$

20.43 The model to be fit is

$$S = b_0 D^{b_1} Q^{b_2}$$

Taking the common logarithm gives

$$\log_{10} S = \log_{10} b_0 + b_1 \log_{10} D + b_2 \log_{10} Q$$

We can use a number of different approaches to fit this model. The following shows how it can be done with MATLAB,

```

>> D=[1 2 3 1 2 3 1 2 3]';
>> S=[0.001 0.001 0.001 0.01 0.01 0.01 0.05 0.05 0.05]';
>> Q=[1.4 8.3 24.2 4.7 28.9 84 11.1 69 200]';
>> Z=[ones(size(S)) log10(D) log10(Q)];
>> y=log10(S);
>> a=inv(Z'*Z)*(Z'*y)

a =
-3.2563
-4.8726
1.8627

```

Therefore, the result is

$$\log_{10} S = -3.2563 - 4.8726 \log_{10} D + 1.862733 \log_{10} Q$$

or in untransformed format

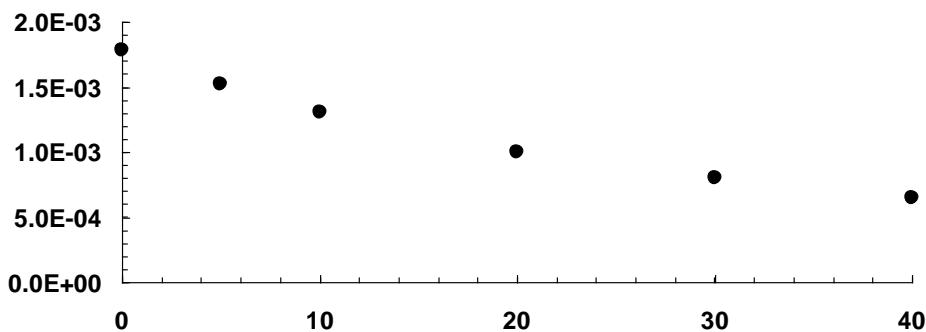
$$S = 0.000554 D^{-4.8726} Q^{1.862733} \quad (1)$$

We can compare this equation with the original model developed in Sec. 20.4 by solving Eq. 1 for Q ,

$$Q = 55.9897 D^{2.6158} S^{0.5368}$$

This is very close to the original model.

20.44 (a) The data can be plotted as



(b) This part of the problem is well-suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible.

$$\begin{aligned}
x_0 &= 5 & f(x_0) &= 1.519 \times 10^{-3} \\
x_1 &= 10 & f(x_1) &= 1.307 \times 10^{-3} \\
x_2 &= 0 & f(x_2) &= 1.787 \times 10^{-3} \\
x_3 &= 20 & f(x_3) &= 1.002 \times 10^{-3} \\
x_4 &= 30 & f(x_4) &= 0.7975 \times 10^{-3}
\end{aligned}$$

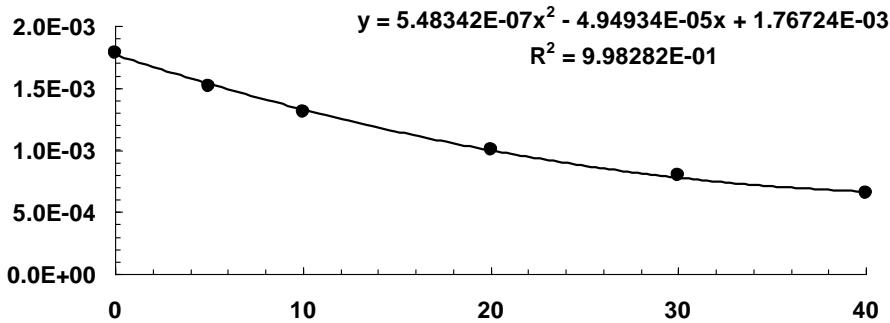
$$x_5 = 40 \quad f(x_5) = 0.6529 \times 10^{-3}$$

The results of applying Newton's polynomial at $T = 7.5$ are

Order	$f(x)$	Error
0	1.51900E-03	-0.00011
1	1.41300E-03	-7E-06
2	1.40600E-03	7.66E-07
3	1.40677E-03	9.18E-08
4	1.40686E-03	5.81E-09
5	1.40686E-03	

The minimum error occurs for the fourth-order version so we conclude that the interpolation is 1.40686×10^{-3} .

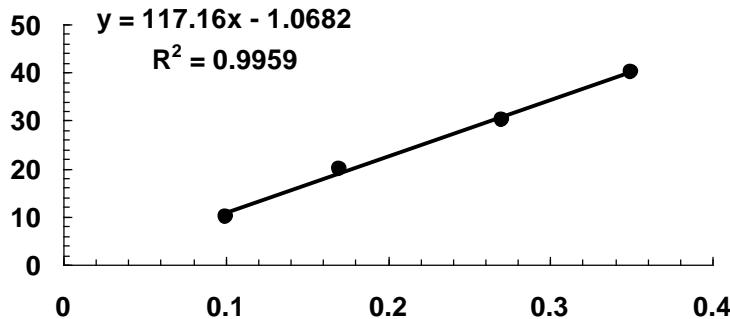
(b) Polynomial regression yields



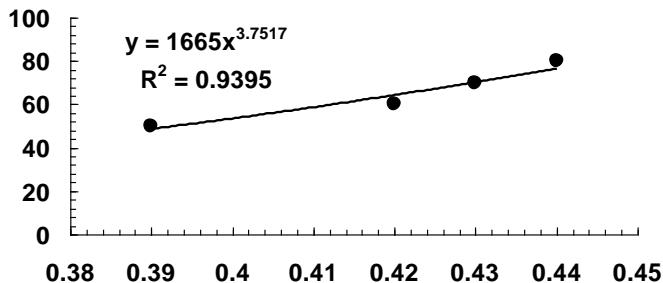
This leads to a prediction of

$$\mu = 5.48342 \times 10^{-7} (7.5)^2 - 4.94934 \times 10^{-5} (7.5) + 1.76724 \times 10^{-3} = 1.4269 \times 10^{-3}$$

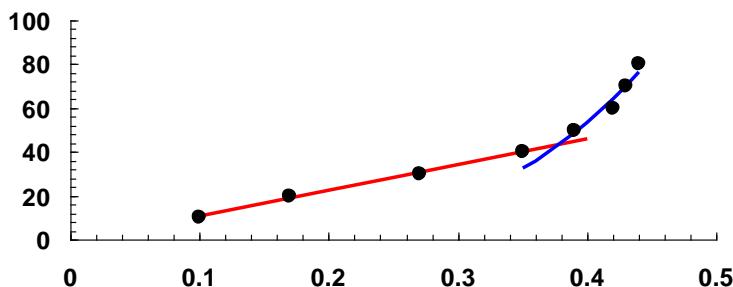
20.45 For the first four points, linear regression yields



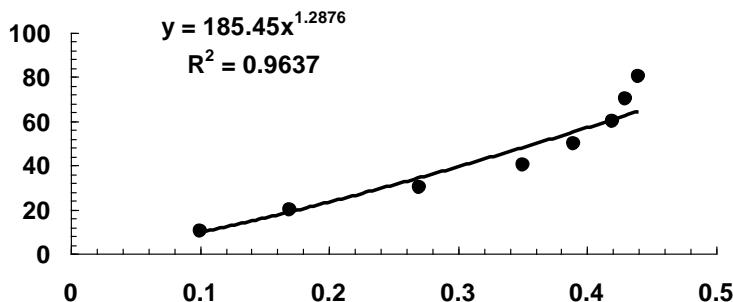
We can fit a power equation to the last four points.



Both curves can be plotted along with the data.

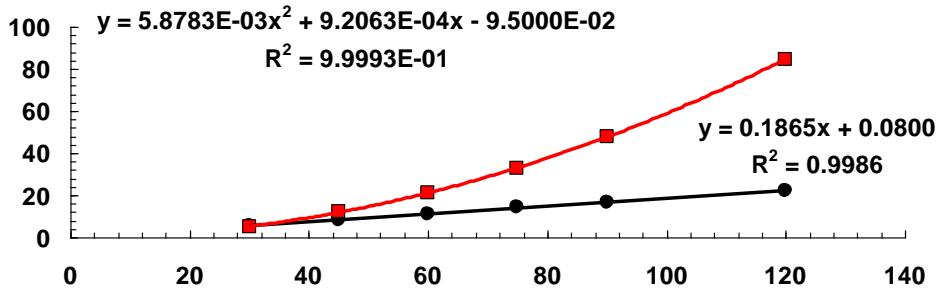


20.46 A power fit to all the data can be implemented as in the following plot.



Although the coefficient of determination is relatively high ($r^2 = 0.9637$), this fit is not very acceptable. In contrast, the piecewise fit from Prob. 20.45 does a much better job of tracking on the trend of the data.

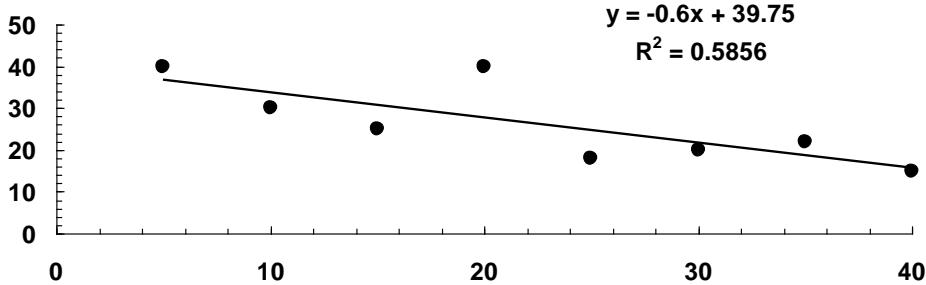
20.47 The “Thinking” curve can be fit with a linear model whereas the “Braking” curve can be fit with a quadratic model as in the following plot.



A prediction of the total stopping distance for a car traveling at 110 km/hr can be computed as

$$d = 0.1865(110) + 0.0800 + 5.8783 \times 10^{-3}(110)^2 + 9.2063 \times 10^{-4}(110) - 9.5000 \times 10^{-2} = 91.726 \text{ m}$$

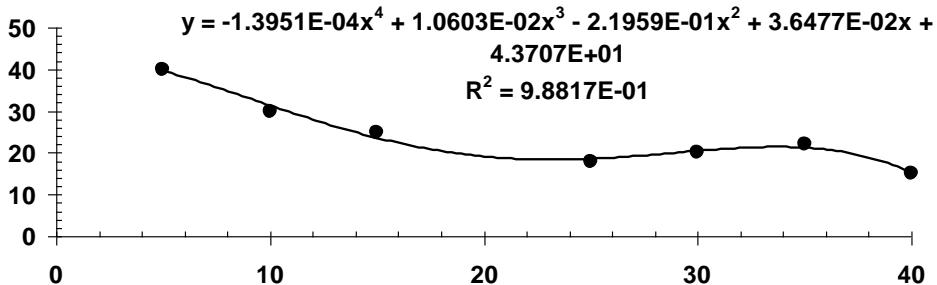
20.48 Using linear regression gives



A prediction of the fracture time for an applied stress of 20 can be computed as

$$t = -0.6(20) + 39.75 = 27.75$$

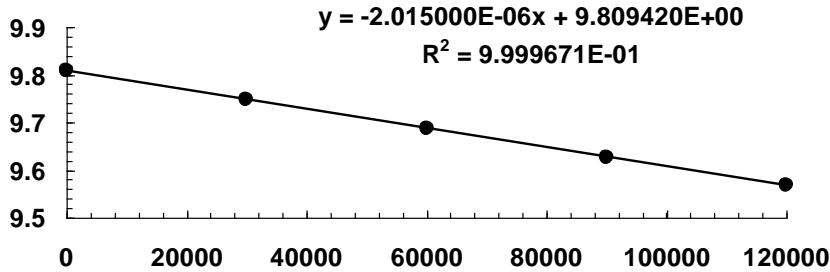
Some students might consider the point at (20, 40) as an outlier and discard it. If this is done, 4th-order polynomial regression gives the following fit



A prediction of the fracture time for an applied stress of 20 can be computed as

$$t = -0.00013951(20)^4 + 0.010603(20)^3 - 0.21959(20)^2 + 0.036477(20) + 43.707 = 19.103$$

20.49 Using linear regression gives



A prediction of g at 55,000 m can be made as

$$g(55,000) = -2.015 \times 10^{-6} (55,000) + 9.80942 = 9.6986$$

Note that we can also use linear interpolation to give

$$g_1(55,000) = 9.698033$$

Quadratic interpolation yields

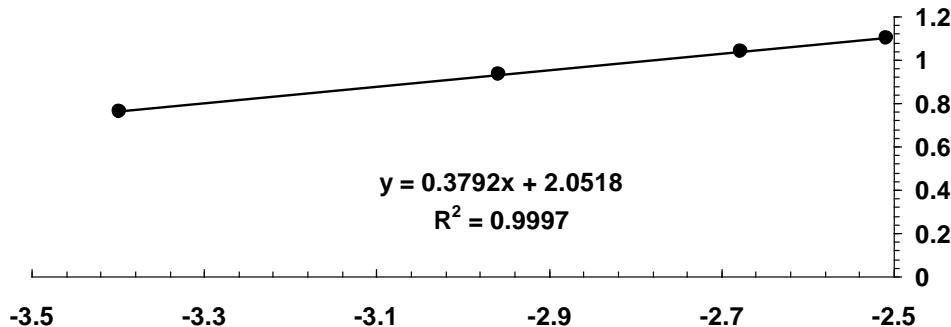
$$g_2(55,000) = 9.697985$$

Cubic interpolation gives

$$g_3(55,000) = 9.69799$$

Based on all these estimates, we can be confident that the result to 3 significant digits is 9.698.

20.50 A plot of $\log_{10} \dot{\varepsilon}$ versus $\log_{10} \sigma$ can be developed as



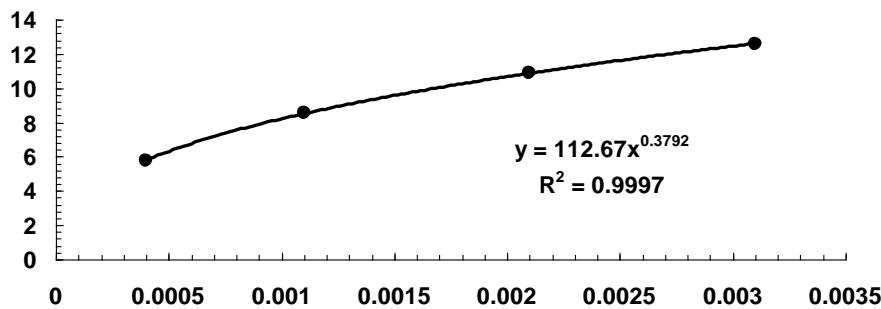
As shown, the best fit line is

$$\log_{10} \dot{\varepsilon} = 0.3792 \log_{10} \sigma + 2.0518$$

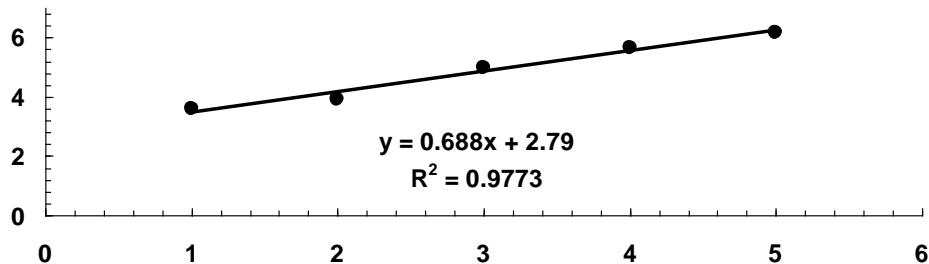
Therefore, $B = 10^{2.0518} = 112.67$ and $m = 0.3792$, and the power model is

$$y = 112.67x^{0.3792}$$

The model and the data can be plotted as

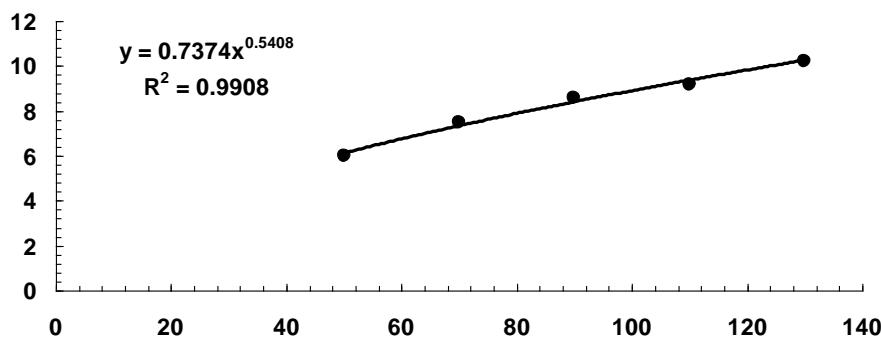


20.51 Using linear regression gives



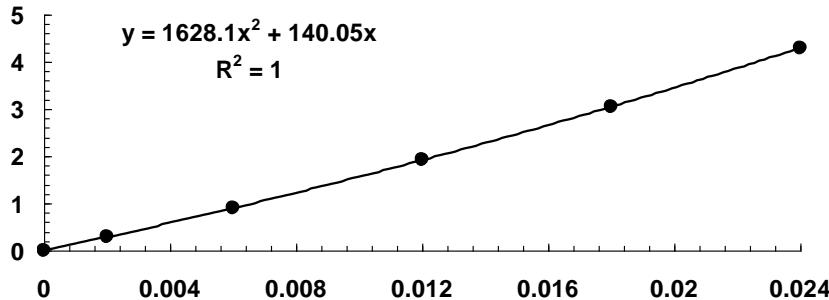
Therefore, $\tau_y = 2.79$ and $\mu = 0.688$.

20.52 A power fit can be developed as



Therefore, $\mu = 0.7374$ and $n = 0.5408$.

20.53 We fit a number of curves to this data and obtained the best fit with a second-order polynomial with zero intercept



Therefore, the best-fit curve is

$$u = 1628.1y^2 + 140.05y$$

We can differentiate this function

$$\frac{du}{dy} = 3256.2y + 140.05$$

Therefore, the derivative at the surface is 140.05 and the shear stress can be computed as $1.8 \times 10^{-5}(140.05) = 0.002521 \text{ N/m}^2$.

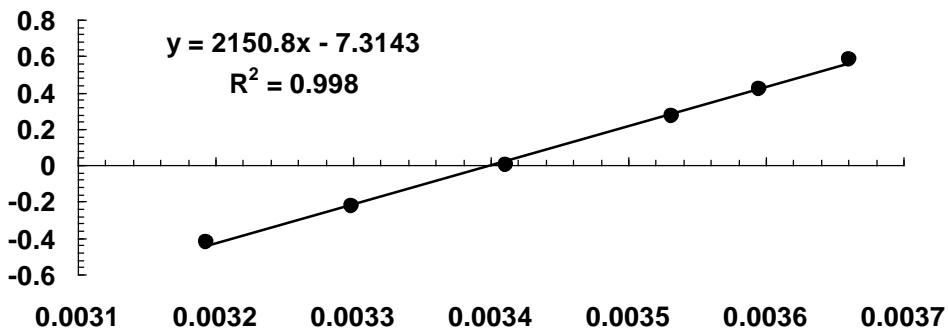
20.54 We can use transformations to linearize the model as

$$\ln \mu = \ln D + B \frac{1}{T_a}$$

Thus, we can plot the natural log of μ versus $1/T_a$ and use linear regression to determine the parameters. Here is the data showing the transformations.

T	μ	T_a	$1/T_a$	$\ln \mu$
0	1.787	273.15	0.003661	0.580538
5	1.519	278.15	0.003595	0.418052
10	1.307	283.15	0.003532	0.267734
20	1.002	293.15	0.003411	0.001998
30	0.7975	303.15	0.003299	-0.226267
40	0.6529	313.15	0.003193	-0.42633

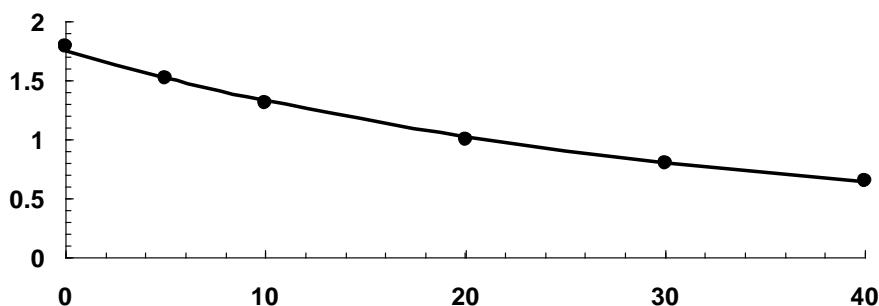
Here is the fit:



Thus, the parameters are estimated as $D = e^{-7.3143} = 6.65941 \times 10^{-4}$ and $B = 2150.8$, and the Andrade equation is

$$\mu = 6.65941 \times 10^{-4} e^{2150.8/Ta}$$

This equation can be plotted along with the data

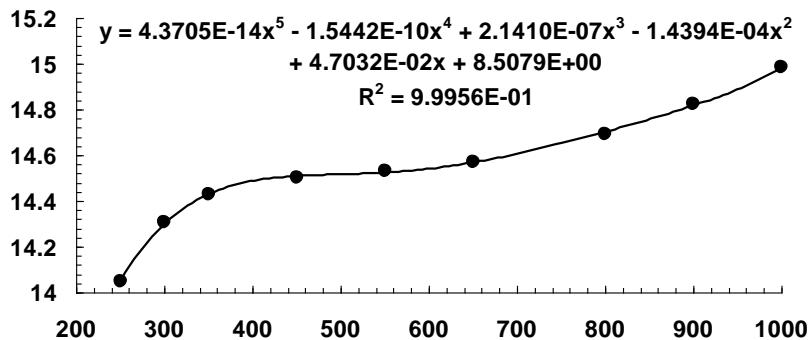


Note that this model can also be fit with nonlinear regression. If this is done, the result is

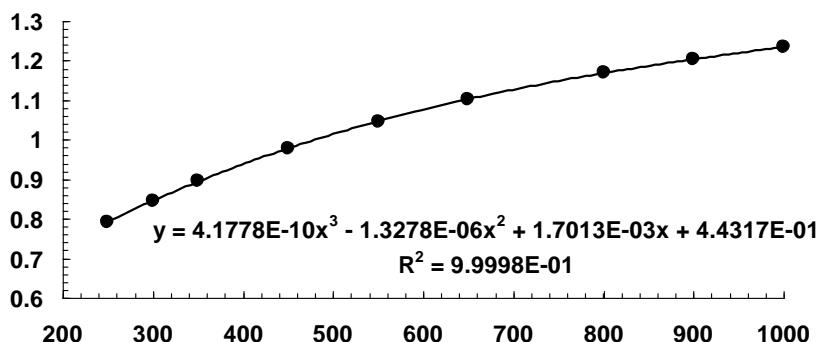
$$\mu = 5.39872 \times 10^{-4} e^{2210.66/Ta}$$

Although it is difficult to discern graphically, this fit is slightly superior ($r^2 = 0.99816$) to that obtained with the transformed model ($r^2 = 0.99757$).

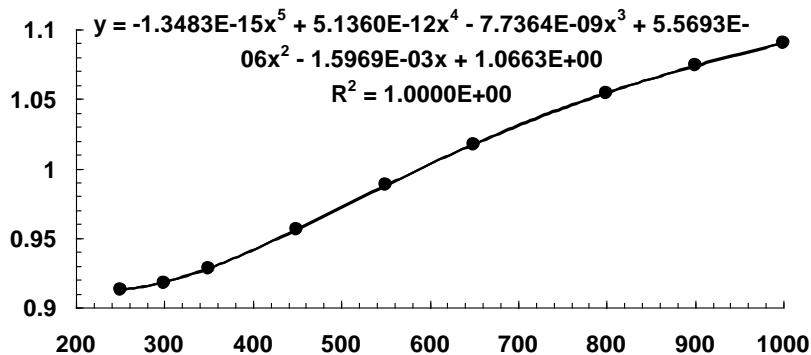
20.55 Hydrogen: Fifth-order polynomial regression provides a good fit to the data,



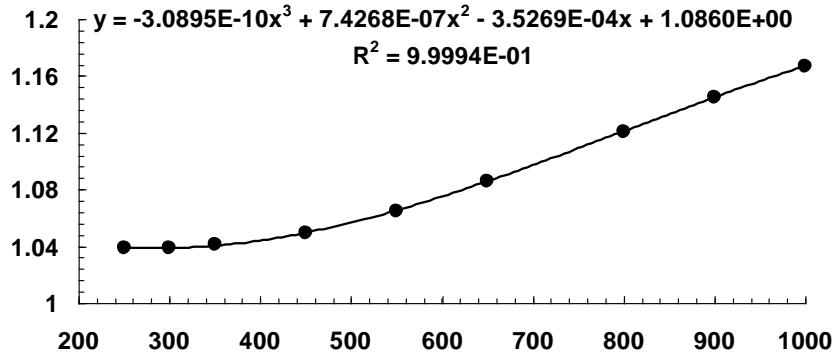
Carbon dioxide: Third-order polynomial regression provides a good fit to the data,



Oxygen: Fifth-order polynomial regression provides a good fit to the data,



Nitrogen: Third-order polynomial regression provides a good fit to the data,



20.56 This part of the problem is well-suited for Newton interpolation. First, order the points so that they are as close to and as centered about the unknown as possible, for $x = 4$.

$$\begin{array}{ll} y_0 = 4 & f(y_0) = 38.43 \\ y_1 = 2 & f(y_1) = 53.5 \\ y_2 = 6 & f(y_2) = 30.39 \\ y_3 = 0 & f(y_3) = 80 \\ y_4 = 8 & f(y_4) = 30 \end{array}$$

The results of applying Newton's polynomial at $y = 3.2$ are

Order	f(y)	Error
0	38.43	6.028
1	44.458	-0.8436
2	43.6144	-0.2464
3	43.368	0.112448
4	43.48045	

The minimum error occurs for the third-order version so we conclude that the interpolation is 43.368.

(b) This is an example of two-dimensional interpolation. One way to approach it is to use cubic interpolation along the y dimension for values at specific values of x that bracket the unknown. For example, we can utilize the following points at $x = 2$.

$$\begin{array}{ll} y_0 = 0 & f(y_0) = 90 \\ y_1 = 2 & f(y_1) = 64.49 \\ y_2 = 4 & f(y_2) = 48.9 \\ y_3 = 6 & f(y_3) = 38.78 \end{array}$$

$$T(x = 2, y = 2.7) = 58.13288438$$

All the values can be tabulated as

$$\begin{aligned} T(x = 2, y = 2.7) &= 58.13288438 \\ T(x = 4, y = 2.7) &= 47.1505625 \end{aligned}$$

$$\begin{aligned}T(x = 6, y = 2.7) &= 42.74770188 \\T(x = 8, y = 2.7) &= 46.5\end{aligned}$$

These values can then be used to interpolate at $x = 4.3$ to yield

$$T(x = 4.3, y = 2.7) = 46.03218664$$

Note that some software packages allow you to perform such multi-dimensional interpolations very efficiently. For example, MATLAB has a function interp2 that provides numerous options for how the interpolation is implemented. Here is an example of how it can be implemented using linear interpolation,

```
>> Z=[100 90 80 70 60;
85 64.49 53.5 48.15 50;
70 48.9 38.43 35.03 40;
55 38.78 30.39 27.07 30;
40 35 30 25 20];
>> X=[0 2 4 6 8];
>> Y=[0 2 4 6 8];
>> T=interp2(X,Y,Z,4.3,2.7)

T =
47.5254
```

It can also perform the same interpolation but using bicubic interpolation,

```
>> T=interp2(X,Y,Z,4.3,2.7,'cubic')

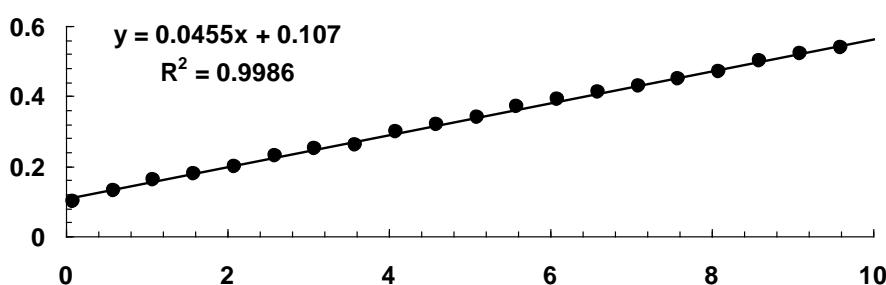
T =
46.0062
```

Finally, the interpolation can be implemented using splines,

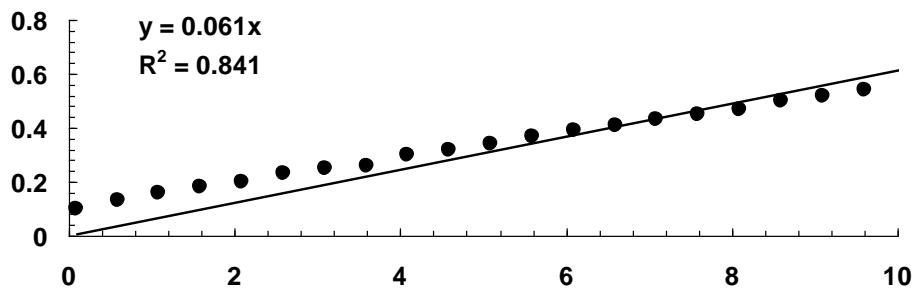
```
>> T=interp2(X,Y,Z,4.3,2.7,'spline')

T =
46.1507
```

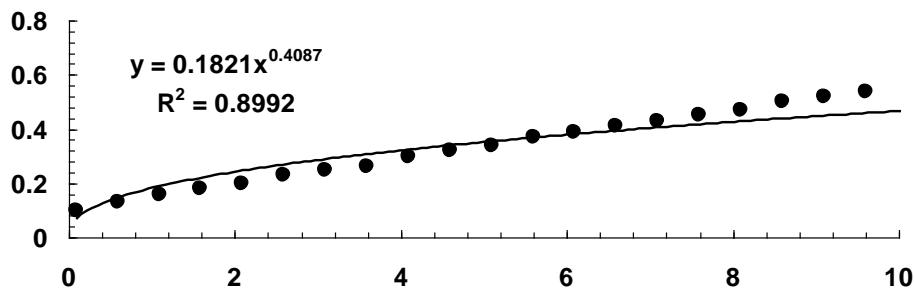
20.57 This problem was solved using an Excel spreadsheet and TrendLine. Linear regression gives



Forcing a zero intercept yields



One alternative that would force a zero intercept is a power fit



However, this seems to represent a poor compromise since it misses the linear trend in the data. An alternative approach would be to assume that the physically-unrealistic non-zero intercept is an artifact of the measurement method. Therefore, if the linear slope is valid, we might try $y = 0.0455x$.

CHAPTER 21

21.1 (a) Analytical solution:

$$\int_0^4 (1 - e^{-2x}) dx = \left[x + 0.5e^{-2x} \right]_0^4 = 4 + 0.5e^{-2(4)} - 0 - 0.5e^{-2(0)} = 3.500168$$

(b) Trapezoidal rule ($n = 1$):

$$I = (4 - 0) \frac{0 + 0.999665}{2} = 1.999329$$

$$\varepsilon_t = \left| \frac{3.500168 - 1.999329}{3.500168} \right| \times 100\% = 42.88\%$$

(c) Trapezoidal rule ($n = 2$):

$$I = (4 - 0) \frac{0 + 2(0.981684) + 0.999665}{4} = 2.963033 \quad \varepsilon_t = 15.35\%$$

Trapezoidal rule ($n = 4$):

$$I = (4 - 0) \frac{0 + 2(0.864665 + 0.981684 + 0.997521) + 0.999665}{8} = 3.343703 \quad \varepsilon_t = 4.47\%$$

(d) Simpson's 1/3 rule:

$$I = (4 - 0) \frac{0 + 4(0.981684) + 0.999665}{6} = 3.284268 \quad \varepsilon_t = 6.17\%$$

(e) Simpson's rule ($n = 4$):

$$I = (4 - 0) \frac{0 + 4(0.864665 + 0.997521) + 2(0.981684) + 0.999665}{12} = 3.470592 \quad \varepsilon_t = 0.84\%$$

(f) Simpson's 3/8 rule:

$$I = (4 - 0) \frac{0 + 3(0.930517 + 0.995172) + 0.999665}{8} = 3.388365 \quad \varepsilon_t = 3.19\%$$

(g) Simpson's rules ($n = 5$):

$$\begin{aligned}
I &= (1.6 - 0) \frac{0 + 4(0.798103) + 0.959238}{6} \\
&\quad + (4 - 1.6) \frac{0.959238 + 3(0.99177 + 0.998338) + 0.999665}{8} \\
&= 1.107107 + 2.378769 = 3.485876 \quad \varepsilon_t = 0.41\%
\end{aligned}$$

21.2 (a) Analytical solution:

$$\int_0^{\pi/2} (6 + 3\cos x) dx = [6x + 3\sin x]_0^{\pi/2} = 6(\pi/2) + 3\sin(\pi/2) - 0 = 12.42478$$

(b) Trapezoidal rule ($n = 1$):

$$I = (1.570796 - 0) \frac{9 + 6}{2} = 11.78097 \quad \varepsilon_t = \left| \frac{12.42478 - 11.78097}{12.42478} \right| \times 100\% = 5.182\%$$

(c) Trapezoidal rule ($n = 2$):

$$I = (1.570796 - 0) \frac{9 + 2(8.12132) + 6}{4} = 12.26896 \quad \varepsilon_t = 1.254\%$$

Trapezoidal rule ($n = 4$):

$$I = (1.570796 - 0) \frac{9 + 2(8.771639 + 8.12132 + 7.14805) + 6}{8} = 12.38613 \quad \varepsilon_t = 0.311\%$$

(d) Simpson's 1/3 rule:

$$I = (1.570796 - 0) \frac{9 + 4(8.12132) + 6}{6} = 12.43162 \quad \varepsilon_t = 0.055\%$$

(e) Simpson's rule ($n = 4$):

$$I = (1.570796 - 0) \frac{9 + 4(8.771639 + 7.14805) + 2(8.12132) + 6}{12} = 12.42518 \quad \varepsilon_t = 0.0032\%$$

(f) Simpson's 3/8 rule:

$$I = (1.570796 - 0) \frac{9 + 3(8.598076 + 7.5) + 6}{8} = 12.42779 \quad \varepsilon_t = 0.024\%$$

(g) Simpson's rules ($n = 5$):

$$\begin{aligned}
I &= (0.628319 - 0) \frac{9 + 4(8.85317) + 8.427051}{6} \\
&\quad + (1.570796 - 0.628319) \frac{8.427051 + 3(7.763356 + 6.927051) + 6}{8} \\
&= 5.533364 + 6.891665 = 12.42503 \quad \varepsilon_t = 0.002\%
\end{aligned}$$

21.3 (a) Analytical solution:

$$\int_{-2}^4 (1 - x - 4x^3 + 2x^5) dx = \left[x - \frac{x^2}{2} - x^4 + \frac{x^6}{3} \right]_{-2}^4 = 1104$$

(b) Trapezoidal rule ($n = 1$):

$$I = (4 - (-2)) \frac{-29 + 1789}{2} = 5280 \quad \varepsilon_t = \left| \frac{1104 - 5280}{1104} \right| \times 100\% = 378.26\%$$

(c) Trapezoidal rule ($n = 2$):

$$I = (4 - (-2)) \frac{-29 + 2(-2) + 1789}{4} = 2634 \quad \varepsilon_t = 138.59\%$$

Trapezoidal rule ($n = 4$):

$$I = (4 - (-2)) \frac{-29 + 2(1.9375 - 2 + 131.3125) + 1789}{8} = 1516.875 \quad \varepsilon_t = 37.398\%$$

(d) Simpson's 1/3 rule ($n = 2$):

$$I = (4 - (-2)) \frac{-29 + 4(-2) + 1789}{6} = 1752 \quad \varepsilon_t = 58.7\%$$

(e) Simpson's 3/8 rule:

$$I = (4 - (-2)) \frac{-29 + 3(1 + 31) + 1789}{8} = 1392 \quad \varepsilon_t = 26.087\%$$

(f) Boole's rule ($n = 5$):

$$I = (4 - (-2)) \frac{7(-29) + 32(1.9375) + 12(-2) + 32(131.3125) + 7(1789)}{90} = 1104 \quad \varepsilon_t = 0\%$$

21.4 Analytical solution:

$$\int_1^2 (x + 2/x)^2 dx = \left[\frac{x^3}{3} + 4x - \frac{4}{x} \right]_1^2 = 8.33333$$

Trapezoidal rule ($n = 1$):

$$(2-1) \frac{9+9}{2} = 9$$

The results are summarized below:

<i>n</i>	Integral	ε_t
1	9	8%
2	8.513889	2.167%
3	8.415185	0.982%
4	8.379725	0.557%

21.5 Analytical solution:

$$\int_{-3}^5 (4x-3)^3 dx = \left[\frac{1}{16} (4x-3)^4 \right]_{-3}^5 = 2056$$

Simpson's rule ($n = 4$):

$$I = (5 - (-3)) \frac{-3375 + 4(-343 + 729) + 2(1) + 4913}{12} = 2056 \quad \varepsilon_t = 0\%$$

Simpson's rules ($n = 5$):

$$I = (0.2 - (-3)) \frac{-3375 + 4(-636.056) - 10.648}{6} + (5 - 0.2) \frac{-10.648 + 3(74.088 + 1191.016) + 4913}{8} = -3162.6 + 5218.598 = 2056 \quad \varepsilon_t = 0\%$$

Because Simpson's rules are perfect for cubics, both versions yield the exact result for this cubic polynomial.

21.6 Analytical solution:

$$\int_0^3 x^2 e^x dx = [(x^2 - 2x + 2)e^x]_0^3 = 98.42768$$

Trapezoidal rule ($n = 4$):

$$I = (3 - 0) \frac{0 + 2(1.190813 + 10.0838 + 48.03166) + 180.7698}{8} = 112.2684 \quad \varepsilon_t = 14.062\%$$

Simpson's rule ($n = 4$):

$$I = (3 - 0) \frac{0 + 4(1.190813 + 48.03166) + 2(10.0838) + 180.7698}{12} = 99.45683 \quad \varepsilon_t = 1.046\%$$

21.7 Analytical solution:

$$\int_{0.5}^{1.5} 14^{2x} dx = \left[\frac{1}{2 \ln 14} 14^{2x} \right]_{0.5}^{1.5} = 517.2301$$

(a) Trapezoidal rule ($n = 1$):

$$I = (1.5 - 0.5) \frac{14 + 2744}{2} = 1379 \quad \varepsilon_t = \left| \frac{517.2301 - 1379}{517.2301} \right| \times 100\% = 166.612\%$$

(b) Simpson's 1/3 rule ($n = 2$):

$$I = (1.5 - 0.5) \frac{14 + 4(196) + 2744}{6} = 590.3333 \quad \varepsilon_t = 14.134\%$$

(c) Simpson's 3/8 rule:

$$I = (1.5 - 0.5) \frac{14 + 3(81.323 + 472.3879) + 2744}{8} = 552.3916 \quad \varepsilon_t = 6.798\%$$

(d) Boole's rule:

$$I = (1.5 - 0.5) \frac{7(14) + 32(52.3832) + 12(196) + 32(733.3648) + 7(2744)}{90} = 520.0215 \quad \varepsilon_t = 0.5397\%$$

(e) Midpoint method:

$$I = (1.5 - 0.5)196 = 196 \quad \varepsilon_t = 62.106\%$$

(f) 3-segment-2-point open integration formula:

$$I = (1.5 - 0.5) \frac{81.323 + 472.3879}{2} = 276.8554 \quad \varepsilon_t = 46.473\%$$

(g) 4-segment-3-point open integration formula:

$$I = (1.5 - 0.5) \frac{2(52.3832) - 196 + 2(733.3648)}{3} = 458.4987 \quad \varepsilon_t = 11.355\%$$

21.8 Analytical solution:

$$\int_0^3 (5 + 3\cos x) dx = [5x + 3\sin x]_0^3 = 15.42336$$

(a) Trapezoidal rule ($n = 1$):

$$I = (3 - 0) \frac{8 + 2.030023}{2} = 15.04503 \quad \varepsilon_t = \left| \frac{15.42336 - 15.04503}{15.42336} \right| \times 100\% = 2.453\%$$

(b) Simpson's 1/3 rule ($n = 2$):

$$I = (3 - 0) \frac{8 + 4(5.212212) + 2.030023}{6} = 15.43943 \quad \varepsilon_t = 0.104\%$$

(c) Simpson's 3/8 rule:

$$I = (3 - 0) \frac{8 + 3(6.620907 + 3.751559) + 2.030023}{8} = 15.43028 \quad \varepsilon_t = 0.045\%$$

(d) Simpson's rules ($n = 5$):

$$I = (1.2 - 0) \frac{8 + 4(7.476007) + 6.087073}{6} \\ + (3 - 1.2) \frac{6.087073 + 3(4.318394 + 2.787819) + 2.030023}{8} \\ = 8.79822 + 6.62304 = 15.42126 \quad \varepsilon_t = 0.014\%$$

(e) Boole's rule:

$$I = (3 - 0) \frac{7(8) + 32(7.195067) + 12(5.212212) + 32(3.115479) + 7(2.030023)}{90} = 15.42314 \\ \varepsilon_t = 0.0014\%$$

(f) Midpoint method:

$$I = (3 - 0)5.212212 = 15.63663 \quad \varepsilon_t = 1.383\%$$

(g) 3-segment-2-point open integration formula:

$$I = (3 - 0) \frac{6.620907 + 3.751559}{2} = 15.5587 \quad \varepsilon_t = 0.877\%$$

(h) 4-segment-3-point open integration formula:

$$I = (3 - 0) \frac{2(7.195067) - 5.212212 + 2(3.115479)}{3} = 15.40888 \quad \varepsilon_t = 0.094\%$$

21.9 Analytical solution:

$$z(t) = \int_0^t \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) dt = \left[\frac{m}{c_d} \ln\left[\cosh\left(\sqrt{\frac{gc_d}{m}} t\right)\right] \right]_0^t$$

$$z(10) = \left[\frac{68.1}{0.25} \ln\left[\cosh\left(\sqrt{\frac{9.8(0.25)}{68.1}}(10)\right)\right] \right]_0^{10} = 333.9262$$

Thus, the result to 3 significant digits is 334. Here are results for various multiple-segment trapezoidal rules:

<i>n</i>	<i>I</i>
1	246.9593
2	314.4026
3	325.4835
4	329.216
5	330.9225
6	331.8443
7	332.3984
8	332.7573
9	333.0031
10	333.1788
11	333.3087
12	333.4074
13	333.4842
14	333.5452

Thus, a 14-segment application gives the result to 4 significant digits.

21.10

(a) Trapezoidal rule (*n* = 5):

$$I = (0.5 - 0) \frac{1 + 2(8 + 4 + 3.5 + 5) + 1}{10} = 2.15$$

(b) Simpson's rules (*n* = 5):

$$I = (0.2 - 0) \frac{1 + 4(8) + 4}{6} + (0.5 - 0.2) \frac{4 + 3(3.5 + 5) + 1}{8} = 1.233333 + 1.14375 = 2.377083$$

21.11

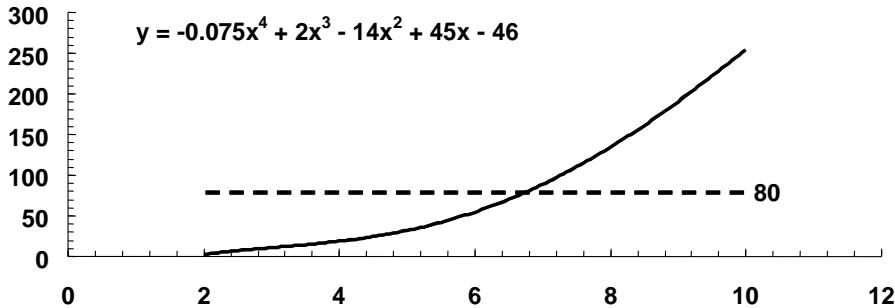
(a) Trapezoidal rule (*n* = 6):

$$I = (10 - (-2)) \frac{35 + 2(5 - 10 + 2 + 5 + 3) + 20}{12} = 65$$

(b) Simpson's rules ($n = 6$):

$$I = (10 - (-2)) \frac{35 + 4(5 + 2 + 3) + 2(-10 + 5) + 20}{18} = 56.66667$$

21.12 (a) A graph suggests that the mean value is about 80.



(b) Analytical solution:

$$\begin{aligned} & \frac{\int_2^{10} -46 + 45x - 14x^2 + 2x^3 - 0.075x^4 dx}{10 - 2} \\ &= \frac{[-46x + 22.5x^2 - 4.6667x^3 + 0.5x^4 - 0.015x^5]_2^{10}}{10 - 2} = \frac{655.1467}{8} = 81.89333 \end{aligned}$$

(c) Numerical solution:

$$\begin{aligned} I &= (5.2 - 2) \frac{2.8 + 4(15.27488) + 35.81888}{6} \\ &\quad + (10 - 5.2) \frac{35.81888 + 3(81.14368 + 156.1645) + 254}{8} \\ &= 53.18315 + 601.046 = 654.2292 \end{aligned}$$

$$\text{Average} = \frac{654.2292}{10 - 2} = 81.77865$$

$$\varepsilon_t = \left| \frac{81.89333 - 81.77865}{81.89333} \right| \times 100\% = 0.14\%$$

21.13 (a) Analytical solution:

$$\int_0^{0.6} 2e^{-1.5x} dx = \left[-1.33333e^{-1.5x} \right]_0^{0.6} = -0.54209 - (-1.33333) = 0.79124$$

(b) Trapezoidal rule

$$\begin{aligned} I &= (0.05 - 0) \frac{2 + 1.8555}{2} + (0.15 - 0.05) \frac{1.8555 + 1.597}{2} + (0.25 - 0.15) \frac{1.597 + 1.3746}{2} \\ &\quad + (0.35 - 0.25) \frac{1.3746 + 1.1831}{2} + (0.475 - 0.35) \frac{1.1831 + 0.9808}{2} \\ &\quad + (0.6 - 0.475) \frac{0.9808 + 0.8131}{2} = 0.79284 \end{aligned}$$

$$\varepsilon_t = \left| \frac{0.79124 - 0.79284}{0.79124} \right| \times 100\% = 0.2022\%$$

(c) Trapezoidal/Simpson's rules

$$\begin{aligned} I &= (0.05 - 0) \frac{2 + 1.8555}{2} + (0.35 - 0.05) \frac{1.8555 + 3(1.597 + 1.3746) + 1.1831}{8} \\ &\quad + (0.6 - 0.35) \frac{1.1831 + 4(0.9808) + 0.8131}{6} = 0.791282 \end{aligned}$$

$$\varepsilon_t = \left| \frac{0.79124 - 0.791282}{0.79124} \right| \times 100\% = 0.0052\%$$

21.14 (a) Analytical solution:

$$\begin{aligned} \int_{-1}^1 \int_0^2 (x^2 - 2y^2 + xy^3) dx dy &= \int_{-1}^1 \left[\frac{x^3}{3} - 2y^2 x + y^3 \frac{x^2}{2} \right]_0^2 dy \\ &= \int_{-1}^1 \frac{8}{3} - 4y^2 + 2y^3 dy = \left[\frac{8}{3}y - \frac{4}{3}y^3 + \frac{1}{2}y^4 \right]_{-1}^1 = 2.666667 \end{aligned}$$

(b) Sweeping across the x dimension:

$y = -1$:

$$I = (2 - 0) \frac{-2 + 2(-2) + 0}{4} = -3$$

$y = 0$:

$$I = (2 - 0) \frac{0 + 2(1) + 4}{4} = 3$$

$y = 1$:

$$I = (2 - 0) \frac{-2 + 2(0) + 4}{4} = 1$$

These values can then be integrated along the y dimension:

$$I = (1 - (-1)) \frac{-3 + 2(3) + 1}{4} = 2$$

$$\varepsilon_t = \left| \frac{2.666667 - 2}{2.666667} \right| \times 100\% = 25\%$$

(c) Sweeping across the x dimension:

$y = -1$:

$$I = (2 - 0) \frac{-2 + 4(-2) + 0}{6} = -3.33333$$

$y = 0$:

$$I = (2 - 0) \frac{0 + 4(1) + 4}{6} = 2.666667$$

$y = 1$:

$$I = (2 - 0) \frac{-2 + 4(0) + 4}{6} = 0.666667$$

These values can then be integrated along the y dimension:

$$I = (1 - (-1)) \frac{-3.33333 + 4(2.666667) + 0.666667}{6} = 2.666667 \quad \varepsilon_t = 0\%$$

21.15 (a) Analytical solution:

$$\begin{aligned} \int_{-2}^2 \int_0^2 \int_{-3}^1 (x^3 - 3yz) \, dx \, dy \, dz &= \int_{-2}^2 \int_0^2 \left[\frac{x^4}{4} - 3yzx \right]_{-3}^1 \, dy \, dz \\ &= \int_{-2}^2 \int_0^2 -20 - 12yz \, dy \, dz = \int_{-2}^2 \left[-20y - 6y^2 z \right]_0^2 \, dz = \int_{-2}^2 -40 - 24z \, dz \\ &= \left[-40z - 12z^2 \right]_{-2}^2 = -160 \end{aligned}$$

(b) For $z = -2$, sweeping across the x dimension:

$z = -2, y = 0$:

$$I = (1 - (-3)) \frac{-27 + 4(-1) + 1}{6} = -20$$

$z = -2, y = 1$:

$$I = (1 - (-3)) \frac{-21 + 4(5) + 7}{6} = 4$$

$z = -2; y = 2:$

$$I = (1 - (-3)) \frac{-15 + 4(11) + 13}{6} = 28$$

These values can then be integrated along the y dimension:

$$I = (2 - 0) \frac{-20 + 4(4) + 28}{6} = 8$$

For $z = 0$, similar calculations yield

$z = 0; y = 0: I = -20$

$z = 0; y = 1: I = -20$

$z = 0; y = 2: I = -20$

These values can then be integrated along the y dimension:

$$I = (2 - 0) \frac{-20 + 4(-20) - 20}{6} = -40$$

For $z = 2$, similar calculations yield

$z = 2; y = 0: I = -20$

$z = 2; y = 1: I = -44$

$z = 2; y = 2: I = -68$

These values can then be integrated along the y dimension:

$$I = (2 - 0) \frac{-20 + 4(-44) - 68}{6} = -88$$

Finally, these results can be integrated along the z dimension,

$$I = (2 - (-2)) \frac{8 + 4(-40) - 88}{6} = -160 \quad \varepsilon_t = 0\%$$

21.16 Here is a VBA program that is set up to duplicate the computation performed in Example 21.2.

```
Option Explicit

Sub TestTrapm()
    Dim n As Integer, i As Integer
    Dim a As Double, b As Double, h As Double
    Dim x(100) As Double, f(100) As Double
    'Enter data and integration parameters
    a = 0
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

b = 0.8
n = 2
h = (b - a) / n
'generate data from function
x(0) = a
f(0) = fx(a)
For i = 1 To n
    x(i) = x(i - 1) + h
    f(i) = fx(x(i))
Next i
'invoke function to determine and display integral
MsgBox "The integral is " & Trapm(h, n, f)
End Sub

Function Trapm(h, n, f)
Dim i As Integer, sum As Double
sum = f(0)
For i = 1 To n - 1
    sum = sum + 2 * f(i)
Next i
sum = sum + f(n)
Trapm = h * sum / 2
End Function

Function fx(x)
fx = 0.2 + 25 * x - 200 * x ^ 2 + 675 * x ^ 3 - 900 * x ^ 4 + 400 * x ^ 5
End Function

```

21.17 Here is a VBA program that is set up to duplicate the computation performed in Example 21.5.

```

Option Explicit

Sub TestSimp()
Dim n As Integer, i As Integer
Dim a As Double, b As Double, h As Double
Dim x(100) As Double, f(100) As Double
'Enter data and integration parameters
a = 0
b = 0.8
n = 4
h = (b - a) / n
'generate data from function
x(0) = a
f(0) = fx(a)
For i = 1 To n
    x(i) = x(i - 1) + h
    f(i) = fx(x(i))
Next i
'invoke function to determine and display integral
MsgBox "The integral is " & Simp13(h, n, f)
End Sub

Function Simp13(h, n, f)
Dim i As Integer
Dim sum As Double
sum = f(0)
For i = 1 To n - 2 Step 2
    sum = sum + 4 * f(i) + 2 * f(i + 1)
Next i
sum = sum + 4 * f(n - 1) + f(n)
End Function

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

Simp13m = h * sum / 3
End Function

Function fx(x)
fx = 0.2 + 25 * x - 200 * x ^ 2 + 675 * x ^ 3 - 900 * x ^ 4 + 400 * x ^ 5
End Function

```

21.18 Here is a VBA program that is set up to duplicate the computation performed in Example 21.8.

```

Option Explicit

Sub TestUneven()
Dim n As Integer, i As Integer
Dim label As String
Dim a As Double, b As Double, h As Double
Dim x(100) As Double, f(100) As Double
'Enter data
Sheets("Sheet1").Select
Range("a6").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
'Input data from sheet
Range("a6").Select
For i = 0 To n
    x(i) = ActiveCell.Value
    ActiveCell.Offset(0, 1).Select
    f(i) = ActiveCell.Value
    ActiveCell.Offset(1, -1).Select
Next i
'invoke function to determine and display integral
MsgBox "The integral is " & Uneven(n, x, f)
End Sub

Function Uneven(n, x, f)
Dim k As Integer, j As Integer
Dim h As Double, sum As Double, hf As Double
h = x(1) - x(0)
k = 1
sum = 0#
For j = 1 To n
    hf = x(j + 1) - x(j)
    If Abs(h - hf) < 0.000001 Then
        If k = 3 Then
            sum = sum + Simp13(h, f(j - 3), f(j - 2), f(j - 1))
            k = k - 1
        Else
            k = k + 1
        End If
    Else
        If k = 1 Then
            sum = sum + Trap(h, f(j - 1), f(j))
        Else
            If k = 2 Then
                sum = sum + Simp13(h, f(j - 2), f(j - 1), f(j))
            Else
                sum = sum + Simp38(h, f(j - 3), f(j - 2), f(j - 1), f(j))
            End If
            k = 1
        End If
    End If
Next j
End Function

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

End If
h = hf
Next j
Uneven = sum
End Function

Function Trap(h, f0, f1)
Trap = h * (f0 + f1) / 2
End Function

Function Simp13(h, f0, f1, f2)
Simp13 = 2 * h * (f0 + 4 * f1 + f2) / 6
End Function

Function Simp38(h, f0, f1, f2, f3)
Simp38 = 3 * h * (f0 + 3 * (f1 + f2) + f3) / 8
End Function

Function fx(x)
fx = 0.2 + 25 * x - 200 * x ^ 2 + 675 * x ^ 3 - 900 * x ^ 4 + 400 * x ^ 5
End Function

```

To run the program, the data is entered in columns A and B on a worksheet labeled Sheet1. After the macro is run, a message box displays the integral estimate as shown.

A	B	C	D	E	F	G
1						
2		RUN				
3						
4						
5	x	f(x)				
6	0	0.20000				
7	0.12	1.30973				
8	0.22	1.30524				
9	0.32	1.74339				
10	0.36	2.07490				
11	0.4	2.45600				
12	0.44	2.84298				
13	0.54	3.50730				
14	0.64	3.18193				
15	0.7	2.36300				
16	0.8	0.23200				



21.19 (a) Analytical solution:

$$M = \int_0^{11} 5 + 0.25x^2 \, dx = \left[5x + 0.083333x^3 \right]_0^{11} = 165.9167$$

(b) Trapezoidal rule:

$$I = (1-0) \frac{5+5.25}{2} + (2-1) \frac{5.25+6}{2} + \dots = 166.375$$

(c) Simpson's rule:

$$I = (2-0) \frac{5+4(5.25)+6}{6} + (4-2) \frac{6+4(7.25)+9}{6} + \dots = 165.9167$$

21.20 Trapezoidal/Simpsons rules

$$\begin{aligned}
 I &= (2 - 0.5) \frac{336 + 294.4}{2} + (4 - 2) \frac{294.4 + 4(266.4) + 260.8}{6} \\
 &\quad + (10 - 4) \frac{260.8 + 3(260.5 + 249.6) + 193.6}{8} + (11 - 10) \frac{193.6 + 165.6}{2} = 2681.192
 \end{aligned}$$

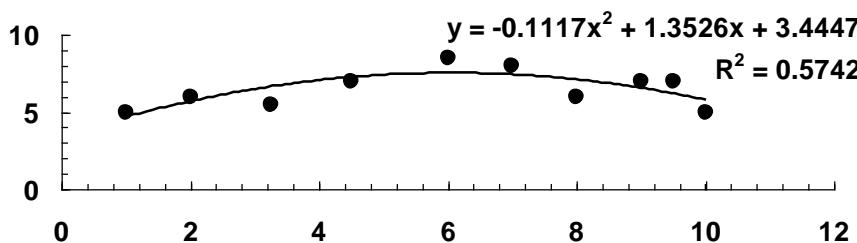
21.21 (a) Trapezoidal rule

$$I = (2 - 1) \frac{5 + 6}{2} + (3.25 - 2) \frac{6 + 5.5}{2} + \dots = 60.375 \frac{\text{m} \cdot \text{min}}{\text{s}} \times \frac{60 \text{ s}}{\text{min}} = 3,622.5 \text{ m}$$

(b) Trapezoidal/Simpsons rules

$$\begin{aligned}
 I &= (2 - 1) \frac{5 + 6}{2} + (4.5 - 2) \frac{6 + 4(5.5) + 7}{6} + (6 - 4.5) \frac{7 + 8.5}{2} \\
 &\quad + (9 - 6) \frac{8.5 + 3(8 + 6) + 7}{8} + (10 - 9) \frac{7 + 4(7) + 5}{6} = 59.9375 \frac{\text{m} \cdot \text{min}}{\text{s}} \times \frac{60 \text{ s}}{\text{min}} = 3,596.25 \text{ m}
 \end{aligned}$$

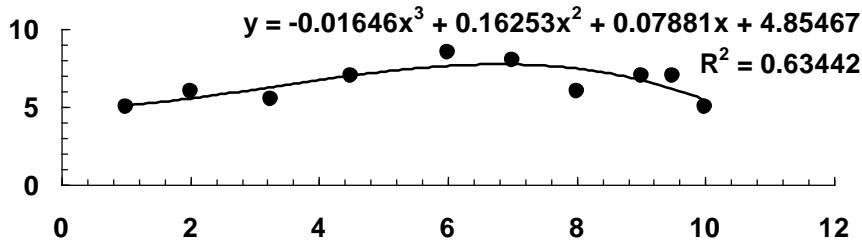
(c) We can use regression to fit a quadratic equation to the data



This equation can be integrated to yield

$$\begin{aligned}
 M &= \int_1^{10} -0.1117x^2 + 1.3526x + 3.4447 \, dx = \left[-0.03723x^3 + 0.6763x^2 + 3.4447x \right]_1^{10} \\
 &= 60.7599 \frac{\text{m} \cdot \text{min}}{\text{s}} \times \frac{60 \text{ s}}{\text{min}} = 3,645.594 \text{ m}
 \end{aligned}$$

We can use regression to fit a cubic equation to the data



This equation can be integrated to yield

$$\begin{aligned}
 M &= \int_1^{10} -0.01646x^3 + 0.16253x^2 + 0.07881x + 4.85467 \, dx \\
 &= \left[-0.00412x^4 + 0.054177x^3 + 0.039405x^2 + 4.85467x \right]_1^{10} \\
 &= 60.56973 \frac{\text{m} \cdot \text{min}}{\text{s}} \times \frac{60 \text{ s}}{\text{min}} = 3,634.184 \text{ m}
 \end{aligned}$$

21.22 We can set up a table that contains the values comprising the integrand

x, cm	ρ , g/cm ³	A_c , cm ²	$\rho \times A_c$, g/cm
0	4	100	400
200	3.95	103	406.85
300	3.89	106	412.34
400	3.8	110	418
600	3.6	120	432
800	3.41	133	453.53
1000	3.3	150	495

We can integrate this data using a combination of the trapezoidal and Simpson's rules,

$$\begin{aligned}
 I &= (200 - 0) \frac{400 + 406.85}{2} + (400 - 200) \frac{406.85 + 4(412.34) + 418}{6} \\
 &\quad + (1000 - 400) \frac{418 + 3(432 + 453.53) + 495}{8} = 430,877.9 \text{ g} = 430.8779 \text{ kg}
 \end{aligned}$$

21.23 We can set up a table that contains the values comprising the integrand

t, hr	t, d	rate (cars/4 min)	rate (cars/d)
7:30	0.312500	18	6480
7:45	0.322917	24	8640
8:00	0.333333	14	5040
8:15	0.343750	24	8640
8:45	0.364583	21	7560
9:15	0.385417	9	3240

We can integrate this data using a combination of Simpson's 3/8 and 1/3 rules. This yields the number of cars that go through the intersection between 7:30 and 9:15,

$$\begin{aligned} I &= (0.34375 - 0.3125) \frac{6480 + 3(8640 + 5040) + 8640}{8} \\ &\quad + (0.385417 - 0.34375) \frac{8640 + 4(7560) + 3240}{6} \\ &= 219.375 + 292.5 = 511.875 \text{ cars} \end{aligned}$$

The number of cars going through the intersection per minute can be computed as

$$\frac{511.875 \text{ cars}}{1.75 \text{ hr}} \frac{\text{hr}}{60 \text{ min}} = 4.875 \frac{\text{cars}}{\text{min}}$$

21.24 We can use Simpson's 1/3 rule to integrate across the y dimension,

$x = 0$:

$$I = (4 - 0) \frac{-2 + 4(-4) - 8}{6} = -17.3333$$

$x = 4$:

$$I = (4 - 0) \frac{-1 + 4(-3) - 8}{6} = -14$$

$x = 8$:

$$I = (4 - 0) \frac{4 + 4(1) - 6}{6} = 1.3333$$

$x = 12$:

$$I = (4 - 0) \frac{10 + 4(7) + 4}{6} = 28$$

These values can then be integrated along the x dimension with Simpson's 3/8 rule:

$$I = (12 - 0) \frac{-17.3333 + 3(-14 + 1.3333) + 28}{8} = -41$$

CHAPTER 22

22.1 Analytical solution:

$$I = \int_1^2 \left(2x + \frac{3}{x} \right)^2 dx = \left[\frac{4}{3}x^3 + 12x - \frac{9}{x} \right]_1^2 = 25.8333333$$

The first iteration involves computing 1 and 2 segment trapezoidal rules and combining them as

$$I = \frac{4(26.3125) - 27.625}{3} = 25.875$$

and computing the approximate error as

$$\varepsilon_a = \left| \frac{25.875 - 26.3125}{25.875} \right| \times 100\% = 1.6908\%$$

The computation can be continues as in the following tableau until $\varepsilon_a < 0.5\%$.

	1	2	3
<i>n</i>	$\varepsilon_a \rightarrow$	1.6908%	0.0098%
1	27.62500000	25.87500000	25.83456463
2	26.31250000	25.83709184	
4	25.95594388		

The true error of the final result can be computed as

$$\varepsilon_t = \left| \frac{25.8333333 - 25.83456463}{25.8333333} \right| \times 100\% = 0.0048\%$$

22.2 Analytical solution:

$$I = \int_0^3 xe^x dx = [xe^x - e^x]_0^3 = 41.17107385$$

	1	2	3	4
<i>n</i>	$\varepsilon_t \rightarrow$	5.8349%	0.1020%	0.0004%
	$\varepsilon_a \rightarrow$	26.8579%	0.3579%	0.0015862%
1	90.38491615	43.57337260	41.21305531	41.17125852
2	55.27625849	41.36057514	41.17191160	
4	44.83949598	41.18370307		
8	42.09765130			

22.3

n	1 $\varepsilon_a \rightarrow$	2 7.9715%	3 0.0997%
1	1.34376994	1.97282684	1.94183605
2	1.81556261	1.94377297	
4	1.91172038		

22.4 Change of variable:

$$x = \frac{2+1}{2} + \frac{2-1}{2} x_d = 1.5 + 0.5x_d$$

$$dx = \frac{2-1}{2} dx_d = 0.5dx_d$$

$$I = \int_{-1}^1 \left(3 + x_d + \frac{3}{1.5 + 0.5x_d} \right)^2 0.5dx_d$$

Therefore, the transformed function is

$$f(x_d) = 0.5 \left(3 + x_d + \frac{3}{1.5 + 0.5x_d} \right)^2$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 12.00146 + 13.80525 = 25.8067$$

$$\varepsilon_t = \left| \frac{25.833333 - 25.8067}{25.833333} \right| \times 100\% = 0.103\%$$

Three-point formula:

$$\begin{aligned} I &= 0.5555556 f(-0.7745967) + 0.8888889 f(0) + 0.5555556 f(0.7745967) \\ &= 0.5555556(12.1108) + 0.8888889(12.5) + 0.5555556(14.38716) \\ &= 6.72822 + 11.11111 + 7.992868 = 25.8322 \end{aligned}$$

$$\varepsilon_t = \left| \frac{25.833333 - 25.8322}{25.833333} \right| \times 100\% = 0.0044\%$$

Four-point formula:

$$\begin{aligned}
I &= 0.3478548 f(-0.861136312) + 0.6521452 f(-0.339981044) + 0.6521452 f(0.339981044) \\
&\quad + 0.3478548 f(0.861136312) \\
&= 0.3478548(12.22202) + 0.6521452(12.08177) + 0.6521452(13.19129) + 0.3478548(14.66156) \\
&= 4.251489 + 7.879067 + 8.602639 + 5.100095 = 25.83329
\end{aligned}$$

$$\varepsilon_t = \left| \frac{25.833333 - 25.83329}{25.833333} \right| \times 100\% = 0.000169\%$$

22.5 Change of variable:

$$x = \frac{3+0}{2} + \frac{3-0}{2} x_d = 1.5 + 1.5x_d$$

$$dx = \frac{3-0}{2} dx_d = 1.5 dx_d$$

$$I = \int_{-1}^1 (1.5 + 1.5x_d) e^{1.5+1.5x_d} 1.5 dx_d$$

Therefore, the transformed function is

$$f(x_d) = (2.25 + 2.25x_d) e^{1.5+1.5x_d}$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 1.792647 + 37.81485 = 39.6075$$

$$\varepsilon_t = \left| \frac{41.17107 - 39.6075}{41.17107} \right| \times 100\% = 3.7977\%$$

Three-point formula:

$$\begin{aligned}
I &= 0.5555556 f(-0.7745967) + 0.8888889 f(0) + 0.5555556 f(0.7745967) \\
&= 0.5555556(0.711181) + 0.8888889(10.0838) + 0.5555556(57.19111) \\
&= 0.3951 + 8.963378 + 31.77284 = 41.13131
\end{aligned}$$

$$\varepsilon_t = \left| \frac{41.17107 - 41.13131}{41.17107} \right| \times 100\% = 0.09657\%$$

Four-point formula:

$$\begin{aligned}
I &= 0.3478548 f(-0.861136312) + 0.6521452 f(-0.339981044) + 0.6521452 f(0.339981044) \\
&\quad + 0.3478548 f(0.861136312) \\
&= 0.3478548(0.384798) + 0.6521452(3.996712) + 0.6521452(22.50094) + 0.3478548(68.294) \\
&= 0.133854 + 2.606436 + 14.67388 + 23.7564 = 41.17057
\end{aligned}$$

$$\varepsilon_t = \left| \frac{41.17107 - 41.17057}{41.17107} \right| \times 100\% = 0.001229\%$$

22.6 Change of variable:

$$x = \frac{2+0}{2} + \frac{2-0}{2} x_d = 1 + x_d$$

$$dx = \frac{2-0}{2} dx_d = dx_d$$

$$I = \int_{-1}^1 \frac{e^{1+x_d} \sin(1+x_d)}{1+(1+x_d)^2} dx_d$$

Therefore, the transformed function is

$$f(x_d) = \frac{e^{1+x_d} \sin(1+x_d)}{1+(1+x_d)^2}$$

Five-point formula:

$$\begin{aligned}
I &= 0.236927 f(-0.90618) + 0.478629 f(-0.53847) + 0.568889 f(0) \\
&\quad + 0.478629 f(0.53847) + 0.236927 f(0.90618) \\
&= 0.236927(0.102) + 0.478629(0.582434) + 0.568889(1.143678) \\
&\quad + 0.478629(1.382589) + 0.236927(1.370992) \\
&= 0.024166442 + 0.278769811 + 0.650625504 + 0.661746769 + 0.324824846 = 1.940133372
\end{aligned}$$

$$\varepsilon_t = \left| \frac{1.940130022 - 1.940133372}{1.940130022} \right| \times 100\% = 0.000173\%$$

22.7 Here is the Romberg tableau for this problem.

	1	2	3
<i>n</i>	<i>ε_a →</i>	5.5616%	0.0188%
1	224.36568786	288.56033084	289.43080513
2	272.51167009	289.37640049	
4	285.16021789		

Therefore, the estimate is 289.430805.

22.8 Change of variable:

$$x = \frac{3-3}{2} + \frac{3-(-3)}{2} x_d = 3x_d$$

$$dx = \frac{3-(-3)}{2} dx_d = 3dx_d$$

$$I = \int_{-1}^1 \frac{3}{1+9x_d^2} dx_d$$

Therefore, the transformed function is

$$f(x_d) = \frac{3}{1+9x_d^2}$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 0.75 + 0.75 = 1.5$$

$$\varepsilon_t = \left| \frac{2.49809 - 1.5}{2.49809} \right| \times 100\% = 39.95\%$$

The remaining formulas can be implemented with the results summarized in this table.

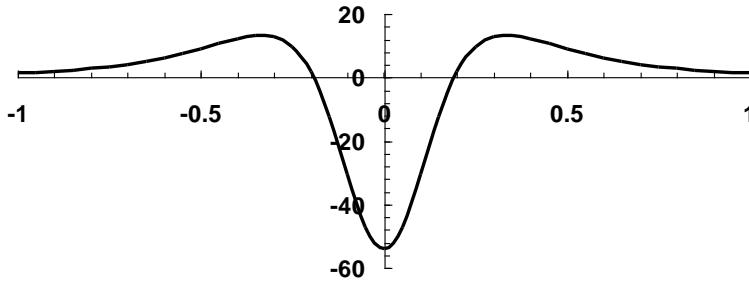
n	Integral	ε_t
2	1.5	39.95%
3	3.1875	27.60%
4	2.189781	12.34%
5	2.671698	6.95%
6	2.411356	3.47%

Thus, the results are converging, but at a very slow rate. Insight into this behavior can be gained by looking at the function and its derivatives.

$$f'(x_d) = -\frac{54x_d}{(1+9x_d^2)^2}$$

$$f''(x_d) = -\frac{54(1+9x_d^2)^2 - 1944x_d^2(1+9x_d^2)}{(1+9x_d^2)^4}$$

We can plot the second derivative as



The second and higher derivatives are large. Thus, the integral evaluation is inaccurate because the error is related to the magnitudes of the derivatives.

22.9 (a)

$$\int_2^\infty \frac{dx}{x(x+2)} = \int_0^{0.5} \frac{1}{t^2} (t) \frac{1}{1/t+2} dt = \int_0^{0.5} \frac{1}{1+2t} dt$$

We can use 8 applications of the extended midpoint rule.

$$\frac{1}{16} (0.941176 + 0.842105 + 0.761905 + 0.695652 + 0.64 + 0.592593 + 0.551724 + 0.516129) = 0.34633$$

This result is close to the analytical solution

$$\int_2^\infty \frac{dx}{x(x+2)} = \left[0.5 \ln\left(\frac{x}{x+2}\right) \right]_2^\infty = 0.5 \ln\left(\frac{\infty}{\infty+2}\right) - 0.5 \ln\left(\frac{2}{2+2}\right) = 0.346574$$

(b)

$$\int_0^\infty e^{-y} \sin^2 y dy = \int_0^2 e^{-y} \sin^2 y dy + \int_2^\infty e^{-y} \sin^2 y dy$$

For the first part, we can use 4 applications of Simpson's 1/3 rule

$$I = (2-0) \frac{0 + 4(0.048 + 0.219 + 0.258 + 0.168) + 2(0.139 + 0.26 + 0.222) + 0.112}{24} = 0.344115$$

For the second part,

$$\int_2^\infty e^{-y} \sin^2 y dy = \int_0^{1/2} \frac{1}{t^2} e^{-1/t} \sin^2(1/t) dt$$

We can use the extended midpoint rule with $h = 1/8$.

$$I = \frac{1}{8} (0 + 0.0908 + 0.00142 + 0.303) = 0.0494$$

The total integral is

$$I = 0.344115 + 0.0494 = 0.393523$$

This result is close to the analytical solution of 0.4.

(c) Errata: In the book's first printing, this function was erroneously printed as

$$\int_0^\infty \frac{1}{(1+y^2)(1-y^2/2)} dy$$

The correct function, which appears in later printings, is

$$\int_0^\infty \frac{1}{(1+y^2)(1+y^2/2)} dy$$

Solution:

$$\int_0^\infty \frac{1}{(1+y^2)(1+y^2/2)} dy = \int_0^2 \frac{1}{(1+y^2)(1+y^2/2)} dy + \int_2^\infty \frac{1}{(1+y^2)(1+y^2/2)} dy$$

For the first part, we can use Simpson's 1/3 rule

$$(2-0) \frac{1+4(0.9127 + 0.4995 + 0.2191 + 0.0972) + 2(0.7111 + 0.3333 + 0.1448) + 0.0667}{24} = 0.863262$$

For the second part,

$$\int_2^\infty \frac{1}{(1+y^2)(1+y^2/2)} dy = \int_0^{1/2} \frac{1}{t^2(1+(1/t)^2)(1+1/(2t^2))} dt$$

We can use the extended midpoint rule with $h = 1/8$.

$$I = \frac{1}{8}(0.007722 + 0.063462 + 0.148861 + 0.232361) = 0.056551$$

The total integral is

$$I = 0.863262 + 0.056551 = 0.919813$$

This result is close to the analytical solution of 0.920151.

(d)

$$\int_{-2}^\infty ye^{-y} dy = \int_{-2}^2 ye^{-y} dy + \int_2^\infty ye^{-y} dy$$

For the first part, we can use 4 applications of Simpson's 1/3 rule

$$I = (2 - (-2)) \frac{-14.78 + 4(-6.72 - 0.824 + 0.303 + 0.335) + 2(-2.72 + 0 + 0.368) + 0.2707}{24} = -7.807$$

For the second part,

$$\int_2^{\infty} ye^{-y} dy = \int_0^{1/2} \frac{1}{t^3} e^{-1/t} dt$$

We can use the extended midpoint rule with $h = 1/8$.

$$I = \frac{1}{8} (0.000461 + 0.732418 + 1.335696 + 1.214487) = 0.410383$$

The total integral is

$$I = -7.80733 + 0.410383 = -7.39695$$

(e) Errata: In the book's first printing, this function was erroneously printed as

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2} dx$$

The correct function, which appears in later printings, is

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

Solution:

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \int_0^2 \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx + \int_2^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

For the first part, we can use 4 applications of Simpson's 1/3 rule

$$I = (2 - 0) \frac{0.399 + 4(0.387 + 0.301 + 0.183 + 0.086) + 2(0.352 + 0.242 + 0.130) + 0.054}{24} = 0.47725$$

For the second part,

$$\int_2^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \int_0^{1/2} \frac{1}{\sqrt{2\pi}} \frac{1}{t^2} e^{-1/(2t^2)} dt$$

We can use the extended midpoint rule with $h = 1/8$.

$$I = \frac{1}{8}(0 + 0 + 0.024413063 + 0.152922154) = 0.02217$$

The total integral is

$$I = 0.47725 + 0.02217 = 0.499415$$

This is close to the exact value of 0.5.

22.10 (a) Here is a VBA program to implement the algorithm from Fig. 22.1a. It is set up to evaluate the integral in the problem statement,

```
Option Explicit

Sub TrapTest()
    Dim a As Double, b As Double
    Dim n As Integer
    a = 0
    b = 1
    n = 4
    MsgBox TrapEq(n, a, b)
End Sub

Function TrapEq(n, a, b)
    Dim h As Double, x As Double, sum As Double
    Dim i As Integer
    h = (b - a) / n
    x = a
    sum = f(x)
    For i = 1 To n - 1
        x = x + h
        sum = sum + 2 * f(x)
    Next i
    sum = sum + f(b)
    TrapEq = (b - a) * sum / (2 * n)
End Function

Function f(x)
    f = x ^ 0.1 * (1.2 - x) * (1 - Exp(20 * (x - 1)))
End Function
```

When the program is run, the result is



The percent relative error can be computed as

$$\varepsilon_t = \left| \frac{0.602298 - 0.478602}{0.602298} \right| \times 100\% = 20.54\%$$

(b) Here is a VBA program to implement the algorithm from Fig. 22.1b. It is set up to evaluate the integral in the problem statement,

```

Option Explicit

Sub SimpTest()
    Dim a As Double, b As Double
    Dim n As Integer
    a = 0
    b = 1
    n = 4
    MsgBox SimpEq(n, a, b)
End Sub

Function SimpEq(n, a, b)
    Dim h As Double, x As Double, sum As Double
    Dim i As Integer
    h = (b - a) / n
    x = a
    sum = f(x)
    For i = 1 To n - 2 Step 2
        x = x + h
        sum = sum + 4 * f(x)
        x = x + h
        sum = sum + 2 * f(x)
    Next i
    x = x + h
    sum = sum + 4 * f(x)
    sum = sum + f(b)
    SimpEq = (b - a) * sum / (3 * n)
End Function

Function f(x)
    f = x ^ 0.1 * (1.2 - x) * (1 - Exp(20 * (x - 1)))
End Function

```

When the program is run, the result is



The percent relative error can be computed as

$$\varepsilon_t = \left| \frac{0.602298 - 0.529287}{0.602298} \right| \times 100\% = 12.12\%$$

22.11 Here is a VBA program to implement the algorithm from Fig. 22.4. It is set up to evaluate the integral from Prob. 22.10.

```

Option Explicit

Sub RhombTest()
    Dim maxit As Integer
    Dim a As Double, b As Double, es As Double
    a = 0
    b = 1
    maxit = 10
    es = 0.00000001
    MsgBox Rhomberg(a, b, maxit, es)
End Sub

Function Rhomberg(a, b, maxit, es)
    Dim n As Long, j As Integer, k As Integer, iter As Integer
    Dim i(11, 11) As Double, ea As Double
    n = 1
    i(1, 1) = TrapEq(n, a, b)
    iter = 0
    Do
        iter = iter + 1
        n = 2 ^ iter
        i(iter + 1, 1) = TrapEq(n, a, b)
        For k = 2 To iter + 1
            j = 2 + iter - k
            i(j, k) = (4 ^ (k - 1)) * i(j + 1, k - 1) - i(j, k - 1)) / (4 ^ (k - 1) - 1)
        Next k
        ea = Abs((i(1, iter + 1) - i(2, iter)) / i(1, iter + 1)) * 100
        If (iter >= maxit Or ea <= es) Then Exit Do
    Loop
    Rhomberg = i(1, iter + 1)
End Function

Function TrapEq(n, a, b)
    Dim i As Long
    Dim h As Double, x As Double, sum As Double
    h = (b - a) / n
    x = a
    sum = f(x)
    For i = 1 To n - 1
        x = x + h
        sum = sum + 2 * f(x)
    Next i
    sum = sum + f(b)
    TrapEq = (b - a) * sum / (2 * n)
End Function

Function f(x)
    f = x ^ 0.1 * (1.2 - x) * (1 - Exp(20 * (x - 1)))
End Function

```

When the program is run for the function from Prob. 22.10, the result is



The percent relative error can be computed as

$$\varepsilon_t = \left| \frac{0.602298 - 0.6021617}{0.602298} \right| \times 100\% = 0.0226\%$$

22.12 Here is a VBA program to implement an algorithm for Gauss quadrature. It is set up to evaluate the integral from Prob. 22.10.

```

Option Explicit

Sub GaussQuadTest()
    Dim i As Integer, j As Integer, k As Integer
    Dim a As Double, b As Double, a0 As Double, a1 As Double, sum As
    Double
    Dim c(11) As Double, x(11) As Double, j0(5) As Double, j1(5) As
    Double
    'set constants
    c(1) = 1#: c(2) = 0.888888889: c(3) = 0.555555556: c(4) =
    0.652145155
    c(5) = 0.347854845: c(6) = 0.568888889: c(7) = 0.478628671: c(8) =
    0.236926885
    c(9) = 0.467913935: c(10) = 0.360761573: c(11) = 0.171324492
    x(1) = 0.577350269: x(2) = 0: x(3) = 0.774596669: x(4) = 0.339981044
    x(5) = 0.861136312: x(6) = 0: x(7) = 0.53846931: x(8) = 0.906179846
    x(9) = 0.238619186: x(10) = 0.661209386: x(11) = 0.932469514
    j0(1) = 1: j0(2) = 3: j0(3) = 4: j0(4) = 7: j0(5) = 9
    j1(1) = 1: j1(2) = 3: j1(3) = 5: j1(4) = 8: j1(5) = 11
    a = 0
    b = 1
    Sheets("Sheet1").Select
    Range("a1").Select
    For i = 1 To 5
        ActiveCell.Value = GaussQuad(i, a, b, c, x, j0, j1)
        ActiveCell.Offset(1, 0).Select
    Next i
    End Sub

    Function GaussQuad(n, a, b, c, x, j0, j1)
        Dim k As Integer, j As Integer
        Dim a0 As Double, a1 As Double
        Dim sum As Double
        a0 = (b + a) / 2
        a1 = (b - a) / 2

```

```

sum = 0
If Int(n / 2) - n / 2# = 0 Then
    k = (n - 1) * 2
    sum = sum + c(k) * a1 * f(fc(x(k), a0, a1))
End If
For j = j0(n) To j1(n)
    sum = sum + c(j) * a1 * f(fc(-x(j), a0, a1))
    sum = sum + c(j) * a1 * f(fc(x(j), a0, a1))
Next j
GaussQuad = sum
End Function

Function fc(xd, a0, a1)
fc = a0 + a1 * xd
End Function

Function f(x)
f = x ^ 0.1 * (1.2 - x) * (1 - Exp(20 * (x - 1)))
End Function

```

When the program is run, the results along with the error estimate are

<i>n</i>	<i>integral</i>	<i>&_t</i>
2	0.621078	3.12%
3	0.609878	1.26%
4	0.605242	0.49%
5	0.603822	0.25%
6	0.603317	0.17%

22.13 Change of variable:

$$x = \frac{1.5+0}{2} + \frac{1.5-0}{2} x_d = 0.75 + 0.75 x_d$$

$$dx = \frac{1.5-0}{2} dx_d = 0.75 dx_d$$

$$I = \int_{-1}^1 \frac{1.5}{\sqrt{\pi}} e^{-(0.75+0.75x_d)^2} dx_d$$

Therefore, the transformed function is

$$f(x_d) = \frac{1.5}{\sqrt{\pi}} e^{-(0.75+0.75x_d)^2}$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 0.765382 + 0.208792 = 0.974173$$

$$\varepsilon_t = \left| \frac{0.966105 - 0.974173}{0.966105} \right| \times 100\% = 0.835\%$$

22.14 The function to be integrated is

$$M = \int_2^8 (9 + 4 \cos^2(0.4t)) (5e^{-0.5t} + 2e^{0.15t}) dt$$

The first iteration involves computing 1 and 2 segment trapezoidal rules and combining them as

$$I = \frac{4(340.68170896) - 411.26095167}{3} = 317.15529472$$

and computing the approximate error as

$$\varepsilon_a = \left| \frac{317.15529472 - 340.68170896}{317.15529472} \right| \times 100\% = 7.4179\%$$

The computation can be continues as in the following tableau until $\varepsilon_a < 0.1\%$.

	1	2	3	4
<i>n</i>	$\varepsilon_a \rightarrow$	7.4179%	0.1054%	0.0012119%
1	411.26095167	317.15529472	322.59571622	322.34570788
2	340.68170896	322.25568988	322.34961426	
4	326.86219465	322.34374398		
8	323.47335665			

22.15 The first iteration involves computing 1 and 2 segment trapezoidal rules and combining them as

$$I = (16 - 0) \frac{0 + 0}{2} = 0$$

$$I = (16 - 0) \frac{0 + 2(2.4) + 0}{4} = 19.2$$

$$I = \frac{4(19.2) - 0}{3} = 25.6$$

and computing the approximate error as

$$\varepsilon_a = \left| \frac{25.6 - 19.2}{25.6} \right| \times 100\% = 25\%$$

The computation can be continues as in the following tableau until $\varepsilon_a < 1\%$.

n	$\varepsilon_a \rightarrow$	1	2	3
1	0.00000000	25.0000%	0.7888%	
2	19.20000000	29.06666667		
4	26.60000000			

CHAPTER 23

23.1

	x	$f(x)$
x_{i-2}	0.261799388	0.965925826
x_{i-1}	0.523598776	0.866025404
x_i	0.785398163	0.707106781
x_{i+1}	1.047197551	0.5
x_{i+2}	1.308996939	0.258819045

$$\text{true} = -\sin(\pi/4) = -0.70710678$$

The results are summarized as

	first-order	second-order
Forward	-0.79108963 -11.877%	-0.72601275 -2.674%
Backward	-0.60702442 14.154%	-0.71974088 -1.787%
Centered	-0.69905703 1.138%	-0.70699696 0.016%

23.2

	x	$f(x)$
x_{i-2}	21	1.322219295
x_{i-1}	23	1.361727836
x_i	25	1.397940009
x_{i+1}	27	1.431363764
x_{i+2}	29	1.462397998

$$\text{truth} = \frac{\log_{10}(e)}{25} = 0.017371779$$

The results are summarized as

	first-order	second-order
Forward	0.016711878 3.799%	0.017309258 0.360%
Backward	0.018106086 -4.227%	0.017281994 0.517%
Centered	0.017408982 -0.214%	0.017371197 0.003%

23.3

	x	f(x)
x_{i-2}	1.8	6.049647464
x_{i-1}	1.9	6.685894442
x_i	2	7.389056099
x_{i+1}	2.1	8.166169913
x_{i+2}	2.2	9.025013499

Both the first and second derivatives have the same value,

$$\text{truth} = e^2 = 7.389056099$$

The results are summarized as

	first-order	second-order
First derivative	7.401377351 -0.166750%	7.389031439 0.000334%
Second derivative	7.395215699 -0.083361%	7.389047882 0.000111%

23.4 The true value is $-\sin(\pi/4) = -0.70710678$.

$$D(\pi/3) = \frac{-0.25882 - 0.965926}{2(1.047198)} = -0.58477$$

$$D(\pi/6) = \frac{0.258819 - 0.965926}{2(0.523599)} = -0.67524$$

$$D = \frac{4}{3}(-0.67524) - \frac{1}{3}(-0.58477) = -0.70539$$

23.5 The true value is $1/x = 1/5 = 0.2$.

$$D(2) = \frac{1.94591 - 1.098612}{2(2)} = 0.211824$$

$$D(1) = \frac{1.791759 - 1.386294}{2(1)} = 0.202733$$

$$D = \frac{4}{3}(0.202733) - \frac{1}{3}(0.211824) = 0.199702$$

23.6 The true value

$$f'(0) = 8(0)^3 - 18(0)^2 - 12 = -12$$

Equation (23.9) can be used to compute the derivative as

$$x_0 = -0.5 \quad f(x_0) = -1.125$$

$$x_1 = 1 \quad f(x_1) = -24$$

$$x_2 = 2 \quad f(x_2) = -48$$

$$\begin{aligned} f'(0) &= -1.125 \frac{2(0) - 1 - 2}{(-0.5 - 1)(-0.5 - 2)} + (-24) \frac{2(0) - (-0.5) - 2}{(1 - (-0.5))(1 - 2)} + (-48) \frac{2(0) - (-0.5) - 1}{(2 - (-0.5))(2 - 1)} \\ &= 0.9 - 24 + 9.6 = -13.5 \end{aligned}$$

Centered difference:

$$f'(0) = \frac{-24 - 12}{1 - (-1)} = -18$$

23.7 At $x = x_i$, Eq. (23.9) is

$$\begin{aligned} f'(x) &= f(x_{i-1}) \frac{2x_i - x_i - x_{i+1}}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + f(x_i) \frac{2x_i - x_{i-1} - x_{i+1}}{(x_i - x_{i-1})(x_i - x_{i+1})} \\ &\quad + f(x_{i+1}) \frac{2x_i - x_{i-1} - x_i}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} \end{aligned}$$

For equispaced points that are h distance apart, this equation becomes

$$\begin{aligned} f'(x) &= f(x_{i-1}) \frac{-h}{-h(-2h)} + f(x_i) \frac{2x_i - (x_i - h) - (x_i + h)}{h(-h)} + f(x_{i+1}) \frac{h}{2h(h)} \\ &= \frac{-f(x_{i-1})}{2h} + 0 + \frac{f(x_{i+1}) - f(x_{i-1})}{2h} = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \end{aligned}$$

23.8 (a)

	<i>x</i>	<i>f(x)</i>
x_{i-2}	-0.5	-17.125
x_{i-1}	-0.25	-16.0156
x_i	0	-15
x_{i+1}	0.25	-13.9844
x_{i+2}	0.5	-12.875

$$f'(x) = \frac{-(-12.875) + 8(-13.9844) - 8(-16.0156) - 17.125}{12(0.25)} = 4$$

$$f''(x) = \frac{-(-12.875) + 16(-13.9844) - 30(-15) + 16(-16.0156) - (-17.125)}{12(0.25)^2} = 0$$

(b)

	x	f(x)
x_{i-2}	0.2	0.039203
x_{i-1}	0.3	0.08598
x_i	0.4	0.14737
x_{i+1}	0.5	0.219396
x_{i+2}	0.6	0.297121

$$f'(x) = \frac{-(0.297121) + 8(0.219396) - 8(0.08598) + 0.039203}{12(0.1)} = 0.674504$$

$$f''(x) = \frac{-(0.297121) + 16(0.219396) - 30(0.14737) + 16(0.08598) - (0.039203)}{12(0.1)^2} = 1.071654$$

(c)

	x	f(x)
x_{i-2}	2	0.786843
x_{i-1}	2.5	1.100778
x_i	3	1.557408
x_{i+1}	3.5	2.338254
x_{i+2}	4	4.131729

$$f'(x) = \frac{-(4.131729) + 8(2.338254) - 8(1.100778) + 0.786843}{12(0.5)} = 1.092486$$

$$f''(x) = \frac{-(4.131729) + 16(2.338254) - 30(1.557408) + 16(1.100778) - (0.786843)}{12(0.5)^2} = 1.127902$$

(d)

	x	f(x)
x_{i-2}	0.6	0.62948
x_{i-1}	0.8	0.540569
x_i	1	0.479426
x_{i+1}	1.2	0.433954
x_{i+2}	1.4	0.398355

$$f'(x) = \frac{-(0.398355) + 8(0.433954) - 8(0.540569) + 0.62948}{12(0.2)} = -0.25908$$

$$f''(x) = \frac{-(0.398355) + 16(0.433954) - 30(0.479426) + 16(0.540569) - (0.62948)}{12(0.2)^2} = 0.378652$$

(e)

	x	f(x)
x_{i-2}	1.6	6.553032
x_{i-1}	1.8	7.849647
x_i	2	9.389056
x_{i+1}	2.2	11.22501
x_{i+2}	2.4	13.42318

$$f'(x) = \frac{-(13.42318) + 8(11.22501) - 8(7.849647) + 6.553032}{12(0.2)} = 8.38866$$

$$f''(x) = \frac{-(13.42318) + 16(11.22501) - 30(9.389056) + 16(7.849647) - (6.553032)}{12(0.2)^2} = 7.388924$$

23.9 The first forward difference formula of $O(h^2)$ from Fig. 23.1 can be used to estimate the velocity for the first point at $t = 0$,

$$f'(0) = \frac{-58 + 4(32) - 3(0)}{2(25)} = 1.4 \frac{\text{km}}{\text{s}}$$

The acceleration can be estimated with the second forward difference formula of $O(h^2)$ from Fig. 23.1

$$f''(0) = \frac{-78 + 4(58) - 5(32) + 2(0)}{(25)^2} = -0.0096 \frac{\text{km}}{\text{s}^2}$$

For the interior points, centered difference formulas of $O(h^2)$ from Fig. 23.3 can be used to estimate the velocities and accelerations. For example, at the second point at $t = 25$,

$$f'(25) = \frac{58 - 0}{2(25)} = 1.16 \frac{\text{km}}{\text{s}}$$

$$f''(25) = \frac{58 - 2(32) + 0}{(25)^2} = -0.0096 \frac{\text{km}}{\text{s}^2}$$

For the final point, backward difference formulas of $O(h^2)$ from Fig. 23.2 can be used to estimate the velocities and accelerations. The results for all values are summarized in the following table.

t	y	v	a
0	0	1.40	-0.0096
25	32	1.16	-0.0096
50	58	0.92	-0.0096
75	78	0.68	-0.0096
100	92	0.44	-0.0096

125	100	0.20	-0.0096
-----	-----	------	---------

23.10 Here is a VBA program to implement a Romberg algorithm to estimate the derivative of a given function. It is set up to evaluate the derivative from Example 23.1.

```

Option Explicit

Sub RhombTest()
Dim maxit As Integer
Dim es As Double, x As Double
x = 0.5
maxit = 3
es = 0.001
MsgBox RhomDiff(x, maxit, es)
End Sub

Function RhomDiff(x, maxit, es)
Dim n As Integer, j As Integer, k As Integer, iter As Integer
Dim i(10, 10) As Double, ea As Double, del As Double
n = 1
i(1, 1) = DyDx(x, n)
iter = 0
Do
    iter = iter + 1
    n = 2 ^ iter
    i(iter + 1, 1) = DyDx(x, n)
    For k = 2 To iter + 1
        j = 2 + iter - k
        i(j, k) = (4 ^ (k - 1)) * i(j + 1, k - 1) - i(j, k - 1)) / (4 ^ (k - 1) - 1)
    Next k
    ea = Abs((i(1, iter + 1) - i(1, iter)) / i(1, iter + 1)) * 100
    If (iter >= maxit Or ea <= es) Then Exit Do
Loop
RhomDiff = i(1, iter + 1)
End Function

Function DyDx(x, n)
Dim a As Double, b As Double
a = x - x / n
b = x + x / n
DyDx = (f(b) - f(a)) / (b - a)
End Function

Function f(x)
f = -0.1 * x ^ 4 - 0.15 * x ^ 3 - 0.5 * x ^ 2 - 0.25 * x + 1.2
End Function

```

When the program is run, the result is



23.11 Here is a VBA program uses Eq. 23.9 to obtain first-derivative estimates for unequally spaced data.

```

Option Explicit

Sub TestDerivUnequal()
Dim n As Integer, i As Integer
Dim x(100) As Double, y(100) As Double, dy(100) As Double
Range("a5").Select
n = ActiveCell.Row
Selection.End(xlDown).Select
n = ActiveCell.Row - n
Range("a5").Select
For i = 0 To n
    x(i) = ActiveCell.Value
    ActiveCell.Offset(0, 1).Select
    y(i) = ActiveCell.Value
    ActiveCell.Offset(1, -1).Select
Next i
For i = 0 To n
    dy(i) = DerivUnequal(x, y, n, x(i))
Next i
Range("c5").Select
For i = 0 To n
    ActiveCell.Value = dy(i)
    ActiveCell.Offset(1, 0).Select
Next i
End Sub

Function DerivUnequal(x, y, n, xx)
Dim ii As Integer
If xx < x(0) Or xx > x(n) Then
    DerivUnequal = "out of range"
Else
    If xx < x(1) Then
        DerivUnequal = DyDx(xx, x(0), x(1), x(2), y(0), y(1), y(2))
    ElseIf xx > x(n - 1) Then
        DerivUnequal =
            DyDx(xx, x(n - 2), x(n - 1), x(n), y(n - 2), y(n - 1), y(n))
    Else
        For ii = 1 To n - 2
            If xx >= x(ii) And xx <= x(ii + 1) Then
                If xx - x(ii - 1) < x(ii) - xx Then
                    'If the unknown is closer to the lower end of the range,
                    'x(ii) will be chosen as the middle point
                    DerivUnequal =
                        DyDx(xx, x(ii - 1), x(ii), x(ii + 1), y(ii - 1), y(ii), y(ii + 1))
                Else
                    'Otherwise, if the unknown is closer to the upper end,
                    'x(ii+1) will be chosen as the middle point
                    DerivUnequal =
                        DyDx(xx, x(ii), x(ii + 1), x(ii + 2), y(ii), y(ii + 1), y(ii + 2))
                End If
                Exit For
            End If
        Next ii
    End If
End If
End If

End Function

```

```

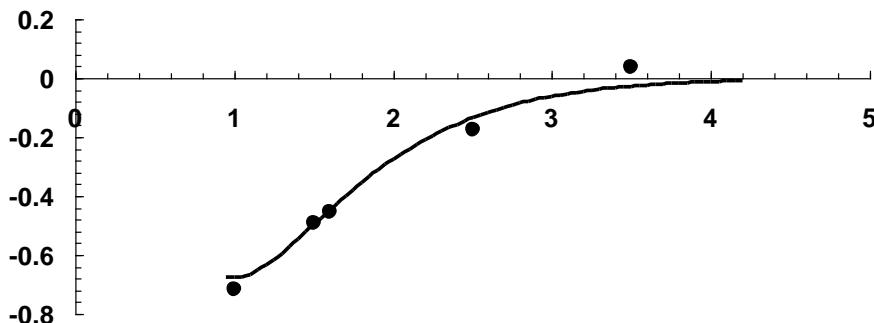
Function DyDx(x, x0, x1, x2, y0, y1, y2)
DyDx = y0 * (2 * x - x1 - x2) / (x0 - x1) / (x0 - x2) -
       + y1 * (2 * x - x0 - x2) / (x1 - x0) / (x1 - x2) -
       + y2 * (2 * x - x0 - x1) / (x2 - x0) / (x2 - x1)
End Function

```

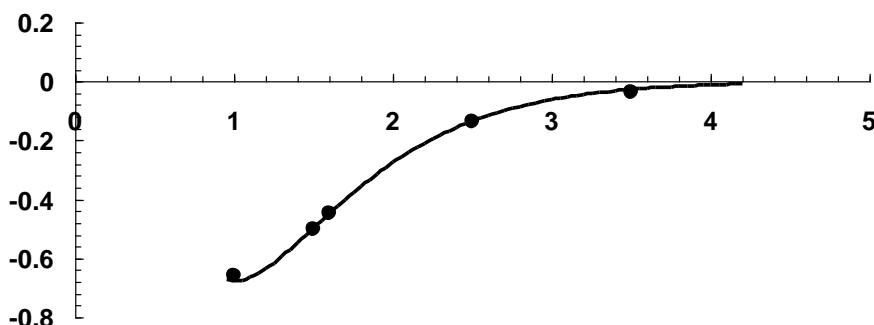
When the program is run, the result is shown below:

	A	B	C
1	Problem 23.11		
2			
3	Data:		
4	x	y	dy/dx
5	1	0.6767	-0.71793
6	1.5	0.3734	-0.49342
7	1.6	0.3261	-0.45258
8	2.5	0.08422	-0.17378
9	3.5	0.01596	0.037264

The results can be compared with the true derivatives which can be calculated with analytical solution, $f'(x) = 5e^{-2x} - 10xe^{-2x}$. The results can be displayed graphically below where the computed values are represented as points and the true values as the curve.



An even more elegant approach is to put cubic splines through the data (recall Sec. 20.2 and the program used for the solution to Prob. 20.10) to evaluate the derivatives. Here is the result of applying that program to this problem.



23.12 (a) Create the following M function:

```
function y=f(t)
y=9.81*70/12*(1-exp(-12/70*t));
```

Then implement the following MATLAB session:

```
>> Q=quad(@f, 0, 10)
```

```
Q =
298.5546
```

(b)

$$d(t) = \frac{gm}{c} \int_0^t (1 - e^{-(c/m)t}) dt$$

$$d(t) = \frac{gm}{c} \left[t + \frac{m}{c} e^{-(c/m)t} \right]_0^t$$

$$d(10) = \frac{9.81(70)}{12} \left[10 + \frac{70}{12} e^{-(12/70)10} - 0 - \frac{70}{12} \right] = 298.5546$$

(c) Implement the following MATLAB session:

```
>> x=[9.99 10.01];
>> y=f(x);
>> d=diff(y)./diff(x)
```

```
d =
1.7667
```

(d)

$$a(t) = \frac{gm}{c} \frac{d}{dt} (1 - e^{-(c/m)t})$$

$$a(t) = ge^{-(c/m)t}$$

$$a(10) = 9.81e^{-(12/70)10} = 1.766706$$

23.13 (a) Create the following M function:

```
function y=fn(x)
y=1/sqrt(2*pi)*exp(-(x.^2)/2);
```

Then implement the following MATLAB session:

```
>> x=-2:.1:2;
>> y=fn(x);
>> Q=quad(@fn,-1,1)
```

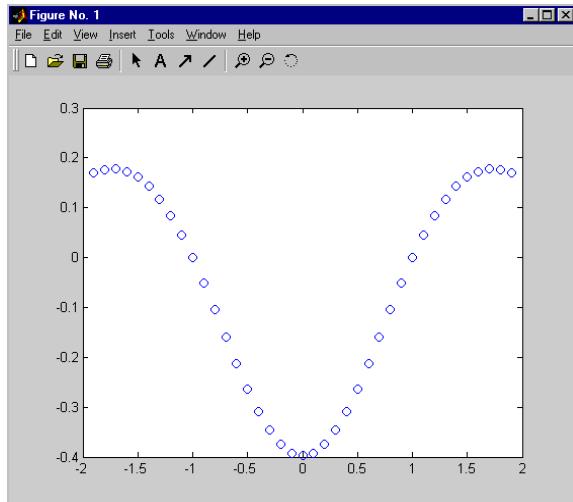
```
Q =
0.6827
```

```
>> Q=quad(@fn, -2, 2)
Q =
    0.9545
```

Thus, about 68.3% of the area under the curve falls between -1 and 1 and about 95.45% falls between -2 and 2 .

(b)

```
>> x=-2:.1:2;
>> y=fn(x);
>> d=diff(y)./diff(x);
>> x=-1.95:.1:1.95;
>> d2=diff(d)./diff(x);
>> x=-1.9:.1:1.9;
>> plot(x,d2,'o')
```



Thus, inflection points ($d^2y/dx^2 = 0$) occur at -1 and 1 .

23.14 (a) Create the following M function:

```
function y=fn(x)
y=1/sqrt(2*pi)*exp(-(x.^2)/2);
```

Then implement the following MATLAB session:

```
>> x=-1:.5:1;
>> y=fn(x);
>> I=trapz(x,y)

I =
    0.6725

>> x=-2:.5:2;
>> y=fn(x);
>> I=trapz(x,y)
```

```
I =
    0.9500
```

Note that there is another MATLAB function called `trap` that can also be used to solve this problem. To learn more about this function type

```
>> help trap
```

The resulting description will be displayed,

```
I = trap(func,a,b,n):
    multiple-application trapezoidal rule.
input:
    func = name of function to be integrated
    a, b = integration limits
    n = number of segments
output:
    I = integral estimate
```

Here are the results of using this function to solve the problem:

```
>> I=trap(@fn,-1,1,4)
```

```
I =
    0.6725
```

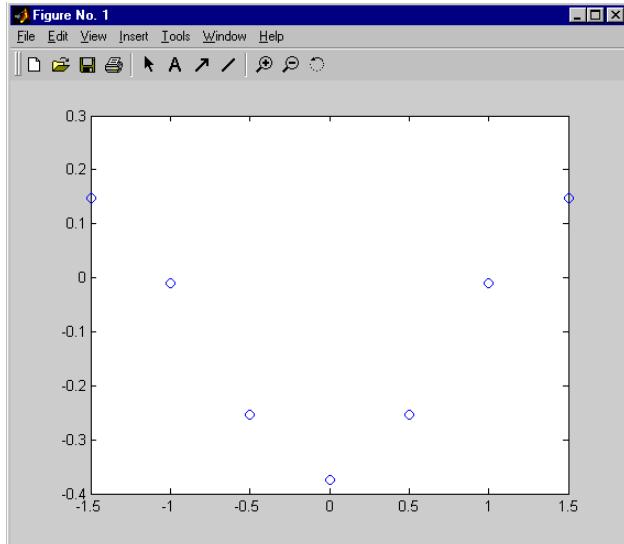
```
>> I=trap(@fn,-2,2,8)
```

```
I =
    0.9500
```

Thus, about 67.25% of the area under the curve falls between -1 and 1 and about 95% falls between -2 and 2 .

(b)

```
>> d=diff(y)./diff(x);
>> x=-1.75:.5:1.75;
>> d2=diff(d)./diff(x);
>> x=-1.5:.5:1.5;
>> plot(x,d2,'o')
```



Thus, inflection points ($d^2y/dx^2 = 0$) occur at -1 and 1 .

23.15

```
Program Integrate
Use imsl
Implicit None
Integer::irule=1
Real::a=-1.,b=1,errabs=0.0,errrel=0.001
Real::errest,res,f
External f
Call QDAG(f,a,b,errabs,errrel,irule,res,errest)
Print '('' Computed = ''',F8.4)',res
Print '('' Error estimate ='',1PE10.3)',errest
End Program

Function f(x)
Implicit None
Real:: x , f
Real::pi
Parameter(pi=3.1415927)
f=1/sqrt(2.*pi)*exp(-x**2/2.)
End Function
```

Answers:

```
x = -1 to 1: Computed = 0.6827 Error estimate = 4.069E-06
x = -2 to 2: Computed = 0.9545 Error estimate = 7.975E-06
x = -3 to 3: Computed = 0.9973 Error estimate = 5.944E-06
```

23.16 MATLAB Script:

```
% Prob2316 Integration program
a=0;
b=pi/2;
integral=quad(@ff,a,b)
```

```

function y=ff(x)
y=cos(cos(x));

>> prob2316

integral =
1.2020

```

23.17 MATLAB Script saved as prob2317.m:

```

%Numerical Integration of sin(t)/t = function sint(t)
%Limits: a=0, b=2pi
%Using the "quad" and "quadl" function for numerical integration
%Plot of function
t=0.01:0.01:2*pi;
y=ff2(t);
plot(t,y); grid
%Integration
format long
a=0.01;
b=2*pi;
Iquad=quad('ff2',a,b)
Iquadl=quadl('ff2',a,b)

function y=ff2(t)
y=sin(t)./t;

```

MATLAB execution:

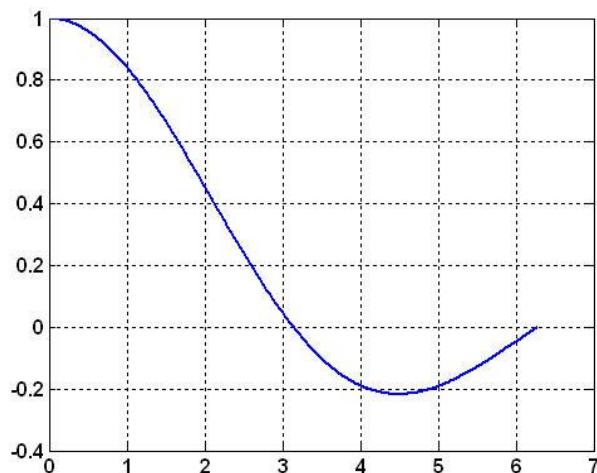
```

>> prob2317

Iquad =
1.40815163720125

Iquadl =
1.40815163168846

```



23.18

```
%Centered Finite Difference First & Second Derivatives of Order O(dx^2)
%Using diff(y)

dx=1.;
y=[1.4 2.1 3.3 4.8 6.8 6.6 8.6 7.5 8.9 10.9 10];
dyf=diff(y);

% First Derivative Centered FD using diff
n=length(y);
for i=1:n-2
    dydxc(i)=(dyf(i+1)+dyf(i))/(2*dx);
end

%Second Derivative Centered FD using diff
dy2dx2c=diff(dyf)/(dx*dx);

fprintf('first derivative \n'); fprintf('%f\n', dydxc)
fprintf('second derivative \n'); fprintf('%f\n', dy2dx2c)

first derivative
0.950000
1.350000
1.750000
0.900000
0.900000
0.450000
0.150000
1.700000
0.550000
second derivative
0.500000
0.300000
0.500000
-2.200000
2.200000
-3.100000
2.500000
0.600000
-2.900000
```

23.19

```
% Finite Difference Approximation of slope
% For f(x)=exp(-2*x)-x
%     f'(x)=-2*exp(-2*x)-1
% Centered diff. df/dx=(f(i+1)-f(i-1))/2dx      + O(dx^2)
% Fwd. diff.      df/dx=(-f(i+2)+4f(i+1)-3f(i))/2dx + O(dx^2)
% Bkwd. diff.     df/dx=(3f(i)-4f(i-1)+f(i-2))/2dx + O(dx^2)

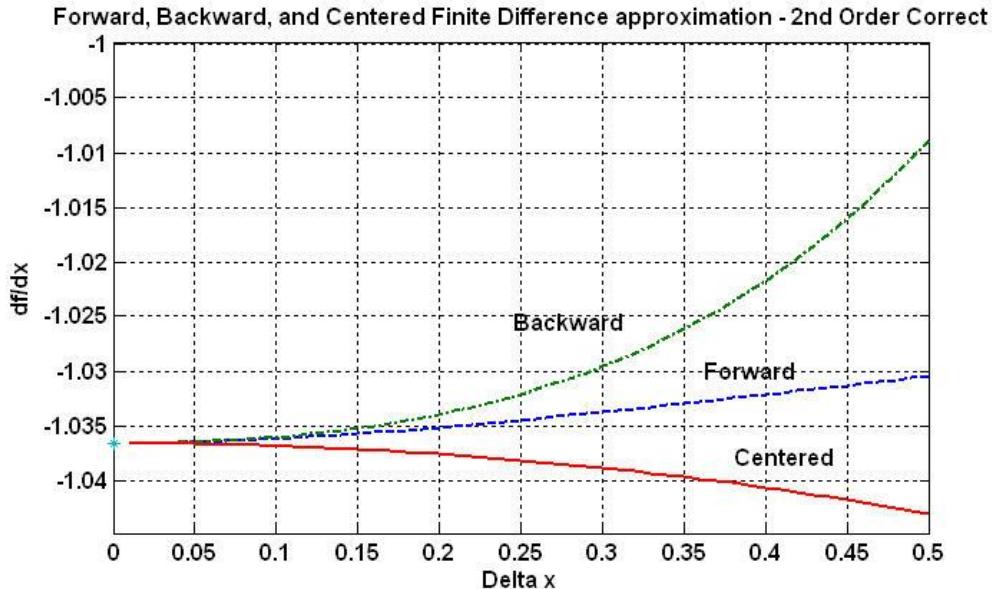
x=2;
fx=exp(-2*x)-x;
dfdx2=-2*exp(-2*x)-1;

%approximation
dx=0.5:-0.01:.01;
for i=1:length(dx)
    %x-values at i-dx and +-2dx
    xp(i)=x+dx(i);
    x2p(i)=x+2*dx(i);
    xn(i)=x-dx(i);
```

```

x2n(i)=x-2*dx(i);
%f(x)-values at i+-dx and +-2dx
fp(i)=exp(-2*xp(i))-xp(i);
f2p(i)=exp(-2*x2p(i))-x2p(i);
fn(i)=exp(-2*xn(i))-xn(i);
f2n(i)=exp(-2*x2n(i))-x2n(i);
%Finite Diff. Approximations
Cdfdx(i)=(fp(i)-fn(i))/(2*dx(i));
Fdfdx(i)=(-f2p(i)+4*fp(i)-3*fx)/(2*dx(i));
Bdfdx(i)=(3*fx-4*fn(i)+f2n(i))/(2*dx(i));
end
dx0=0;
plot(dx,Fdfdx,'--',dx,Bdfdx,'-.',dx,Cdfdx,'-',dx0,dfdx2,'*')
grid
title('Forward, Backward, and Centered Finite Difference approximation - 2nd Order Correct')
xlabel('Delta x')
ylabel('df/dx')
gtext('Centered'); gtext('Forward'); gtext('Backward')

```



23.20 First, we will use forward expansions. The Taylor series expansion about $a = x_i$ and $x = x_{i+2}$ ($2\Delta x$ steps forward) can be written as:

$$\begin{aligned}
 f(x_{i+2}) &= f(x_i) + f'(x_i)2\Delta x + \frac{1}{2}f''(x_i)(2\Delta x)^2 + \frac{1}{6}f'''(x_i)(2\Delta x)^3 + \frac{1}{24}f^{(4)}(x_i)(2\Delta x)^4 \\
 &\quad + \frac{1}{120}f^{(5)}(x_i)(2\Delta x)^5 + \dots \\
 f(x_{i+2}) &= f(x_i) + 2f'(x_i)\Delta x + 2f''(x_i)\Delta x^2 + \frac{8}{6}f'''(x_i)\Delta x^3 + \frac{16}{24}f^{(4)}(x_i)\Delta x^4 \\
 &\quad + \frac{32}{120}f^{(5)}(x_i)\Delta x^5 + \dots
 \end{aligned} \tag{1}$$

Taylor series expansion about $a = x_i$ and $x = x_{i+1}$ (Δx steps forward):

$$f(x_{i+1}) = f(x_i) + f'(x_i)\Delta x + \frac{1}{2}f''(x_i)\Delta x^2 + \frac{1}{6}f'''(x_i)\Delta x^3 + \frac{1}{24}f^{(4)}(x_i)\Delta x^4 + \frac{1}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (2)$$

Multiply Eq. 2 by 2 and subtract the result from Eq. 1 to yield

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)\Delta x^2 + \frac{6}{6}f'''(x_i)\Delta x^3 + \frac{14}{24}f^{(4)}(x_i)\Delta x^4 + \frac{30}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (3)$$

Next, we will use backward expansions. The Taylor series expansion about $a = x_i$ and $x = x_{i-2}$ ($2\Delta x$ steps backward) can be written as:

$$f(x_{i-2}) = f(x_i) + f'(x_i)(-2\Delta x) + \frac{1}{2}f''(x_i)(-2\Delta x)^2 + \frac{1}{6}f'''(x_i)(-2\Delta x)^3 + \frac{1}{24}f^{(4)}(x_i)(-2\Delta x)^4 + \frac{1}{120}f^{(5)}(x_i)(-2\Delta x)^5 + \dots$$

$$f(x_{i-2}) = f(x_i) - 2f'(x_i)\Delta x + 2f''(x_i)\Delta x^2 - \frac{8}{6}f'''(x_i)\Delta x^3 + \frac{16}{24}f^{(4)}(x_i)\Delta x^4 - \frac{32}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (4)$$

Taylor series expansion about $a = x_i$ and $x = x_{i-1}$ (Δx steps backward):

$$f(x_{i-1}) = f(x_i) - f'(x_i)\Delta x + \frac{1}{2}f''(x_i)\Delta x^2 - \frac{1}{6}f'''(x_i)\Delta x^3 + \frac{1}{24}f^{(4)}(x_i)\Delta x^4 - \frac{1}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (5)$$

Multiply Eq. 5 by 2 and subtract the result from Eq. 4 to yield

$$2f(x_{i-1}) - f(x_{i-2}) = f(x_i) - f''(x_i)\Delta x^2 + \frac{6}{6}f'''(x_i)\Delta x^3 - \frac{14}{24}f^{(4)}(x_i)\Delta x^4 + \frac{30}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (6)$$

Add Eqs (3) and (6)

$$f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2}) = 2f'''(x_i)\Delta x^3 + \frac{60}{120}f^{(5)}(x_i)\Delta x^5 + \dots \quad (7)$$

Equation 7 can be solved for

$$f'''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2})}{2\Delta x^3} - \frac{\frac{60}{120} f^{(5)}(x_i) \Delta x^5}{2\Delta x^3} + \dots$$

$$f'''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2})}{2\Delta x^3} - \frac{1}{4} f^{(5)}(x_i) \Delta x^2 + \dots$$

23.21 (a)

$$v = \frac{dx}{dt} = x'(t_i) = \frac{x(t_{i+1}) - x(t_{i-1})}{2h} = \frac{7.3 - 5.1}{4} = 0.55 \frac{\text{m}}{\text{s}}$$

$$a = \frac{d^2x}{dt^2} = x''(t_i) = \frac{x(t_{i+1}) - 2x(t_i) + x(t_{i-1})}{h^2} = \frac{7.3 - 2(6.3) + 5.1}{2^2} = -0.05 \frac{\text{m}}{\text{s}^2}$$

(b)

$$v = \frac{-x(t_{i+2}) + 4x(t_{i+1}) - 3x(t_i)}{2h} = \frac{-8 + 4(7.3) - 3(6.3)}{4} = 0.575 \frac{\text{m}}{\text{s}}$$

$$a = \frac{-x(t_{i+3}) + 4x(t_{i+2}) - 5x(t_{i+1}) + 2x(t_i)}{h^2} = \frac{-8.4 + 4(8) - 5(7.3) + 2(6.3)}{2^2} = -0.075 \frac{\text{m}}{\text{s}^2}$$

(c)

$$v = \frac{3x(t_i) - 4x(t_{i-1}) + x(t_{i-2})}{2h} = \frac{3(6.3) - 4(5.1) + 3.4}{4} = 0.475 \frac{\text{m}}{\text{s}}$$

$$a = \frac{2x(t_i) - 5x(t_{i-1}) + 4x(t_{i-2}) - x(t_{i-3})}{h^2} = \frac{2(6.3) - 5(5.1) + 4(3.4) - 1.8}{2^2} = -0.275 \frac{\text{m}}{\text{s}^2}$$

23.22

$$\dot{\theta} = \frac{d\theta}{dt} = \frac{\theta(t_{i+1}) - \theta(t_{i-1})}{2h} = \frac{0.67 - 0.70}{4} = -0.0075 \text{ rad/s}$$

$$\dot{r} = \frac{dr}{dt} = \frac{r(t_{i+1}) - r(t_{i-1})}{2h} = \frac{6030 - 5560}{4} = 117.5 \text{ m/s}$$

$$\ddot{\theta} = \frac{d^2\theta}{dt^2} = \frac{\theta(t_{i+1}) - 2\theta(t_i) + \theta(t_{i-1})}{h^2} = \frac{0.67 - 2(0.68) + 0.70}{(2)^2} = 0.0025 \text{ rad/s}^2$$

$$\ddot{r} = \frac{d^2r}{dt^2} = \frac{r(t_{i+1}) - 2r(t_i) + r(t_{i-1})}{h^2} = \frac{6030 - 2(5800) + 5560}{(2)^2} = -2.5 \text{ m/s}^2$$

$$\vec{v} = 117.5 \vec{e}_r - 43.5 \vec{e}_\theta$$

$$\vec{a} = -2.82625 \vec{e}_r + 12.7375 \vec{e}_\theta$$

23.23 Use the same program as was developed in the solution of Prob. 23.11

```

Option Explicit

Sub TestDerivUnequal()
    Dim n As Integer, i As Integer
    Dim x(100) As Double, y(100) As Double, dy(100) As Double
    Range("a5").Select
    n = ActiveCell.Row
    Selection.End(xlDown).Select
    n = ActiveCell.Row - n
    Range("a5").Select
    For i = 0 To n
        x(i) = ActiveCell.Value
        ActiveCell.Offset(0, 1).Select
        y(i) = ActiveCell.Value
        ActiveCell.Offset(1, -1).Select
    Next i
    For i = 0 To n
        dy(i) = DerivUnequal(x, y, n, x(i))
    Next i
    Range("c5").Select
    For i = 0 To n
        ActiveCell.Value = dy(i)
        ActiveCell.Offset(1, 0).Select
    Next i
    End Sub

Function DerivUnequal(x, y, n, xx)
    Dim ii As Integer
    If xx < x(0) Or xx > x(n) Then
        DerivUnequal = "out of range"
    Else
        If xx < x(1) Then
            DerivUnequal = DyDx(xx, x(0), x(1), x(2), y(0), y(1), y(2))
        ElseIf xx > x(n - 1) Then
            DerivUnequal =
                DyDx(xx, x(n - 2), x(n - 1), x(n), y(n - 2), y(n - 1), y(n))
        Else
            For ii = 1 To n - 2
                If xx >= x(ii) And xx <= x(ii + 1) Then
                    If xx - x(ii - 1) < x(ii) - xx Then
                        'If the unknown is closer to the lower end of the range,
                        'x(ii) will be chosen as the middle point
                        DerivUnequal =
                            DyDx(xx, x(ii - 1), x(ii), x(ii + 1), y(ii - 1), y(ii), y(ii + 1))
                    Else
                        'Otherwise, if the unknown is closer to the upper end,
                        'x(ii+1) will be chosen as the middle point
                        DerivUnequal =
                            DyDx(xx, x(ii), x(ii + 1), x(ii + 2), y(ii), y(ii + 1), y(ii + 2))
                    End If
                    Exit For
                End If
            Next ii
        End If
    End If
End Function

Function DyDx(x, x0, x1, x2, y0, y1, y2)

```

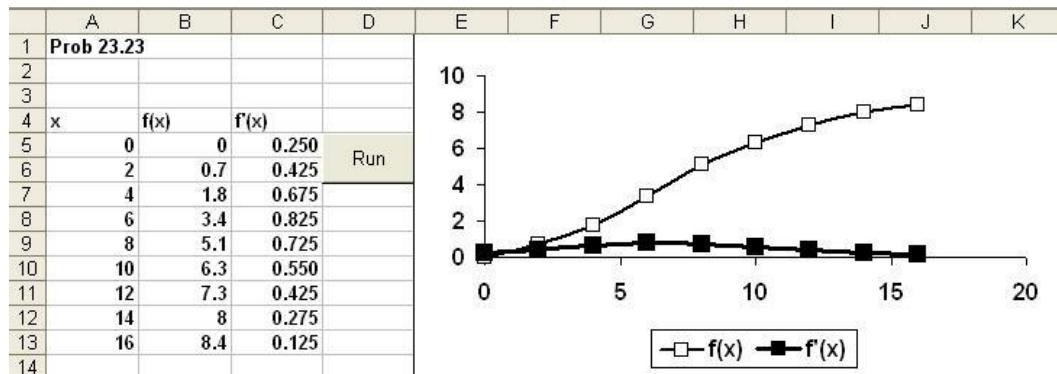
PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

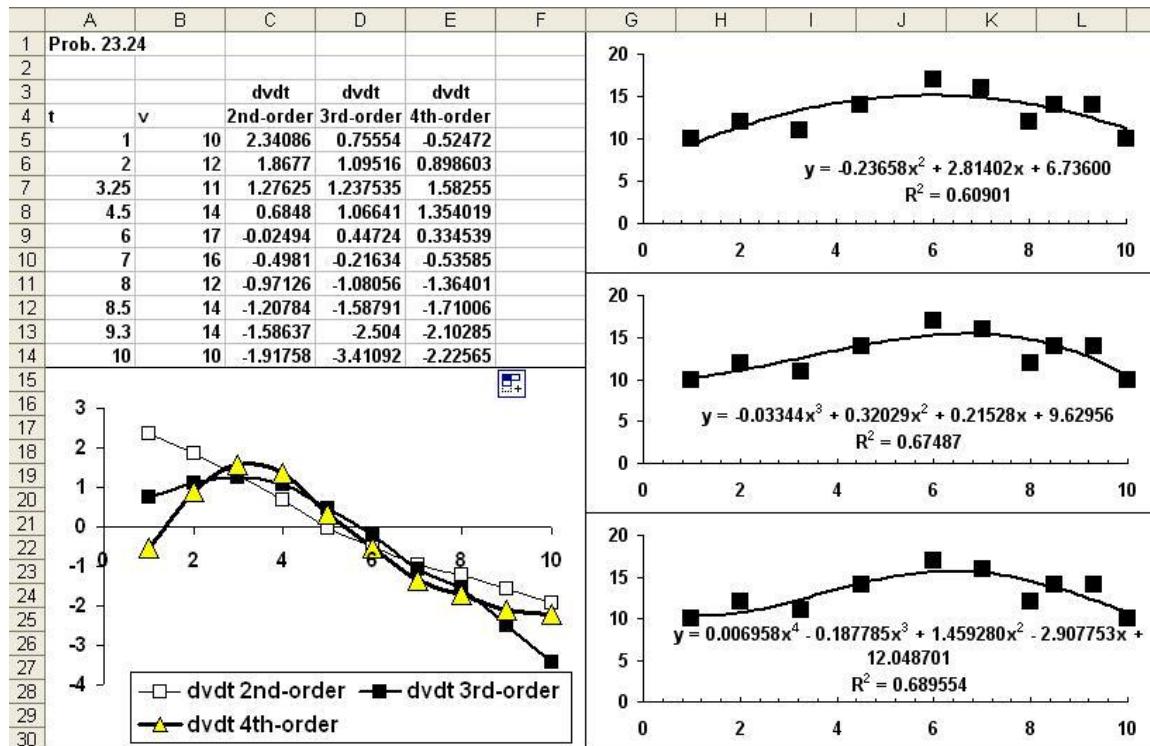
DyDx = y0 * (2 * x - x1 - x2) / (x0 - x1) / (x0 - x2) -
       + y1 * (2 * x - x0 - x2) / (x1 - x0) / (x1 - x2) -
       + y2 * (2 * x - x0 - x1) / (x2 - x0) / (x2 - x1)
End Function

```

The result of running this program is shown below:



23.24



23.25 The flow rate is equal to the derivative of volume with respect to time. Equation (23.9) can be used to compute the derivative as

$$\begin{aligned}
 x_0 &= 1 & f(x_0) &= 1 \\
 x_1 &= 5 & f(x_1) &= 8 \\
 x_2 &= 8 & f(x_2) &= 16.4
 \end{aligned}$$

$$f'(7) = 1 \frac{2(7) - 5 - 8}{(1-5)(1-8)} + 8 \frac{2(7) - 1 - 8}{(5-1)(5-8)} + 16.4 \frac{2(7) - 1 - 5}{(8-1)(8-5)} = 0.035714 - 3.33333 + 6.247619 = 2.95$$

Therefore, the flow is equal to $2.95 \text{ cm}^3/\text{s}$.

23.26 The velocity at the surface can be computed with Eq. (23.9) as

$$\begin{aligned} x_0 &= 0 & f(x_0) &= 0 \\ x_1 &= 0.002 & f(x_1) &= 0.287 \\ x_2 &= 0.006 & f(x_2) &= 0.899 \end{aligned}$$

$$\begin{aligned} f'(0) &= 0 \frac{2(0) - 0.002 - 0.006}{(0 - 0.002)(0 - 0.006)} + 0.287 \frac{2(0) - 0 - 0.006}{(0.002 - 0)(0.002 - 0.006)} + 0.899 \frac{2(0) - 0 - 0.002}{(0.006 - 0)(0.006 - 0.002)} \\ &= 0 + 215.25 - 74.9167 = 140.3333 \end{aligned}$$

Therefore, the shear stress can be computed as

$$\tau = 1.8 \times 10^{-5} \frac{\text{N} \cdot \text{s}}{\text{m}^2} 140.3333 \frac{1}{\text{s}} = 0.00253 \frac{\text{N}}{\text{m}^2}$$

23.27 The first forward difference formula of $O(h^2)$ from Fig. 23.1 can be used to estimate the velocity for the first point at $t = 10$,

$$\frac{dc}{dt}(10) = \frac{-1.75 + 4(2.48) - 3(3.52)}{2(10)} = -0.1195$$

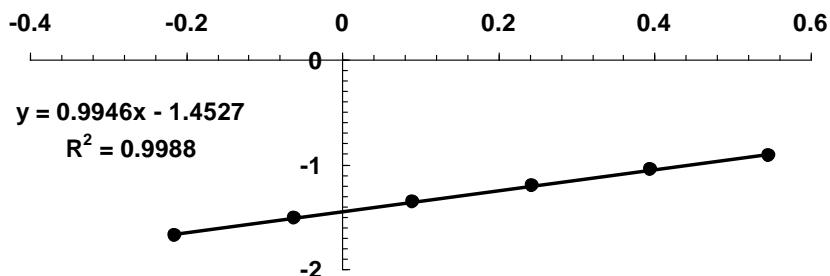
For the interior points, centered difference formulas of $O(h^2)$ from Fig. 23.3 can be used to estimate the derivatives. For example, at the second point at $t = 20$,

$$\frac{dc}{dt}(20) = \frac{1.75 - 3.52}{2(10)} = -0.0885$$

For the final point, backward difference formulas of $O(h^2)$ from Fig. 23.2 can be used to estimate the derivative. The results for all values are summarized in the following table.

<i>t</i>	<i>c</i>	<i>-dc/dt</i>	<i>log c</i>	<i>log(-dc/dt)</i>
10	3.52	0.1195	0.546543	-0.92263
20	2.48	0.0885	0.394452	-1.05306
30	1.75	0.0625	0.243038	-1.20412
40	1.23	0.044	0.089905	-1.35655
50	0.87	0.031	-0.06048	-1.50864
60	0.61	0.021	-0.21467	-1.67778

A log-log plot can be developed



The resulting best-fit equation can be used to compute $k = 10^{-1.45269} = 0.035262$ and $n = 0.994579$.

CHAPTER 24

24.1 Simpson's rules can be implemented as

$$\begin{aligned} I &= (-50 - (-150)) \frac{0.11454 + 4(0.11904) + 0.12486}{6} \\ &\quad + (100 - (-50)) \frac{0.12486 + 3(0.132 + 0.14046) + 0.15024}{8} \\ &= 11.926 + 20.484 = 32.41 \end{aligned}$$

The amount of heat required can be computed as

$$\Delta H = 32.41(1200) = 38892$$

24.2 The first iteration involves computing 1 and 2 segment trapezoidal rules and combining them as

$$I = \frac{4(32.581875) - 33.0975}{3} = 32.41$$

and computing the approximate error as

$$\varepsilon_a = \left| \frac{32.41 - 32.581875}{32.41} \right| \times 100\% = 0.5303\%$$

The computation can be continues as in the following tableau until $\varepsilon_a < 0.01\%$.

	1	2	3
<i>n</i>	$\varepsilon_a \rightarrow$	0.5303%	0.0000%
1	33.09750000	32.41000000	32.41000000
2	32.58187500	32.41000000	
4	32.45296875		

24.3 Change of variable:

$$x = \frac{100 - 150}{2} + \frac{100 - (-150)}{2} x_d = -25 + 125 x_d$$

$$dx = \frac{100 - (-150)}{2} dx_d = 125 dx_d$$

$$I = \int_{-1}^1 (0.132 + 1.56 \times 10^{-4}(-25 + 125 x_d) + 2.64 \times 10^{-7}(-25 + 125 x_d)^2) 125 dx_d$$

Therefore, the transformed function is

$$f(x_d) = 125(0.132 + 1.56 \times 10^{-4}(-25 + 125x_d) + 2.64 \times 10^{-7}(-25 + 125x_d)^2)$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 14.91679 + 17.49321 = 32.41$$

Three-point formula:

$$\begin{aligned} I &= 0.5555556 f(-0.7745967) + 0.8888889 f(0) + 0.5555556 f(0.7745967) \\ &= 0.5555556(14.61418) + 0.8888889(16.03313) + 0.5555556(18.07082) \\ &= 8.11899 + 14.25167 + 10.03934 = 32.41 \end{aligned}$$

Thus, both give exact results because the errors are

$$\begin{aligned} 2 \text{ point: } E_t &\propto f^{(4)}(\xi) \\ 3 \text{ point: } E_t &\propto f^{(6)}(\xi) \end{aligned}$$

which are zero for a second-order polynomial.

24.4 Simpson's rules can be implemented as

$$\begin{aligned} I &= (30 - 0) \frac{10 + 3(35 + 55) + 52}{8} + (40 - 30) \frac{52 + 4(40) + 37}{6} + (50 - 40) \frac{37 + 4(32) + 34}{6} \\ &= 1245 + 415 + 331.6667 = 1991.6667 \end{aligned}$$

The amount of mass can be computed as

$$M = 4 \frac{\text{m}^3}{\text{min}} \left(1991.6667 \frac{\text{mg min}}{\text{m}^3} \right) = 7,966.667 \text{ mg}$$

24.5 Newton-Cotes integration formulas can be implemented to evaluate the integral as

$$\begin{aligned} I &= (1 - 0) \frac{12 + 22}{2} + (4 - 1) \frac{22 + 32}{2} + (8 - 4) \frac{32 + 4(45) + 58}{6} + (20 - 8) \frac{58 + 3(75 + 70) + 48}{8} \\ &= 17 + 81 + 180 + 811.5 = 1089.5 \end{aligned}$$

The amount of mass in grams can be computed as

$$M = 0.3 \frac{\text{m}^3}{\text{s}} \left(1089.5 \frac{\text{mg min}}{\text{m}^3} \right) \frac{60 \text{ s}}{\text{min}} \frac{\text{g}}{1000 \text{ mg}} = 19.611 \text{ g}$$

24.6 Equation (23.9) can be used to compute the derivative as

$$\begin{array}{ll} x_0 = 0 & f(x_0) = 0.06 \\ x_1 = 1 & f(x_1) = 0.32 \\ x_2 = 3 & f(x_2) = 0.6 \end{array}$$

$$f'(0) = 0.06 \frac{2(0) - 1 - 3}{(0-1)(0-3)} + 0.32 \frac{2(0) - 0 - 3}{(1-0)(1-3)} + 0.6 \frac{2(0) - 0 - 1}{(3-0)(3-1)} = -0.08 + 0.48 - 0.1 = 0.3$$

The mass flux can be computed as

$$\text{Mass flux} = -1.52 \times 10^{-6} \frac{\text{cm}^2}{\text{s}} 0.3 \times 10^{-6} \frac{\text{g}}{\text{cm}^4} = -4.56 \times 10^{-13} \frac{\text{g}}{\text{cm}^2 \text{s}}$$

where the negative sign connotes transport from the sediments into the lake. The amount of mass transported into the lake can be computed as

$$\text{Mass transport} = 4.56 \times 10^{-13} \frac{\text{g}}{\text{cm}^2 \text{s}} (3.6 \times 10^6 \text{ m}^2) \frac{365 \text{ d}}{\text{yr}} \frac{86,400 \text{ s}}{\text{d}} \frac{10,000 \text{ cm}^2}{\text{m}^2} \frac{\text{kg}}{1000 \text{ g}} = 517.695 \text{ kg}$$

24.7 For the equally-spaced points, we can use the second-order formulas from Figs. 23.1 through 23.3. For example, for the first point ($t = 0$), we can use

$$f'(0) = \frac{-0.77 + 4(0.7) - 3(0.4)}{2(10)} = 0.0415 \frac{\text{barrels}}{\text{min}}$$

However, the points around $t = 30$ are unequally spaced so we must use Eq. (23.9) to compute the derivative as

$$\begin{array}{ll} x_0 = 20 & f(x_0) = 0.77 \\ x_1 = 30 & f(x_1) = 0.88 \\ x_2 = 45 & f(x_2) = 1.05 \end{array}$$

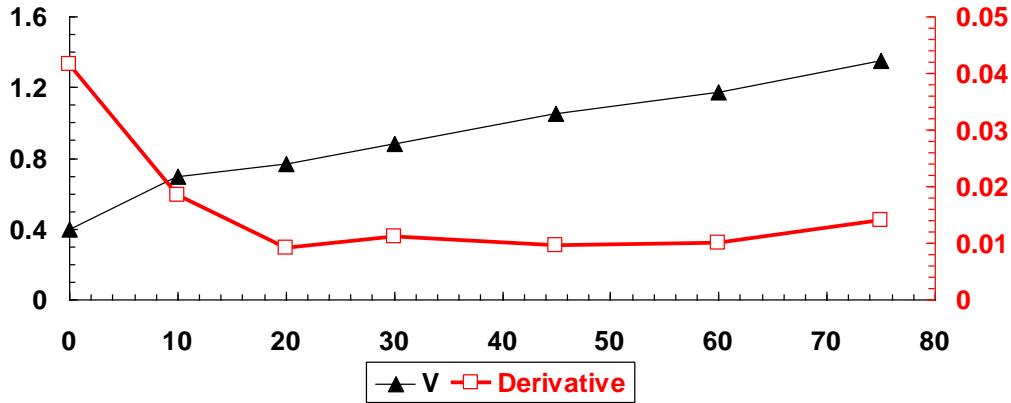
$$f'(30) = 0.77 \frac{2(30) - 30 - 45}{(20-30)(20-45)} + 0.88 \frac{2(30) - 20 - 45}{(30-20)(30-45)} + 1.05 \frac{2(30) - 20 - 30}{(45-20)(45-30)} = 0.011133$$

All the results can be summarized as

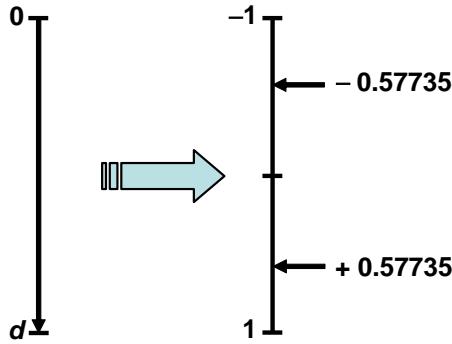
t	V	Derivative	Method
0	0.4	0.0415	Forward
10	0.7	0.0185	Centered
20	0.77	0.009	Centered
30	0.88	0.011133	Unequal
45	1.05	0.009667	Centered
60	1.17	0.01	Centered
75	1.35	0.014	Backward

The derivatives can be plotted versus time as

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.



24.8 If we use 2-point Gauss quadrature the depth domain can be mapped onto the $[-1, 1]$ domain as depicted here



This spacing corresponds to $21.1\%d$ and $78.9\%d$. Thus, if we wanted to get a good estimate, we would average the two samples taken at about 20% and 80% of the depth. It should be noted that the U.S. Geological Survey recommends this procedure for making flow measurements in rivers.

24.9 Because the data is equispaced, we can use the second-order finite divided difference formulas from Figs. 23.1 through 23.3. For the first point, we can use

$$\frac{d\sigma}{d\varepsilon} = \frac{-176 + 4(96.6) - 3(87.8)}{255 - 153} = -0.5196$$

For the intermediate points, we can use centered differences. For example, for the second point

$$\frac{d\sigma}{d\varepsilon} = \frac{176 - 87.8}{255 - 153} = 0.8647$$

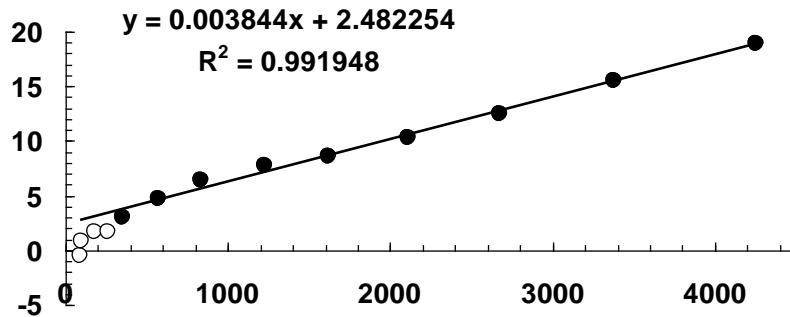
For the last point, we can use a backward difference

$$\frac{d\sigma}{d\varepsilon} = \frac{3(4258) - 4(3380) + 2678}{765 - 663} = 18.94118$$

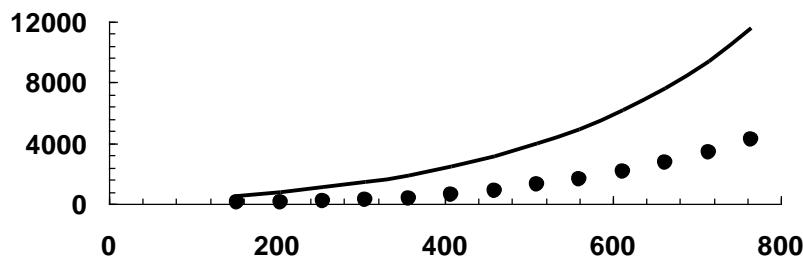
All the values can be tabulated as

σ	ε	$d\sigma/d\varepsilon$
87.8	153	-0.5196
96.6	204	0.8647
176	255	1.6314
263	306	1.7157
351	357	3.0196
571	408	4.7353
834	459	6.4510
1229	510	7.7451
1624	561	8.6078
2107	612	10.3333
2678	663	12.4804
3380	714	15.4902
4258	765	18.9412

We can plot these results and after discarding the first few points, we can fit a straight line as shown (note that the discarded points are displayed as open circles).



Therefore, the parameter estimates are $E_o = 2.48225$ and $a = 0.003844$. The data along with the first equation can be plotted as



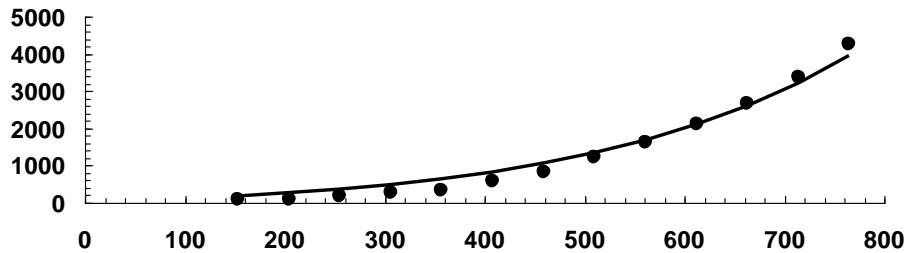
As described in the problem statement, the fit is not very good. We therefore pick a point near the midpoint of the data interval

$$(\bar{\varepsilon} = 612, \bar{\sigma} = 2107)$$

We can then plot the second model

$$\sigma = \left(\frac{2107}{e^{0.003844(612)} - 1} \right) (e^{0.003844\varepsilon} - 1) = 221.5719(e^{0.003844\varepsilon} - 1)$$

The result is a much better fit



24.10 An exponential model can be fit to the four points from $t = 17$ to 23 . The result is

$$c = 2132.9e^{-0.3277t}$$

This equation can then be used to generate data from $t = 25$ through 35 . This synthetic data along with the previous measured data can then be integrated with the trapezoidal rule,

$$I = (7 - 5) \frac{0 + 0.1}{2} + (9 - 7) \frac{0.1 + 0.11}{2} + \dots + (33 - 31) \frac{0.0826 + 0.0429}{2} + (35 - 33) \frac{0.0429 + 0.0223}{2} = 61.2049$$

All the evaluations are summarized in the following table,

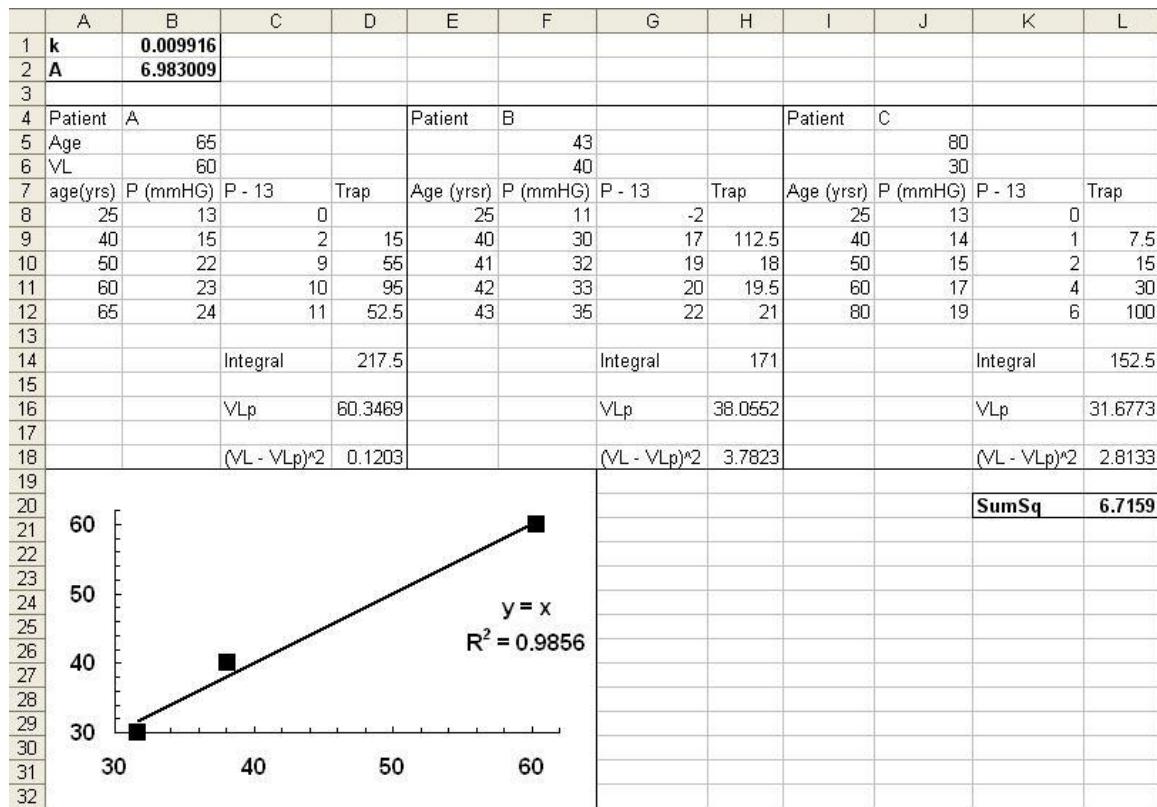
t	c , original	c , synthetic	Trap rule
5	0	0	
7	0.1	0.1	0.1
9	0.11	0.11	0.21
11	0.4	0.4	0.51
13	4.1	4.1	4.5
15	9.1	9.1	13.2
17	8	8	17.1
19	4.2	4.2	12.2
21	2.3	2.3	6.5
23	1.1	1.1	3.4

25	0.9	0.5902	1.6902
27	1.75	0.3065	0.8967
29	2.06	0.1591	0.4656
31	2.25	0.0826	0.2417
33	2.32	0.0429	0.1255
35	2.43	0.0223	0.0652

The cardiac output can then be computed as

$$C = \frac{5.6}{61.2049} 60 = 5.48975 \frac{\text{L}}{\text{min}}$$

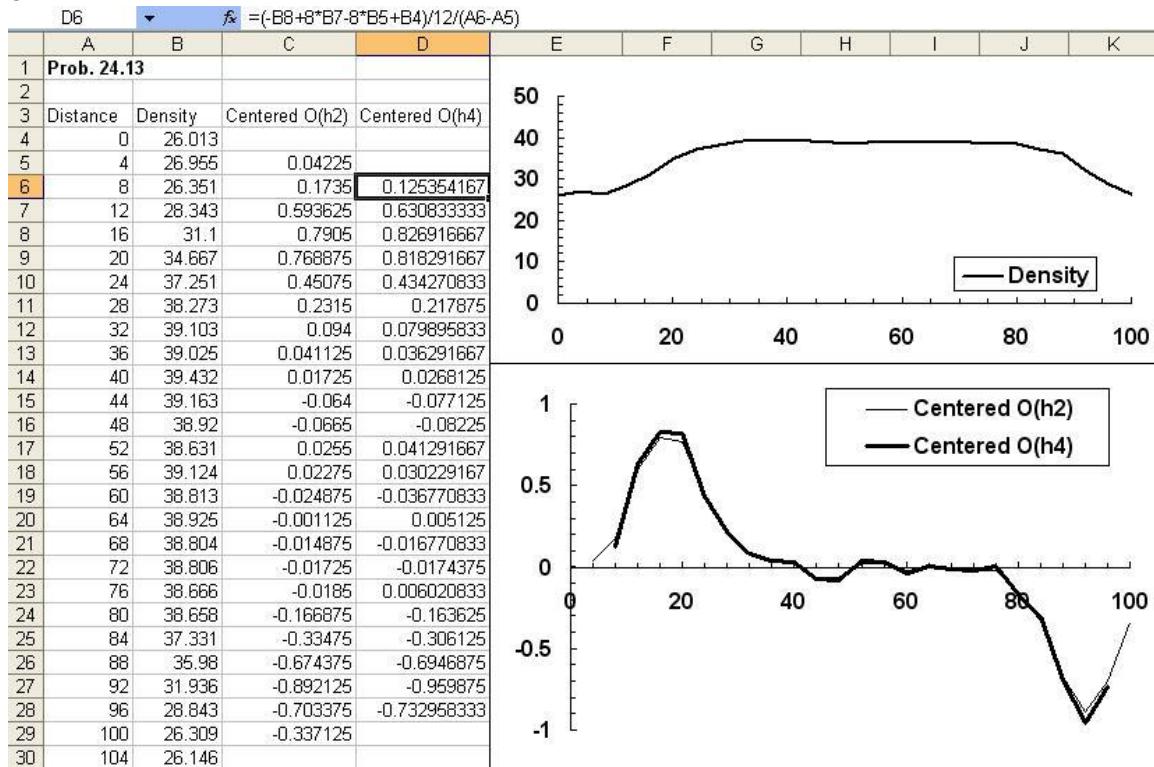
24.11 The following Excel Solver application can be used to estimate: $k = 0.09916$ and $A = 6.98301$.



24.12 The following Excel spreadsheet is set up to use (left) a combination of the trapezoidal and Simpsons rules and (right) just the trapezoidal rule:

	A	B	C	D	E	F	G	H
1	Prob 24.12							
2								
3	Area	12 cm ²						
4								
5	Simp1/3-3/8-Trap				Trap			
6	Time	Flux	Integral	Rule	Time	Flux	Trapezoidal	
7	0	15			0	15		
8	1	14			1	14	14.5	
9	2	12	27.66667	1/3 rule	2	12	13	
10	3	11			3	11	11.5	
11	4	9			4	9	10	
12	5	8	30	3/8 rule	5	8	8.5	
13	10	5			10	5	32.5	
14	15	2.5			15	2.5	18.75	
15	20	2	60.9375	3/8 rule	20	2	11.25	
16	24	1	6	Trap rule	24	1	6	
17								
18		Integral	124.6042				111.5	
19								
20		Average	5.19184			Average	4.64583333	
21								
22	Mass delivered	1495.25					1338	

24.13



The extremes for both cases are at 16 and 92

24.14

$$I = \int_0^{30} 200z \left[\frac{z}{5+z} \right] e^{-z/15} dz$$

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

n	1 $\varepsilon_a \rightarrow$	2	3	4
1	10440.1504	11.9724%	0.1646%	0.01412%
2	17636.0064	19552.6915	19348.4694	
4	19073.5202	19361.2333		
8	19289.3051			

$$I = \int_0^{30} 200 \left[\frac{z}{5+z} \right] e^{-z/15} dz$$

n	1 $\varepsilon_a \rightarrow$	2	3	4
1	348.0050	1219.6400	1440.6846	1476.7973
2	1001.7312	1426.8693	1476.2330	
4	1320.5848	1473.1478		
8	1435.0070			

$$d = \frac{19345.74}{1476.797} = 13.0998$$

$$V = \frac{1476.797(13.0998)}{3} = 6448.58$$

$$T = \frac{6448.58}{0.995} = 6480.985$$

$$H = 1476.797 - 6480.985(0.0995) = 831.94$$

24.15 Change of variable:

$$x = \frac{30+0}{2} + \frac{30-0}{2} x_d = 15 + 15x_d$$

$$dx = \frac{30-0}{2} dx_d = 15 dx_d$$

$$I = \int_0^{30} 200z \left[\frac{z}{5+z} \right] e^{-z/15} dz$$

$$I = \int_{-1}^1 200(15 + 15x_d) \left[\frac{15 + 15x_d}{5 + (15 + 15x_d)} \right] e^{-(15+15x_d)/15} 15 dx_d$$

Therefore, the transformed function is

$$f(x_d) = 3000 \frac{(15+15x_d)^2}{20+15x_d} e^{-(1+x_d)}$$

Five-point formula:

$$\begin{aligned} I &= 0.236927 f(-0.90618) + 0.478629 f(-0.53847) + 0.568889 f(0) \\ &\quad + 0.478629 f(0.53847) + 0.236927 f(0.90618) \\ &= 0.236927(844.2582) + 0.478629(7601.173) + 0.568889(12415.93) \\ &\quad + 0.478629(12217.47) + 0.236927(10852.84) = 19,320.41 \end{aligned}$$

$$I = \int_0^{30} 200 \left[\frac{z}{5+z} \right] e^{-z/15} dz$$

$$I = \int_{-1}^1 200 \left[\frac{15+15x_d}{5+(15+15x_d)} \right] e^{-(15+15x_d)/15} 15 dx_d$$

Therefore, the transformed function is

$$f(x_d) = 3000 \frac{15+15x_d}{20+15x_d} e^{-(1+x_d)}$$

Five-point formula:

$$\begin{aligned} I &= 0.236927 f(-0.90618) + 0.478629 f(-0.53847) + 0.568889 f(0) \\ &\quad + 0.478629 f(0.53847) + 0.236927 f(0.90618) \\ &= 0.236927(599.9124) + 0.478629(1097.966) + 0.568889(827.7287) \\ &\quad + 0.478629(529.4212) + 0.236927(379.5667) = 1,481.865 \end{aligned}$$

$$d = \frac{19320.41}{1481.865} = 13.038$$

$$V = \frac{1481.865(13.038)}{3} = 6440.185$$

$$T = \frac{6440.185}{0.995} = 6472.548$$

$$H = 1481.865 - 6472.548(0.0995) = 837.846$$

24.16 Trapezoidal rule:

$$I = (30 - 0) \frac{0 + 2(68.924 + 57.481 + 39.845 + 26.026 + 16.549) + 10.372}{2(6)} = 1070.057$$

Simpson's 1/3 rule:

$$I = (30 - 0) \frac{0 + 4(68.924 + 39.845 + 16.549) + 2(57.481 + 26.026) + 10.372}{3(6)} = 1131.098$$

Simpson's 3/8 rule:

$$I = (15 - 0) \frac{0 + 3(68.924 + 57.481) + 39.845}{8} + (30 - 15) \frac{39.845 + 3(26.026 + 16.549) + 10.372}{8} = 785.7375 + 333.6435 = 1119.381$$

24.17 Trapezoidal rule ($h = 4$):

$$I = (20 - 0) \frac{0 + 2(2 + 4 + 4 + 3.4) + 0}{10} = 53.6 \text{ m}^2$$

Trapezoidal rule ($h = 2$):

$$I = (20 - 0) \frac{0 + 2(1.8 + 2 + 4 + 4 + 6 + 4 + 3.6 + 3.4 + 2.8) + 0}{20} = 63.2 \text{ m}^2$$

Simpson's 1/3 rule:

$$I = (20 - 0) \frac{0 + 4(1.8 + 4 + 6 + 3.6 + 2.8) + 2(2 + 4 + 4 + 3.4) + 0}{30} = 66.4 \text{ m}^2$$

24.18 A table can be set up to hold the values that are to be integrated:

y, m	H, m	U, m/s	UH, m²/s
0	0.5	0.03	0.015
2	1.3	0.06	0.078
4	1.25	0.05	0.0625
5	1.7	0.12	0.204
6	1	0.11	0.11
9	0.25	0.02	0.005

The cross-sectional area can be evaluated using a combination of Simpson's 1/3 rule and the trapezoidal rule:

$$A_c = (4 - 0) \frac{0.5 + 4(1.3) + 1.25}{6} + (6 - 4) \frac{1.25 + 4(1.7) + 1}{6} + (9 - 6) \frac{1 + 0.25}{2} \\ = 4.633333 + 3.016667 + 1.875 = 9.525 \text{ m}^2$$

The flow can be evaluated in a similar fashion:

$$\begin{aligned}
 Q &= (4 - 0) \frac{0.015 + 4(0.078) + 0.0625}{6} + (6 - 4) \frac{0.0625 + 4(0.204) + 0.11}{6} + (9 - 6) \frac{0.11 + 0.005}{2} \\
 &= 0.259667 + 0.3295 + 0.1725 = 0.761667 \frac{\text{m}^3}{\text{s}}
 \end{aligned}$$

24.19 A table can be set up to hold the values that are to be integrated:

Abscissa (N to S)	Ordinate (W to E)	Abscissa (N to S)	Ordinate (W to E)
0	0	2200	3150
200	600	2400	3200
400	880	2600	3200
600	950	2800	3150
800	1100	3000	3000
1000	1600	3200	2800
1200	1900	3400	2700
1400	2100	3600	2600
1600	2300	3800	2525
1800	2600	4000	2525
2000	2950	4200	2430

We can use the multi-segment Simpson's 1/3 rule for the first 18 segments and the Simpson's 3/8 rule for the last 3 segments:

$$\begin{aligned}
 I &= (3600 - 0)[0 + 4(600 + 950 + 1600 + 2100 + 2600 + 3150 + 3200 + 3000 + 2700) \\
 &\quad + 2(880 + 1100 + 1900 + 2300 + 2950 + 3200 + 3150 + 2800) + 2600]/54 = 7,917,333.33
 \end{aligned}$$

$$I = (4200 - 3600) \frac{2600 + 3(2525 + 2525) + 2430}{8} = 1,513,500$$

Therefore, the total area is

$$A = 7,917,333.33 + 1,513,500 = 9,430,833 \text{ ft}^2$$

24.20 In order to generate the average rate in units of car/min, the integral to be evaluated along with the units is

$$\text{averagerate} = \frac{\int_0^{24 \text{ hr}} \text{rate(car/min)} dt}{24 \text{ hr}}$$

The integral can be evaluated with the trapezoidal and Simpson's rules as tabulated below:

Time (hr)	Rate (car/min)	Method	Integral
0	2		
2	2		

4	0	Simp 1/3	6.666667
5	2		
6	6	Simp 1/3	4.666667
7	7		
8	23		
9	11	Simp 3/8	40.125
10.5	4	Trap	11.25
11.5	11		
12.5	12	Simp 1/3	20
14	8	Trap	15
16	7	Trap	15
17	26		
18	20	Simp 1/3	43.66667
19	10		
20	8	Simp 1/3	22.66667
21	10		
22	8	Simp 1/3	18.66667
23	7		
24	3	Simp 1/3	13

The summation of the last column yields the integral estimate of 210.7083 cars·hr/min.
 Dividing by 24 hrs yields the average rate of $210.7083/24 = 8.779514$ cars/min. This can then be multiplied by 1,440 min/d to yield 12,642.5 cars/d.

24.21 The values needed to perform the evaluation can be tabulated:

Height <i>I</i> , m	Force, <i>F(I)</i> , N/m	<i>IxF(I)</i>
0	0	0
30	340	10200
60	1200	72000
90	1600	144000
120	2700	324000
150	3100	465000
180	3200	576000
210	3500	735000
240	3800	912000

Because there are an even number of equally-spaced segments, we can evaluate the integrals with the multi-segment Simpson's 1/3 rule.

$$F = (240 - 0) \frac{0 + 4(340 + 1600 + 3100 + 3500) + 2(1200 + 2700 + 3200) + 3800}{24} = 521,600$$

$$I = (240 - 0) \frac{0 + 4(10200 + 144000 + 465000 + 735000) + 2(72000 + 324000 + 576000) + 912000}{24} = 82,728,000$$

The line of action can therefore be computed as

$$d = \frac{82,728,000}{521,600} = 158.6043$$

24.22 The values needed to perform the evaluation can be tabulated:

<i>z</i>	<i>w(z)</i>	<i>ρgw(z)(60-z)</i>	<i>zpgw(z)(60-z)</i>
60	200	0	0
50	190	1.862E+07	9.310E+08
40	175	3.430E+07	1.372E+09
30	160	4.704E+07	1.411E+09
20	135	5.292E+07	1.058E+09
10	130	6.370E+07	6.370E+08
0	122	7.174E+07	0

Because there are an even number of equally-spaced segments, we can evaluate the required integrals with the multi-segment Simpson's 1/3 rule.

$$f_t = (60 - 0) \frac{7.174 + 4(6.37 + 4.704 + 1.862) + 2(5.292 + 3.43) + 0}{18} \times 10^7 = 2.54539 \times 10^9$$

$$f_t = (60 - 0) \frac{0 + 4(6.371 + 4.112 + 9.31) + 2(10.584 + 13.72) + 0}{18} \times 10^8 = 5.59253 \times 10^{10}$$

The line of action can therefore be computed as

$$d = \frac{5.59253 \times 10^{10}}{2.54539 \times 10^9} = 21.97 \text{ m}$$

24.23 One way to determine the volume is to integrate the data from mid-Jan through mid-Jan by duplicating the mid-Jan value at the end of the record. Since we are dealing with long-term averages, this is valid. Further, for simplicity, we can assume that every month is 30 days long. We can then integrate the data. As shown in the following table, we have used the a combination of the trapezoidal and Simpson's rules.

day	Flow	Trap/Simp	Method
15	30		
45	38		
75	82		
105	125	5793.75	Simp 3/8
165	95	6600	Trap
255	20	5175	Trap
285	22		
315	24	1320	Simp 1/3
345	35		

375 30 1940 Simp 1/3

The summation of the integral estimates is $20,828.75 \text{ m}^3\cdot\text{d/s}$. We can divide this quantity by the integration interval (360 d) to yield an average flow of $57.8576 \text{ m}^3/\text{s}$. This quantity can then be multiplied by the number of seconds per year to arrive at the volume of water that flows down the river over a typical year.

$$\text{Annual Flow} = 57.8576 \times 60 \frac{\text{s}}{\text{min}} \times 60 \frac{\text{min}}{\text{hr}} \times 24 \frac{\text{hr}}{\text{d}} \times 365 \frac{\text{d}}{\text{yr}} = 1.8246 \times 10^9 \text{ m}^3$$

24.24 The integral to be evaluated is

$$h = e_{ab} A \int_0^t q dt$$

Simpson's rules can be used to evaluate the integral,

$$I = (8 - 0) \frac{0.1 + 4(5.32 + 8) + 2(7.8) + 8.03}{12} + (14 - 8) \frac{8.03 + 3(6.27 + 3.54) + 0.2}{8} \\ = 51.34 + 28.245 = 79.585$$

We can then multiply this result by the area and the efficiency to give

$$h = 0.45(100,000 \text{ cm}^2)79.585 \text{ cal/cm}^2 = 5,371,988 \text{ cal}$$

24.25 Because the values are equally spaced, we can use the second-order forward difference from Fig. 23.1 to compute the derivative as

$$\begin{aligned} x_0 &= 0 & f(x_0) &= 20 \\ x_1 &= 0.08 & f(x_1) &= 17 \\ x_2 &= 0.16 & f(x_2) &= 15 \end{aligned}$$

$$f'(0) = \frac{-15 + 4(17) - 3(20)}{0.16} = -43.75 \frac{\text{°C}}{\text{m}}$$

The coefficient of thermal conductivity can then be estimated as $k = -60 \text{ W/m}^2/(-43.75 \text{ °C/m}) = 1.37143 \text{ W/(°C·m)}$.

24.26 First, we can estimate the areas by numerically differentiating the volume data. Because the values are equally spaced, we can use the second-order difference formulas from Fig. 23.1 to compute the derivatives at each depth. For example, at the first depth, we can use the forward difference to compute

$$A_s(0) = -\frac{dV}{dz}(0) = -\frac{-1,963,500 + 4(5,105,100) - 3(9,817,500)}{8} = 1,374,450 \text{ m}^2$$

For the interior points, second-order centered differences can be used. For example, at the second point at ($z = 4$ m),

$$A_s(4) = -\frac{dV}{dz}(4) = -\frac{1,963,500 - 9,817,500}{8} = 981,750 \text{ m}^2$$

The other interior points can be determined in a similar fashion

$$A_s(8) = -\frac{dV}{dz}(8) = -\frac{392,700 - 5,105,100}{8} = 589,050 \text{ m}^2$$

$$A_s(12) = -\frac{dV}{dz}(12) = -\frac{0 - 1,963,500}{8} = 245,437.5 \text{ m}^2$$

For the last point, the second-order backward formula yields

$$A_s(16) = -\frac{dV}{dz}(16) = -\frac{3(0) - 4(392,700) + 1,963,500}{8} = -49,087.5 \text{ m}^2$$

Since this is clearly a physically unrealistic result, we will assume that the bottom area is 0. The results are summarized in the following table along with the other quantities needed to determine the average concentration.

z, m	V, m^3	$c, \text{g/m}^3$	A_s, m^2	$c \times A_s$
0	9817500	10.2	1374450.0	14019390
4	5105100	8.5	981750.0	8344875
8	1963500	7.4	589050.0	4358970
12	392700	5.2	245437.5	1276275
16	0	4.1	0	0

The necessary integrals can then be evaluated with the multi-segment Simpson's 1/3 rule,

$$\int_0^z A_s(z) dz = (16 - 0) \frac{1,374,450 + 4(981,750 + 245,437.5) + 2(589,050) + 0}{12} = 9,948,400 \text{ m}^3$$

$$\int_0^z c(z) A_s(z) dz = (16 - 0) \frac{14,019,390 + 4(8,344,875 + 1,276,275) + 2(4,358,970) + 0}{12} = 81,629,240 \text{ g}$$

The average concentration can then be computed as

$$\bar{c} = \frac{\int_0^z c(z) A_s(z) dz}{\int_0^z A_s(z) dz} = \frac{81,629,240}{9,948,400} = 8.205263 \frac{\text{g}}{\text{m}^3}$$

24.27 The integral to be determined is

$$I = \int_0^{1/2} (5e^{-1.25t} \sin 2\pi t)^2 dt$$

Change of variable:

$$x = \frac{0.5+0}{2} + \frac{0.5-0}{2} x_d = 0.25 + 0.25x_d$$

$$dx = \frac{0.5-0}{2} dx_d = 0.25 dx_d$$

$$I = \int_{-1}^1 (5e^{-1.25(0.25+0.25x_d)} \sin 2\pi(0.25 + 0.25x_d))^2 0.25 dx_d$$

Therefore, the transformed function is

$$f(x_d) = 0.25 (5e^{-1.25(0.25+0.25x_d)} \sin 2\pi(0.25 + 0.25x_d))^2$$

Five-point formula:

$$\begin{aligned} I &= 0.236927 f(-0.90618) + 0.478629 f(-0.53847) + 0.568889 f(0) \\ &\quad + 0.478629 f(0.53847) + 0.236927 f(0.90618) \\ &= 0.236927(0.127087) + 0.478629(2.059589) + 0.568889(3.345384) \\ &\quad + 0.478629(1.050662) + 0.236927(0.040941) = 3.431617 \end{aligned}$$

Therefore, the RMS current can be computed as

$$I_{\text{RMS}} = \sqrt{3.431617} = 1.852463$$

24.28 The integral to be determined is

$$I = \int_0^{1/2} (5e^{-1.25t} \sin 2\pi t)^2 dt$$

Five applications of Simpson's 1/3 rule can be applied with $h = 0.05$ as tabulated below:

t	f²	Simpsons 1/3 rule
0	0	
0.05	2.106774	
0.1	6.726726	0.252563698
0.15	11.24592	
0.2	13.7153	1.090428284
0.25	13.38154	
0.3	10.68149	1.298715588
0.35	6.820993	
0.4	3.177481	0.685715716
0.45	0.775039	

0.5	0	<u>0.104627262</u>
	Sum →	<u>3.432050547</u>

Therefore, the RMS current can be computed as

$$I_{\text{RMS}} = \sqrt{3.432051} = 1.852579$$

24.29

$$I = \int_0^{1/2} (5e^{-1.25t} \sin 2\pi t)^2 dt$$

n	1 $\varepsilon_a \rightarrow$	2	3	4
1	0.00000000	4.46051190	3.38814962	3.43184278
2	3.34538393	3.45517226	3.43116008	
4	3.42772518	3.43266084		
8	3.43142692			

Therefore, the RMS current can be computed as

$$I_{\text{RMS}} = \sqrt{3.431843} = 1.852523$$

24.30 For the equispaced data, we can use the second-order finite divided difference formulas from Figs. 23.1 through 23.3. For example, for the first point, we can use

$$\frac{di}{dt} = \frac{-0.32 + 4(0.16) - 3(0)}{0.2 - 0} = 1.6$$

For the intermediate equispaced points, we can use centered differences. For example, for the second point

$$\frac{di}{dt} = \frac{0.32 - 0}{0.2 - 0} = 1.6$$

For the last point, we can use a backward difference

$$\frac{di}{dt} = \frac{3(2) - 4(0.84) + 0.56}{0.7 - 0.3} = 8$$

One of the points ($t = 0.3$) has unequally spaced neighbors. For this point, we can use Eq. (23.9) to compute the derivative as

$$\begin{aligned} x_0 &= 0.2 & f(x_0) &= 0.32 \\ x_1 &= 0.3 & f(x_1) &= 0.56 \\ x_2 &= 0.5 & f(x_2) &= 0.84 \end{aligned}$$

$$f'(0) = 0.32 \frac{2(0.3) - 0.3 - 0.5}{(0.2 - 0.3)(0.2 - 0.5)} + 0.56 \frac{2(0.3) - 0.2 - 0.5}{(0.3 - 0.2)(0.3 - 0.5)} + 0.84 \frac{2(0.3) - 0.2 - 0.3}{(0.5 - 0.2)(0.5 - 0.3)} = 2.066667$$

We can then multiply these derivative estimates by the inductance. All the results are summarized below:

<i>t</i>	<i>i</i>	<i>di/dt</i>	<i>V_L</i>
0	0	1.6	6.4
0.1	0.16	1.6	6.4
0.2	0.32	2	8
0.3	0.56	2.066667	8.266667
0.5	0.84	3.6	14.4
0.7	2	8	32

24.31 We can solve Faraday's law for inductance as

$$L = \frac{\int_0^t V_L dt}{i}$$

We can evaluate the integral using a combination of the trapezoidal and Simpson's rules,

$$\begin{aligned} I &= (20 - 0) \frac{0 + 4(18) + 29}{6} + (80 - 20) \frac{29 + 3(44 + 49) + 46}{8} + (120 - 80) \frac{46 + 35}{2} \\ &\quad + (180 - 120) \frac{35 + 26}{2} + (280 - 180) \frac{26 + 15}{2} + (400 - 280) \frac{15 + 7}{2} \\ &= 336.6667 + 2655 + 1620 + 1830 + 2050 + 1320 = 9811.667 \end{aligned}$$

The inductance can then be computed as

$$L = \frac{9811.667}{2} = 4905.833 \frac{\text{volts ms}}{\text{A}} \frac{\text{s}}{1000 \text{ ms}} = 4.905833 \text{ H}$$

24.32 The average voltage can be computed as

$$\bar{V} = \frac{\int_0^{60} i(t)R(i) dt}{60}$$

We can use the formulas to generate values of $i(t)$ and $R(i)$ and their product for various equally-spaced times over the integration interval as summarized in the table below. The last column shows the integral of the product as calculated with Simpson's 1/3 rule.

<i>t</i>	<i>i(t)</i>	<i>R(i)</i>	<i>i(t) × R(i)</i>	Simpson's 1/3
0	3600.000	43669.784	157211223	

6	2950.461	35816.950	105676498	1287149498
12	2288.787	27812.788	63657535	
18	1726.549	21006.431	36268640	456192829
24	1260.625	15360.891	19364321	
30	878.355	10723.694	9419212	122017054
36	569.294	6968.908	3967357	
42	327.533	4025.426	1318460	19048340
48	151.215	1871.341	282974	
54	41.250	518.872	21403	737174
60	0.000	0.000	0	
		Sum →		1885144894

The average voltage can therefore be computed as

$$\bar{V} = \frac{1885144894}{60} = 3.1419 \times 10^7$$

24.33 We can use the trapezoidal rule to integrate the data. For example at $t = 0.2$, the voltage is computed as

$$V(0.2) = \frac{1}{10^{-5}} (0.2 - 0) \frac{0.0002 + 0.0003683}{2} = 5.683$$

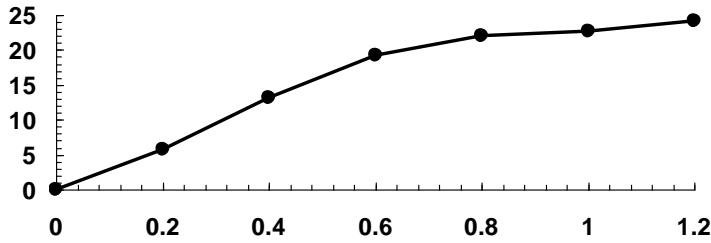
At $t = 0.4$, we add this value to the integral from $t = 0.2$ to 0.4 ,

$$V(0.4) = 5.683 + \frac{1}{10^{-5}} (0.4 - 0.2) \frac{0.0003683 + 0.0003819}{2} = 13.185$$

The remainder of the values can be computed in a similar fashion,

<i>t, s</i>	<i>i, A</i>	Trap	<i>V(t)</i>
0	0.0002000	0	0
0.2	0.0003683	5.683	5.683
0.4	0.0003819	7.502	13.185
0.6	0.0002282	6.101	19.286
0.8	0.0000486	2.768	22.054
1	0.0000082	0.568	22.622
1.2	0.0001441	1.523	24.145

The plot of voltage versus time can be developed as



24.34 The work can be computed as

$$W = \int_0^{30} (1.6x - 0.045x^2) \cos \theta \, dx$$

We can use the formulas to generate values of the integrand as summarized in the table below. The last column shows the integral as calculated with Simpson's 1/3 rule.

x	$F(x)$	$\theta(x)$	$F(x)\cos\theta(x)$	Simpson's 1/3
0	0	0.5	0	
5	6.875	1.4	1.168524107	21.81420
10	11.5	0.75	8.414421992	
15	13.875	0.9	8.62483831	77.76460
20	14	1.3	3.744983601	
25	11.875	1.48	1.076725541	14.30402
30	7.5	1.5	0.530529013	
Sum →				113.88282

24.35 The work can be computed as

$$W = \int_0^{30} (1.6x - 0.045x^2) \cos(0.8 + 0.125x - 0.009x^2 + 0.0002x^3) \, dx$$

For the 4-segment trapezoidal rule, we can compute values of the integrand at equally-spaced values of x with $h = 7.5$. The results are summarized in the following table,

x	$F(x)$	$\theta(x)$	$F(x)\cos\theta(x)$	Trap Rule
0	0	0.8	0	
7.5	9.46875	1.315625	2.390018	8.962569
15	13.875	1.325	3.376187	21.623271
22.5	13.21875	1.334375	3.096162	24.271308
30	7.5	1.85	-2.066927	3.859631
Sum →				58.716779

The finer-segment versions can be generated in a similar fashion. The results are summarized below:

Segments	W
4	58.71678

8	64.89955
16	66.41926

24.36 The work can be computed as

$$W = \int_0^{30} (1.6x - 0.045x^2) \cos(0.8 + 0.125x - 0.009x^2 + 0.0002x^3) dx$$

For the 4-segment Simpson's 1/3 rule, we can compute values of the integrand at equally-spaced values of x with $h = 7.5$. The results are summarized in the following table,

x	$F(x)$	$\theta(x)$	$F(x)\cos\theta(x)$	Simpson's 1/3 Rule
0	0	0.8	0	
7.5	9.46875	1.315625	2.390018	32.34065
15	13.875	1.325	3.376187	
22.5	13.21875	1.334375	3.096162	34.23477
30	7.5	1.85	-2.066927	
			Sum →	66.57542

The finer-segment versions can be generated in a similar fashion. The results are summarized below:

Segments	W
4	66.57542
8	66.96047
16	66.92583

24.37

$$W = \int_0^{30} (1.6x - 0.045x^2) \cos(0.8 + 0.125x - 0.009x^2 + 0.0002x^3) dx$$

n	1 $\epsilon_a \rightarrow$	2	3	4
1	-31.003903	38.5532%	0.9312%	0.0051%
2	35.140854	57.189106	67.201176	66.982729
4	58.716779	66.575421	66.986143	
8	64.899549	66.960473		

24.38 The function to be integrated is

$$W = \int_0^{30} (1.6x - 0.045x^2) \cos(0.8 + 0.125x - 0.009x^2 + 0.0002x^3) dx$$

Change of variable:

$$x = \frac{30+0}{2} + \frac{30-0}{2} x_d = 15 + 15x_d$$

$$dx = \frac{30 - 0}{2} dx_d = 15 dx_d$$

Therefore, the transformed function is

$$\begin{aligned} f(x_d) &= 15(1.6(15 + 15x_d) - 0.045(15 + 15x_d)^2) \\ &\quad \times \cos(0.8 + 0.125(15 + 15x_d) - 0.009(15 + 15x_d)^2 + 0.0002(15 + 15x_d)^3) \end{aligned}$$

Two-point formula:

$$I = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = 35.64278 + 38.20683 = 73.84961$$

Three-point formula:

$$\begin{aligned} I &= 0.5555556 f(-0.7745967) + 0.8888889 f(0) + 0.5555556 f(0.7745967) \\ &= 0.5555556(31.49657) + 0.8888889(50.64281) + 0.5555556(7.748426) \\ &= 17.4981 + 45.01583 + 4.304681 = 66.8186 \end{aligned}$$

The results for the 2- through 6-point formulas are summarized below:

points	<i>W</i>
2	73.8496
3	66.8186
4	66.7734
5	66.9145
6	66.9225

24.39 The work is computed as the product of the force times the distance, where the latter can be determined by integrating the velocity data (recall Eq. PT6.5),

$$W = F \int_0^t v(t) dt$$

A table can be set up holding the velocities at evenly spaced times over the integration interval. The Simpson's 1/3 rule can then be used to integrate this data as shown in the last column of the table

<i>t</i>	<i>v</i>	Simp 1/3 rule
0	0	
1	4	8
2	8	
3	12	24
4	16	
5	17	34.6667

6	20	
7	25	50.6667
8	32	
9	41	82.6667
10	52	
11	65	130.6667
12	80	
13	97	194.6667
14	116	
Sum →		525.3333

Thus, the work can be computed as

$$W = 200 \text{ N}(525.3333 \text{ m}) = 105,066.7 \text{ N} \cdot \text{m}$$

24.40 Because the data is equispaced, we can use the second-order finite divided difference formulas from Figs. 23.1 through 23.3. For the first point, we can use

$$\frac{dT}{dt} = \frac{-30 + 4(44.5) - 3(80)}{10 - 0} = -9.2$$

For the intermediate points, we can use centered differences. For example, for the second point

$$\frac{dT}{dt} = \frac{30 - 80}{10 - 0} = -5$$

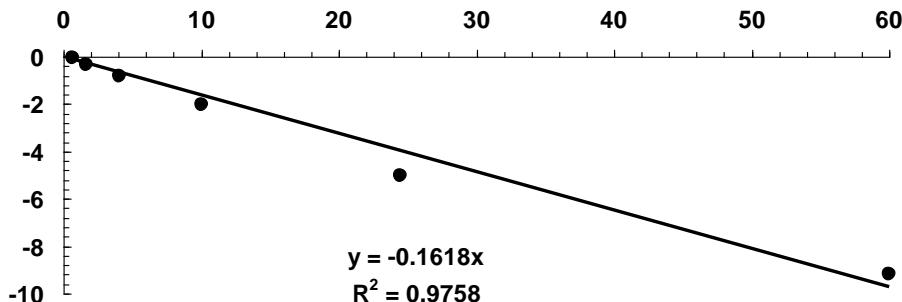
We can analyze the other interior points in a similar fashion. For the last point, we can use a backward difference

$$\frac{dT}{dt} = \frac{3(20.7) - 4(21.7) + 24.1}{25 - 15} = -0.06$$

All the values can be tabulated as

t	T	T – Ta	dT/dt
0	80	60	-9.2
5	44.5	24.5	-5
10	30	10	-2.04
15	24.1	4.1	-0.83
20	21.7	1.7	-0.34
25	20.7	0.7	-0.06

If Newton's law of cooling holds, we can plot dT/dt versus $T - T_a$ and the points can be fit with a linear regression with zero intercept to estimate the cooling rate. As in the following plot, the result is $k = 0.1618/\text{min}$.



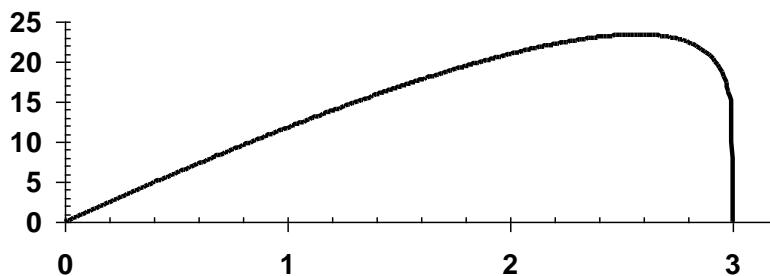
24.41 As in the plot, the initial point is assumed to be $e = 0$, $s = 40$. We can then use a combination of the trapezoidal and Simpsons rules to integrate the data as

$$I = (0.02 - 0) \frac{40 + 40}{2} + (0.05 - 0.02) \frac{40 + 37.5}{2} + (0.25 - 0.05) \frac{37.5 + 4(43 + 60) + 2(52) + 60}{12} = \\ 0.8 + 1.1625 + 4.358333 + 5.783333 = 12.10417$$

24.42 The function to be integrated is

$$Q = \int_0^3 2 \left(1 - \frac{r}{r_0}\right)^{1/6} (2\pi r) dr$$

A plot of the integrand can be developed as



As can be seen, the shape of the function indicates that we must use fine segmentation to attain good accuracy. Here are the results of using a variety of segments.

n	Q
2	25.1896
4	36.1635
8	40.9621
16	43.0705
32	44.0009
64	44.4127
128	44.5955

256	44.6767
512	44.7128
1024	44.7289
2048	44.7361
4096	44.7392
8192	44.7407
16384	44.7413
32768	44.7416
65536	44.7417
131072	44.7418
262144	44.7418

Therefore, the result to 4 significant figures appears to be 44.7418.

24.43 The work is computed as

$$W = \int_0^x F \, dx$$

A table can be set up holding the forces at the various displacements. A combination of Simpson's 1/3 and 3/8 rules can be used to determine the integral as shown in the last two columns,

x, m	F, N	Integral	Method
0	0		
0.05	10		
0.1	28	1.133333	Simp 1/3
0.15	46		
0.2	63	4.583333	Simp 1/3
0.25	82		
0.3	110		
0.35	130	14.41875	Simp 3/8
Sum→		20.13542	

24.44 Although the first three points are equally spaced, the remaining values are unequally spaced. Therefore, a good approach is to use Eq. 23.9 to perform the differentiation for all points. The results are summarized below:

t	x	v	a
0	153	68.26923	-35.14510
0.52	185	54.80769	-16.63005
1.04	210	50.97398	-13.20842
1.75	249	35.93855	-26.99478
2.37	261	16.05181	-23.26046
3.25	271	6.59273	-10.80561
3.83	273	0.30382	-10.88031

24.45 The distance traveled is equal to the integral of velocity

$$y = \int_{t_1}^{t_2} v(t) dt$$

A table can be set up holding the velocities at evenly spaced times ($h = 1$) over the integration interval. The Simpson's 1/3 rule can then be used to integrate this data as shown in the last column of the table

t	v	Simp 1/3 rule
0	0	
1	6	19.33333
2	34	
3	84	175.3333
4	156	
5	250	507.3333
6	366	
7	504	1015.333
8	664	
9	846	1699.333
10	1050	
11	1045	2090
12	1040	
13	1035	2070
14	1030	
15	1025	2050
16	1020	
17	1015	2030
18	1010	
19	1005	2010
20	1000	
21	1052	2105.333
22	1108	
23	1168	2337.333
24	1232	
25	1300	2601.333
26	1372	
27	1448	2897.333
28	1528	
29	1612	3225.333
30	1700	
Sum →		26833.33

Since the underlying functions are second order or less, this result should be exact. We can verify this by evaluating the integrals analytically,

$$y = \int_0^{10} 11t^2 - 5t \, dt = [3.66667t^3 - 2.5t^2]_0^{10} = 3416.667$$

$$y = \int_{10}^{20} 1100 - 5t \, dt = [1100t - 2.5t^2]_{10}^{20} = 10,250$$

$$y = \int_{20}^{30} 50t + 2(t-20)^2 dt = \left[\frac{2}{3}t^3 - 15t^2 + 800t \right]_{20}^{30} = 13,166.67$$

The total distance traveled is therefore $3416.667 + 10,250 + 13,166.67 = 26,833.33$.

24.46 6-segment trapezoidal rule:

$$y = (30 - 0) \frac{0 + 2(97.422 + 207.818 + 333.713 + 478.448 + 646.579) + 844.541}{2(6)} = 10,931.25$$

6-segment Simpson's 1/3 rule:

$$y = (30 - 0) \frac{0 + 4(97.422 + 333.713 + 646.579) + 2(207.818 + 478.448) + 844.541}{3(6)} = 10,879.88$$

6-point Gauss quadrature:

$$y = 10,879.619\overline{14}$$

Romberg integration:

	1	2	3	4
<i>n</i>	<i>ε_a →</i>	4.0634%	0.0098%	0.0000291%
1	12668.10909	10896.96330	10879.82315	10879.62077
2	11339.74974	10880.89441	10879.62393	
4	10995.60824	10879.70333		
8	10908.67956			

24.47 We can use the second-order difference formulas from Fig. 23.1 to compute the derivatives at each depth. For example, at the first point, we can use the forward difference to compute

$$\frac{d\varepsilon}{dt}(0.085) = -\frac{-0.16 + 4(0.13) - 3(0.1)}{8} = 0.05994$$

For the interior points, second-order centered differences can be used. For example, at the second point at ($t = 0.586$),

$$\frac{d\varepsilon}{dt}(0.586) = -\frac{0.16 - 0.1}{8} = 0.05994$$

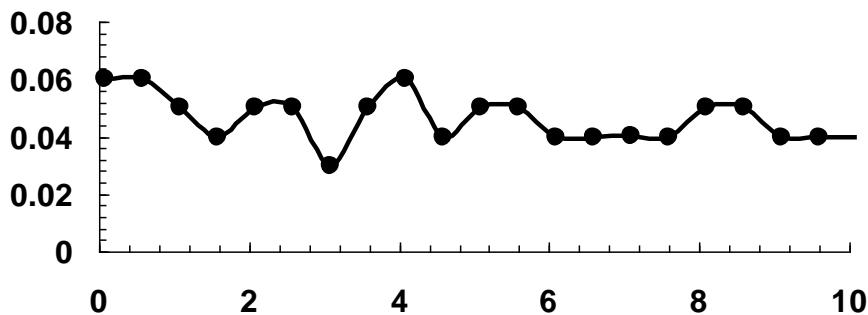
The other interior points can be determined in a similar fashion. For the last point, the second-order backward formula yields

$$\frac{d\varepsilon}{dt}(10.097) = -\frac{3(0.56) - 4(0.54) + 0.52}{8} = 0.03988$$

All the results are summarized in the following table.

t	ε	$d\varepsilon/dt$	t	ε	$d\varepsilon/dt$
0.085	0.10	0.05994	5.591	0.37	0.04995
0.586	0.13	0.05994	6.091	0.39	0.03996
1.086	0.16	0.04995	6.592	0.41	0.03996
1.587	0.18	0.03996	7.092	0.43	0.04
2.087	0.20	0.04995	7.592	0.45	0.03996
2.588	0.23	0.04995	8.093	0.47	0.04995
3.088	0.25	0.02997	8.593	0.50	0.04995
3.589	0.26	0.04995	9.094	0.52	0.03996
4.089	0.30	0.05994	9.594	0.54	0.03988
4.590	0.32	0.03996	10.097	0.56	0.03988
5.090	0.34	0.04995			

The derivatives can be plotted:

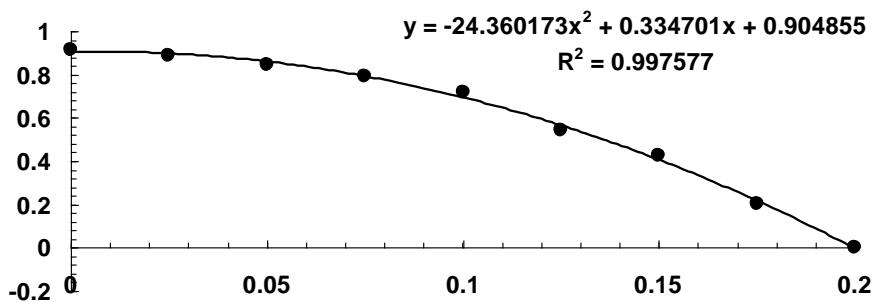


After about nine points, the derivatives have settled down and the mean and standard deviation can be computed as

Mean = 0.04328

Standard deviation = 0.004926

24.48 (a) After converting the radii to units of meters, a polynomial can be fit to the velocity data



The flow can then be evaluated analytically as

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$\begin{aligned}
 Q &= \int_0^{0.2} 2\pi(-24.360173 r^3 + 0.334701 r^2 + 0.904855 r) dr \\
 &= 2\pi \left[-6.09004325 r^4 + 0.111567 r^3 + 0.4524275 r^2 \right]_0^{0.2} = 0.05809161 \frac{\text{m}^3}{\text{s}}
 \end{aligned}$$

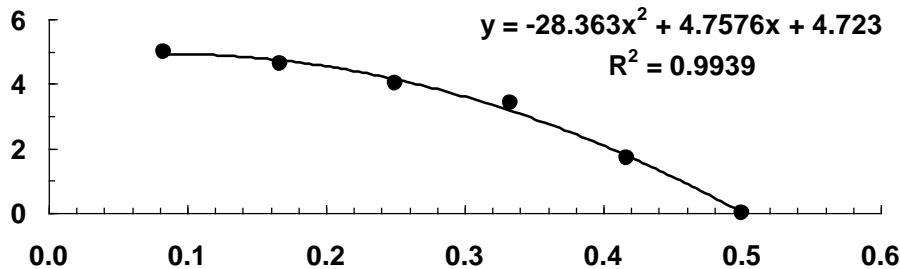
(b) The data and the integrand can be tabulated as shown below. Simpson's 1/3 rule can then be computed and summed in the last column.

$r, \text{ m}$	$v, \text{ m/s}$	$2\pi r v$	Simp 1/3
0	0.914	0	
0.025	0.89	0.139801	0.006877
0.05	0.847	0.266093	
0.075	0.795	0.374635	0.018470
0.1	0.719	0.451761	
0.125	0.543	0.426471	0.021334
0.15	0.427	0.402438	
0.175	0.204	0.22431	0.010831
0.2	0	0	
Sum →			0.057512

(c)

$$\varepsilon_t = \left| \frac{0.05809161 - 0.057512}{0.05809161} \right| \times 100\% = 0.998\%$$

24.49 (a) After converting the radii to units of feet, a polynomial can be fit to the non-core velocity data



The noncore flow can then be evaluated analytically as

$$\begin{aligned}
 Q_{\text{noncore}} &= \int_{0.08333}^{0.5} 2\pi(-28.3629 r^3 + 4.75757 r^2 + 4.723r) dr \\
 &= 2\pi \left[-7.09072 r^4 + 1.585857 r^3 + 2.3615 r^2 \right]_{0.08333}^{0.5} = 2.06379 \frac{\text{ft}^3}{\text{s}}
 \end{aligned}$$

This can be added to the core flow to determine the total flow

$$Q = Q_{\text{noncore}} + Q_{\text{core}} = 2.170447 \frac{\text{ft}^3}{\text{s}} + 5 \frac{\text{ft}}{\text{s}} \pi (0.083333 \text{ ft})^2 = 2.06379 + 0.109083 = 2.172873 \frac{\text{ft}^3}{\text{s}}$$

(b) The data and the integrand can be tabulated as shown below. Simpson's rules can then be computed and summed in the last column. Note that because we have an odd number of segments, we must use a combination of Simpson's 1/3 and 3/8 rules.

r, ft	v, ft/s	2πrv	Integral	Method
0.083333	5	2.617994		
0.166667	4.62	4.838053		
0.250000	4.01	6.298893	0.785253	Simp 1/3
0.333333	3.42	7.162831		
0.416667	1.69	4.42441		
0.500000	0	0	1.283144	Simp 3/8
Sum→		2.068397		

This numerical estimate can be added to the core flow to determine the total flow

$$Q = Q_{\text{noncore}} + Q_{\text{core}} = 2.068397 \frac{\text{ft}^3}{\text{s}} + 5 \frac{\text{ft}}{\text{s}} \pi (0.083333 \text{ ft})^2 = 2.068397 + 0.109083 = 2.17748 \frac{\text{ft}^3}{\text{s}}$$

(c)

$$\varepsilon_t = \left| \frac{2.172873 - 2.17748}{2.172873} \right| \times 100\% = 0.21\%$$

24.50 The following Excel worksheet solves the problem. Note that the derivative is calculated with a centered difference,

$$\frac{dV}{dT} = \frac{V_{450K} - V_{350K}}{100K}$$

	A	B	C	D	E	F	G	H	I
1	Prob. 24.50								
2									
3	P,atm	T=350K	T=400K	T=450K	dV/dT	(V - T (dV/dT)p)	Integral		
4	0.1	220	250	282.5	0.625	0			
5	5	4.1	4.7	5.23	0.0113	0.18	0.441	Trap	4.9
6	10	2.2	2.5	2.7	0.005	0.5	1.7	Trap	5
7	20	1.35	1.49	1.55	0.002	0.69	5.95	Trap	10
8	25	1.1	1.2	1.24	0.0014	0.64			5
9	30	0.9	0.99	1.03	0.0013	0.47	6.2	Simp1/3	5
10	40	0.68	0.75	0.78	0.001	0.35	4.1	Trap	10
11	45	0.61	0.675	0.7	0.0009	0.315			5
12	50	0.54	0.6	0.62	0.0008	0.28	3.15	Simp1/3	5
13					Total Integral =	21.541			
14									

24.51 A single application of the trapezoidal rule yields:

$$I = (23 - 3) \frac{12.5 + 1.2}{2} = 137$$

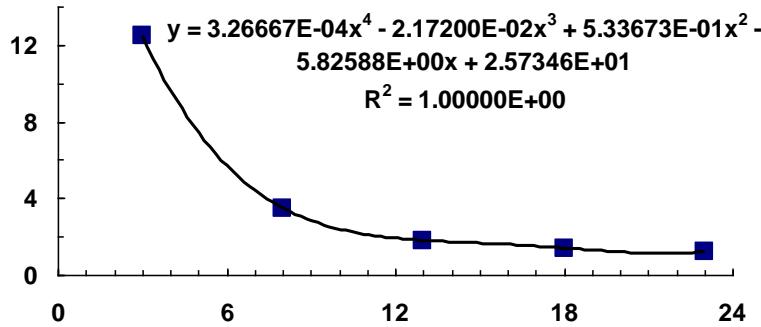
A 2-segment trapezoidal rule gives

$$I = (23 - 3) \frac{12.5 + 2(1.8) + 1.2}{4} = 86.5$$

A 4-segment trapezoidal rule gives

$$I = (23 - 3) \frac{12.5 + 2(3.5 + 1.8 + 1.4) + 1.2}{8} = 67.75$$

Because we do not know the true value, it would seem impossible to estimate the error. However, we can try to fit different order polynomials to see if we can get a decent fit to the data. This yields the surprising result that a 4th-order polynomial results in almost a perfect fit. For example, using the Excel trend line gives:



This can be integrated analytically to give 60.955. Note that the same result would result from using Boole's rule, Romberg integration or Gauss quadrature.

Therefore, we can estimate the errors as

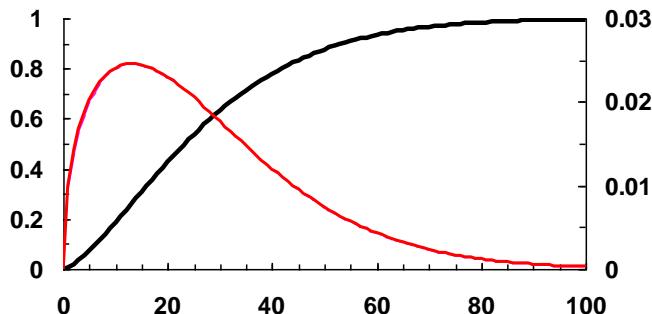
$$I = \left| \frac{60.955 - 137}{60.955} \right| \times 100\% = 124.76\%$$

$$I = \left| \frac{60.955 - 86.5}{60.955} \right| \times 100\% = 41.91\%$$

$$I = \left| \frac{60.955 - 67.75}{60.955} \right| \times 100\% = 11.15\%$$

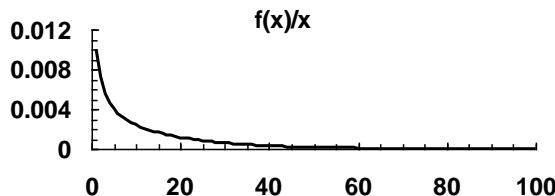
The ratio of these is 124.76:41.91:11.15 = 11.2:3.8:1. Thus, the result approximates the quartering of the error that we would expect according to Eq. 21.13.

24.52 (b) This problem can be solved in a number of ways. One approach is to set up Excel with a series of equally-spaced x values from 0 to 100. Then one of the formulas described in this Part of the book can be used to numerically compute the derivative. For example, x values with an interval of 1 and Eq. 23.9 can be used. The resulting plot of the function and its derivative is



(b) Inspection of this plot indicates that the maximum derivative occurs at about a diameter of 13.3.

(c) The function to be integrated looks like



This can be integrated from 1 to a high number using any of the methods provided in this book. For example, the trapezoidal rule can be used to integrate from 1 to 100, 1 to 200 and 1 to 300 using $h = 1$. The results are:

h	I
100	0.073599883
200	0.073632607
300	0.073632609

Thus, the integral seems to be converging to a value of 0.073633. S_m can be computed as $6 \times 0.073633 = 0.4418$.

24.53 (a) First, the distance can be converted to meters. Then, Eq. (23.9) can be used to compute the derivative at the surface as

$$\begin{aligned} x_0 &= 0 & f(x_0) &= 900 \\ x_1 &= 0.01 & f(x_1) &= 480 \\ x_2 &= 0.03 & f(x_2) &= 270 \end{aligned}$$

$$f'(0) = 900 \frac{2(0) - 0.01 - 0.03}{(0 - 0.01)(0 - 0.03)} + 480 \frac{2(0) - 0 - 0.03}{(0.01 - 0)(0.01 - 0.03)} + 270 \frac{2(0) - 0 - 0.01}{(0.03 - 0)(0.03 - 0.01)} = -52,500 \frac{\text{K}}{\text{m}}$$

The heat flux can be computed as

$$\text{Heat flux} = -0.028 \frac{\text{J}}{\text{s} \cdot \text{m} \cdot \text{K}} \left(-52,500 \frac{\text{K}}{\text{m}} \right) = 1,470 \frac{\text{W}}{\text{m}^2}$$

(b) The heat transfer can be computed by multiplying the flux by the area

$$\text{Heat transfer} = 1,470 \frac{\text{W}}{\text{m}^2} (200 \text{ cm} \times 50 \text{ cm}) \frac{\text{m}^2}{10,000 \text{ cm}^2} = 1,470 \text{ W}$$

24.54 (a) The pressure drop can be determined by integrating the pressure gradient

$$p = \int_{x_1}^{x_2} -\frac{8\mu Q}{\pi r(x)^4} dx$$

After converting the units to meters, a table can be set up holding the data and the integrand. The trapezoidal and Simpson's rules can then be used to integrate this data as shown in the last column of the table.

x, m	r, m	integrand	integral	method
0	0.002	-7957.75		
0.02	0.00135	-38333.2		
0.04	0.00134	-39490.3	-1338.5392	Simp 1/3
0.05	0.0016	-19428.1		
0.06	0.00158	-20430.6		
0.07	0.00142	-31315.3	-713.9319	Simp 3/8
0.1	0.002	-7957.75	-589.0959	Trap
Sum→		-2641.5670		

Therefore, the pressure drop is computed as $-2,641.567 \text{ N/m}^2$.

(b) The average radius can also be computed by integration as

$$\bar{r} = \frac{\int_{x_1}^{x_2} r(x) dx}{x_2 - x_1}$$

The numerical evaluations can again be determined by a combination of the trapezoidal and Simpson's rules.

x, m	r, m	integral	method
0	0.00200		

0.02	0.00135			
0.04	0.00134	5.83E-05	Simp 1/3	
0.05	0.00160			
0.06	0.00158			
0.07	0.00142	4.61E-05	Simp 3/8	
0.1	0.00200	5.13E-05	Trap	
Sum→		0.0001557		

Therefore, the average radius is $0.0001557/0.1 = 0.001557$ m. This value can be used to compute a pressure drop of

$$\Delta p = \frac{dp}{dx} \Delta x = -\frac{8\mu Q}{\pi r^4} \Delta x = -\frac{8(0.005)0.00001}{\pi(0.001557)^4} 0.1 = -2166.95 \frac{\text{N}}{\text{m}^2}$$

Thus, there is less pressure drop if the radius is at the constant mean value.

(c) The average Reynolds number can be computed by first determining the average velocity as

$$v = \frac{Q}{A_c} = \frac{0.00001}{\pi(0.001557)^2} = \frac{0.00001}{7.6152 \times 10^{-6}} = 1.31317 \frac{\text{m}}{\text{s}}$$

Then the Reynolds number can be computed as

$$Re = \frac{1 \times 10^3 (1.31317) 0.0031138}{0.005} = 817.796$$

24.55 After converting the units to meters, a table can be set up holding the data and the integrand. The trapezoidal and Simpson's rules can then be used to integrate this data as shown in the last column of the table.

r, m	v, m/s	integrand	integral	method
0	10	0		
0.016	9.69	1.168974	0.036905	Simp 1/3
0.032	9.3	2.243851		
0.048	8.77	3.173964	0.100138	Simp 1/3
0.064	7.95	3.836262		
0.0747	6.79	3.824296	0.040984	Trap
0.0787	5.57	3.305149	0.014259	Trap
0.0795	4.89	2.931144	0.002495	Trap
0.08	0	0	0.000733	Trap
Sum→		0.195514		

Therefore, the mass flow rate is 0.195514 kg/s.

CHAPTER 25

25.1 The analytical solution can be derived by separation of variables

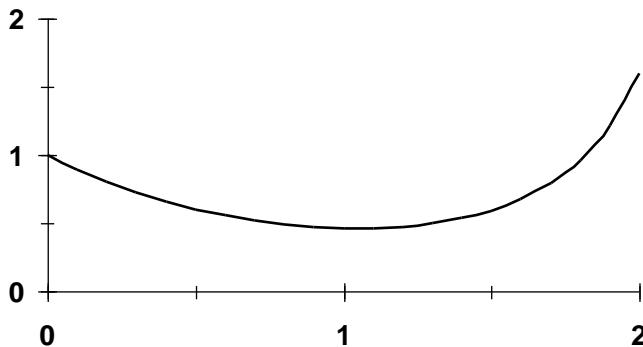
$$\int \frac{dy}{y} = \int x^2 - 1.1 dx$$

$$\ln y = \frac{x^3}{3} - 1.1x + C$$

Substituting the initial conditions yields $C = 0$. Taking the exponential gives the final result

$$y = e^{\frac{x^3}{3} - 1.1x}$$

The result can be plotted as



25.2 Euler's method with $h = 0.5$

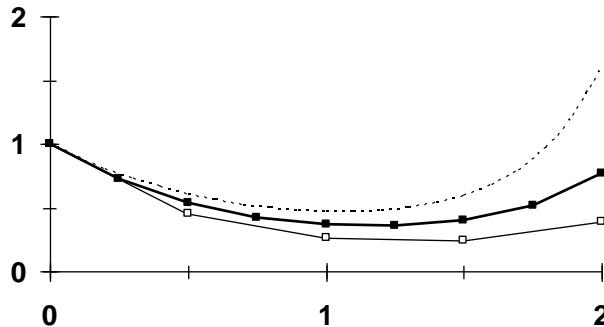
x	y	dy/dx
0	1	-1.1
0.5	0.45	-0.3825
1	0.25875	-0.02588
1.5	0.245813	0.282684
2	0.387155	1.122749

Euler's method with $h = 0.25$ gives

x	y	dy/dx
0	1	-1.1
0.25	0.725	-0.75219
0.5	0.536953	-0.45641
0.75	0.422851	-0.22728
1	0.36603	-0.0366
1.25	0.356879	0.165057
1.5	0.398143	0.457865

1.75	0.51261	1.005997
2	0.764109	2.215916

The results can be plotted along with the analytical solution as



25.3 For Heun's method, the value of the slope at $x = 0$ can be computed as -1.1 which can be used to compute the value of y at the end of the interval as

$$y(0.5) = 1 + (0 - 1.1(1))0.5 = 0.45$$

The slope at the end of the interval can be computed as

$$y'(0.5) = 0.45(0.5)^2 - 1.1(0.45) = -0.3825$$

which can be averaged with the initial slope to predict

$$y(0.5) = 1 + \frac{-1.1 - 0.3825}{2} 0.5 = 0.629375$$

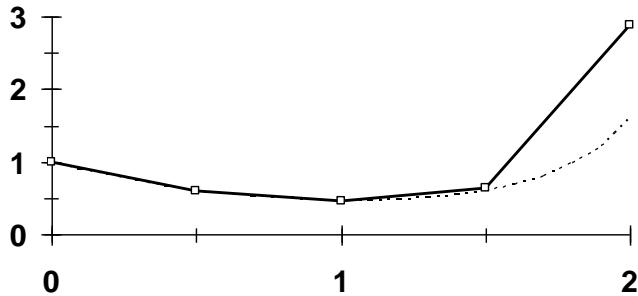
This formula can then be iterated to yield

j	y_j	$ \epsilon_a $
0	0.45	
1	0.629375	28.5%
2	0.5912578	6.45%
3	0.5993577	1.35%
4	0.5976365	0.288%

The remaining steps can be implemented with the result

x_i	y_i
0	1.0000000
0.5	0.5976365
1	0.4590875
1.5	0.6269127
2	2.8784358

The results along with the analytical solution are displayed below:



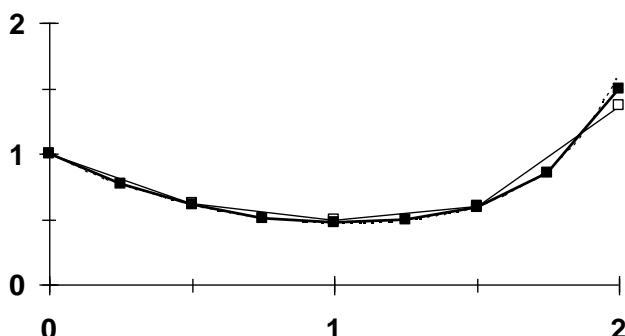
25.4 The midpoint method with $h = 0.5$

x	y	dy/dx	y_m	$dy/dx\text{-mid}$
0	1	-1.1	0.725	-0.75219
0.5	0.623906	-0.53032	0.491326	-0.26409
1	0.491862	-0.04919	0.479566	0.221799
1.5	0.602762	0.693176	0.776056	1.52301
2	1.364267	3.956374	2.35336	9.32519

with $h = 0.25$ gives

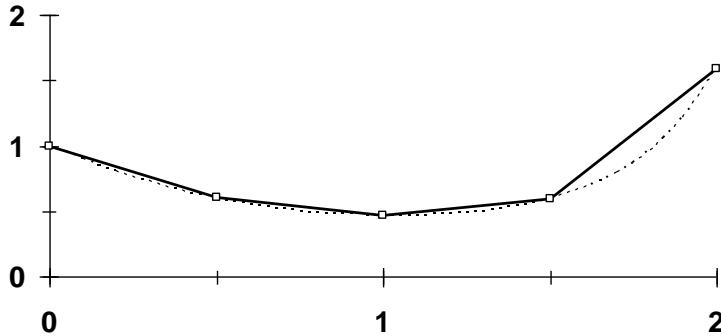
x	y	dy/dx	y_m	$dy/dx\text{-mid}$
0	1	-1.1	0.8625	-0.93527
0.25	0.766182	-0.79491	0.666817	-0.63973
0.5	0.60625	-0.51531	0.541836	-0.38436
0.75	0.510158	-0.27421	0.475882	-0.15912
1	0.470378	-0.04704	0.464498	0.076932
1.25	0.489611	0.226445	0.517916	0.409478
1.5	0.59198	0.680777	0.677077	1.043122
1.75	0.852761	1.673543	1.061954	2.565282
2	1.494081	4.332836	2.035686	6.953139

The results can be plotted along with the analytical solution as



25.5 The 4th-order RK method with $h = 0.5$ gives

x	y	k_1	y_m	k_2	y_m	k_3	y_e	k_4	ϕ
0	1	-1.1	0.725	-0.75219	0.811953	-0.8424	0.578799	-0.49198	-0.79686
0.5	0.60157	-0.51133	0.473737	-0.25463	0.537912	-0.28913	0.457006	-0.0457	-0.27409
1	0.464524	-0.04645	0.452911	0.209471	0.516892	0.239062	0.584055	0.671663	0.253713
1.5	0.59138	0.680087	0.761402	1.494252	0.964943	1.893701	1.538231	4.460869	1.986144
2	1.584452	4.594911	2.73318	10.83023	4.292008	17.00708	10.08799	51.95317	18.70378



25.6 (a) The analytical solution can be derived by separation of variables

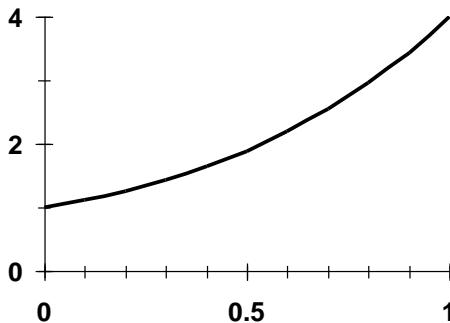
$$\int \frac{dy}{\sqrt{y}} = \int 1 + 2x \, dx$$

$$2\sqrt{y} = x + x^2 + C$$

Substituting the initial conditions yields $C = 2$. Substituting this value and solving for y gives the final result

$$y = \frac{(x^2 + x + 2)^2}{4}$$

The result can be plotted as



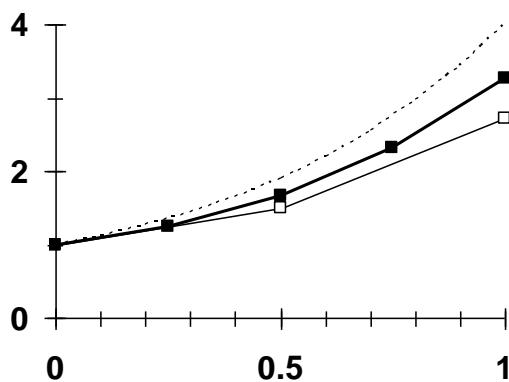
(b) Euler's method with $h = 0.5$

x	y	dy/dx
0	1	1
0.5	1.5	2.44949
1	2.724745	4.95204

Euler's method with $h = 0.25$ gives

x	y	dy/dx
0	1	1
0.25	1.25	1.677051
0.5	1.669263	2.583999
0.75	2.315263	3.803997
1	3.266262	5.421841

The results can be plotted along with the analytical solution as



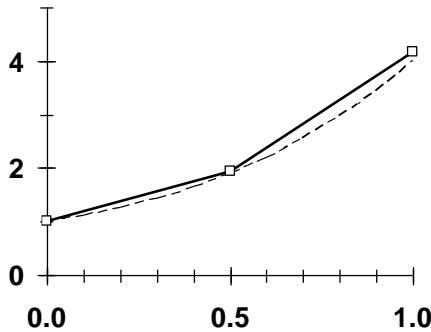
(c) For Heun's method, the first step along with the associated iterations is

j	y_j^j	$ \epsilon_a , \%$
0	1.500000	
1	1.862372	19.458
2	1.932344	3.621
3	1.945044	0.653

The remaining steps can be implemented with the result

x_i	y_i
0	1
0.5	1.945044
1	4.169277

The results along with the analytical solution are displayed below:



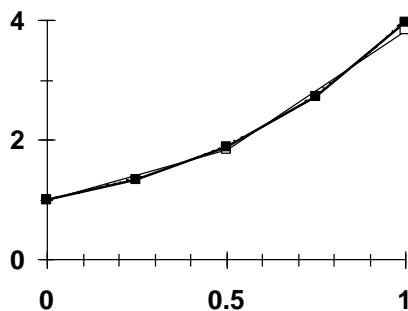
(d) The midpoint method with $h = 0.5$

x	y	dy/dx	y_m	dy/dx-mid
0	1	1	1.25	1.677051
0.5	1.838525	2.711845	2.516487	3.96586
1	3.821455	5.864563	5.287596	8.048171

with $h = 0.25$ gives

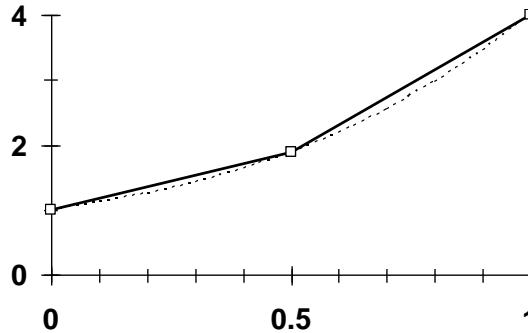
x	y	dy/dx	y_m	dy/dx-mid
0	1	1	1.125	1.325825
0.25	1.331456	1.730831	1.54781	2.177193
0.5	1.875755	2.739164	2.21815	3.351027
0.75	2.713511	4.118185	3.228284	4.941043
1	3.948772	5.961455	4.693954	7.041299

The results can be plotted along with the analytical solution as



(e) The 4th-order RK method with $h = 0.5$ gives

x	y	k_1	y_m	k_2	y_m	k_3	y_e	k_4	ϕ
0	1	1	1.25	1.67705	1.41926	1.78699	1.89350	2.75209	1.78003
0.5	1.89001	2.74956	2.57740	4.01357	2.89341	4.25251	4.01627	6.01219	4.21577
1	3.99784	5.99838	5.49743	8.20631	6.04942	8.60845	8.30206	11.5253	8.52554



25.7 The second-order ODE is transformed into a pair of first-order ODEs as in

$$\begin{aligned}\frac{dy}{dt} &= z & y(0) &= 2 \\ \frac{dz}{dt} &= 0.5x - y & z(0) &= 0\end{aligned}$$

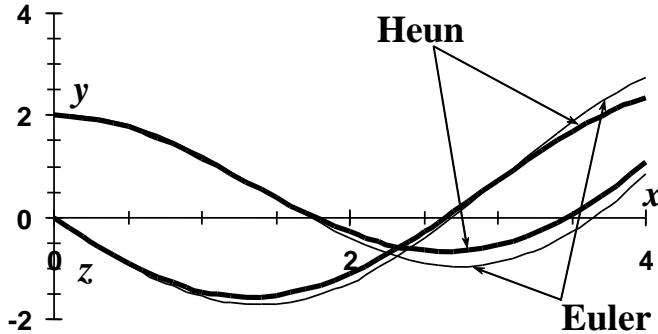
(a) The first few steps of Euler's method are

x	y	z	dy/dx	dz/dx
0	2	0	0	-2
0.1	2	-0.2	-0.2	-1.95
0.2	1.98	-0.395	-0.395	-1.88
0.3	1.9405	-0.583	-0.583	-1.7905
0.4	1.8822	-0.76205	-0.76205	-1.6822
0.5	1.805995	-0.93027	-0.93027	-1.556

(b) For Heun (without iterating the corrector) the first few steps are

x	y	z	dy/dx	dz/dx	y _{end}	z _{end}	dy/dx	dz/dx	ave slope
0	2	0	0	-2	2	-0.2	-0.2	-1.95	-0.1
0.1	1.99	-0.1975	-0.1975	-1.94	1.97025	-0.3915	-0.3915	-1.87025	-0.2945
0.2	1.96055	-0.38801	-0.38801	-1.86055	1.921749	-0.57407	-0.57407	-1.77175	-0.48104
0.3	1.912446	-0.56963	-0.56963	-1.76245	1.855483	-0.74587	-0.74587	-1.65548	-0.65775
0.4	1.846671	-0.74052	-0.74052	-1.64667	1.772619	-0.90519	-0.90519	-1.52262	-0.82286
0.5	1.764385	-0.89899	-0.89899	-1.51439	1.674486	-1.05043	-1.05043	-1.37449	-0.97471

Both results are plotted below:

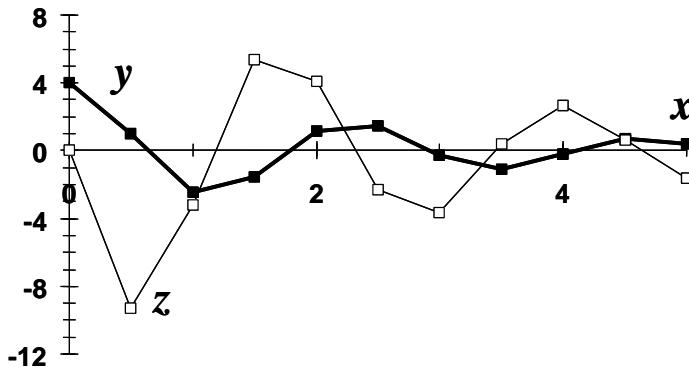


25.8 The second-order ODE is transformed into a pair of first-order ODEs as in

$$\begin{aligned}\frac{dy}{dx} &= z & y(0) &= 4 \\ \frac{dz}{dx} &= -0.6z - 8y & z(0) &= 0\end{aligned}$$

The results for the 4th-order RK method are tabulated and plotted below:

x	y	z	k_{11}	k_{12}	y_{mid}	z_{mid}	k_{21}	k_{22}	y_{mid}	z_{mid}	k_{31}	k_{32}	y_{end}	z_{end}	k_{41}	k_{42}	ϕ_1	ϕ_2
0	4.0000	0.0000	0.00	-32.00	4.00	-8.00	-8.00	-27.20	2.00	-6.80	-6.80	-11.92	0.60	-5.96	-5.96	-1.22	-5.93	-18.58
0.5	1.0367	-9.2887	-9.29	-2.72	-1.29	-9.97	-9.97	16.27	-1.46	-5.22	-5.22	14.78	-1.57	-1.90	-1.90	13.74	-6.93	12.18
1	-2.4276	-3.1969	-3.20	21.34	-3.23	2.14	2.14	24.53	-1.89	2.94	2.94	13.38	-0.96	3.49	3.49	5.58	1.74	17.12
1.5	-1.5571	5.3654	5.37	9.24	-0.22	7.67	7.67	-2.88	0.36	4.65	4.65	-5.68	0.77	2.53	2.53	-7.64	5.42	-2.59
2	1.1539	4.0719	4.07	-11.67	2.17	1.15	1.15	-18.07	1.44	-0.44	-0.44	-11.27	0.93	-1.56	-1.56	-6.51	0.65	-12.81
2.5	1.4810	-2.3334	-2.33	-10.45	0.90	-4.95	-4.95	-4.21	0.24	-3.39	-3.39	0.07	-0.21	-2.30	-2.30	3.08	-3.55	-2.61
3	-0.2935	-3.6375	-3.64	4.53	-1.20	-2.50	-2.50	11.13	-0.92	-0.86	-0.86	7.87	-0.72	0.30	0.30	5.59	-1.68	8.02
3.5	-1.1319	0.3723	0.37	8.83	-1.04	2.58	2.58	6.76	-0.49	2.06	2.06	2.66	-0.10	1.70	1.70	-0.22	1.89	4.58
4	-0.1853	2.6602	2.66	-0.11	0.48	2.63	2.63	-5.42	0.47	1.31	1.31	-4.56	0.47	0.38	0.38	-3.97	1.82	-4.01
4.5	0.7241	0.6564	0.66	-6.19	0.89	-0.89	-0.89	-6.57	0.50	-0.99	-0.99	-3.42	0.23	-1.05	-1.05	-1.21	-0.69	-4.56
5	0.3782	-1.6258	-1.63	-2.05	-0.03	-2.14	-2.14	1.51	-0.16	-1.25	-1.25	2.00	-0.25	-0.63	-0.63	2.34	-1.50	1.22



25.9 (a) The Heun method without iteration can be implemented as in the following table:

t	y	k_1	y_{end}	k_2	ϕ
---	---	-------	-----------	-------	--------

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

0	1	0	1	0.000995	0.000498
0.1	1.00005	0.000995	1.000149	0.007843	0.004419
0.2	1.000492	0.007845	1.001276	0.025841	0.016843
0.3	1.002176	0.025865	1.004762	0.059335	0.0426
0.4	1.006436	0.059434	1.012379	0.11156	0.085497
0.5	1.014986	0.111847	1.02617	0.184731	0.148289
•				•	
•				•	
•				•	
2.9	3.784421	0.051826	3.789604	0.01065	0.031238
3	3.787545	0.010644	3.78861	0.000272	0.005458

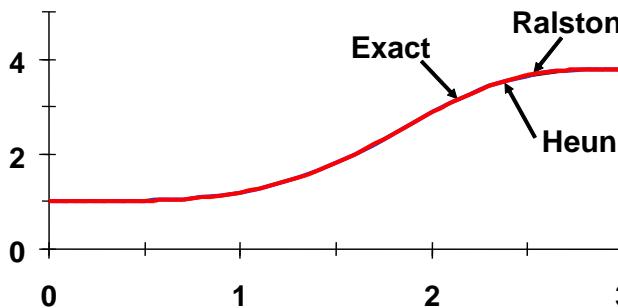
(b) The Ralston 2nd order RK method can be implemented as in the following table:

t	y	k₁	yint	k₂	ϕ
0	1	0	1	0.000421	0.00028
0.1	1.000028	0.000995	1.000103	0.005278	0.003851
0.2	1.000413	0.007845	1.001001	0.020043	0.015977
0.3	1.002011	0.02586	1.00395	0.049332	0.041508
0.4	1.006162	0.059418	1.010618	0.096672	0.084254
0.5	1.014587	0.111803	1.022972	0.164537	0.146959
•				•	
•				•	
•				•	
2.9	3.785066	0.051835	3.788954	0.017276	0.028796
3	3.787946	0.010646	3.788744	0.001116	0.004293

Both methods are displayed on the following plot along with the exact solution,

$$y = e^{-\cos t + \frac{\cos^3 t}{3} + \frac{2}{3}}$$

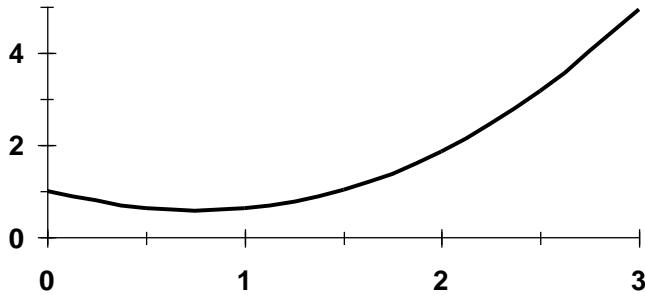
Note that both methods agree closely with the exact solution.



25.10 The solution results are as in the following table and plot:

t	y	k₁	k₂	k₃	ϕ
0	1	-1	-0.6875	-0.5625	-0.71875
0.5	0.640625	-0.39063	0.019531	0.144531	-0.02799

1	0.626628	0.373372	0.842529	0.967529	0.78517
1.5	1.019213	1.230787	1.735591	1.860591	1.67229
2	1.855358	2.144642	2.670982	2.795982	2.604092
2.5	3.157404	3.092596	3.631947	3.756947	3.562889
3	4.938848	4.061152	4.608364	4.733364	4.537995



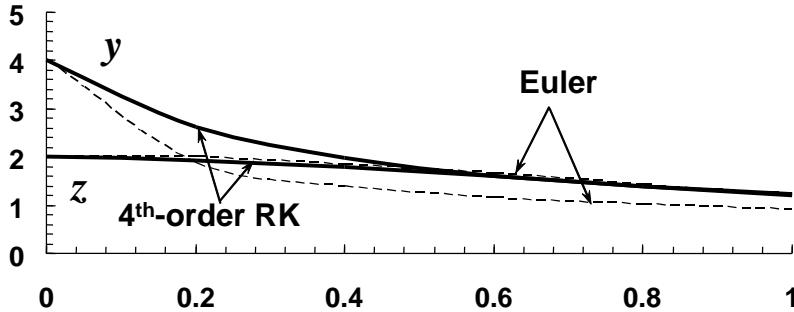
25.11 (a) Euler

x	y	z	dy/dx	dz/dx
0	2.0000	4.0000	0.00	-10.67
0.2	2.0000	1.8667	-0.73	-2.32
0.4	1.8550	1.4021	-1.03	-1.22
0.6	1.6492	1.1590	-1.10	-0.74
0.8	1.4286	1.0113	-1.06	-0.49
1	1.2166	0.9139	-0.96	-0.34

(b) 4th-order RK

x	y	z	k ₁₁	k ₁₂	k ₂₁	k ₂₂	k ₃₁	k ₃₂	k ₄₁	k ₄₂	ϕ ₁	ϕ ₂
0	2.000	4.000	0.000	-10.667	-0.381	-5.736	-0.305	-7.678	-0.603	-3.926	-0.329	-6.903
0.2	1.934	2.619	-0.594	-4.423	-0.786	-2.962	-0.748	-3.338	-0.888	-2.266	-0.758	-3.215
0.4	1.783	1.976	-0.884	-2.321	-0.962	-1.718	-0.947	-1.830	-0.991	-1.377	-0.949	-1.799
0.6	1.593	1.617	-0.990	-1.387	-1.001	-1.087	-0.999	-1.131	-0.989	-0.897	-0.997	-1.120
0.8	1.393	1.392	-0.990	-0.901	-0.963	-0.732	-0.968	-0.753	-0.928	-0.617	-0.963	-0.748
1	1.201	1.243	-0.930	-0.618	-0.884	-0.515	-0.893	-0.526	-0.840	-0.441	-0.887	-0.524

Both methods are plotted on the same graph below. Notice how Euler's method (particularly for z) is not very accurate for this step size. The 4th-order RK is much closer to the exact solution.



$$25.12 \quad \frac{dy}{dx} = 10e^{-\frac{(x-2)^2}{2(0.075)^2}} - 0.6y$$

4th-order RK method:

One step ($h = 0.5$): $y_1 = 0.3704188$

Two steps ($h = 0.25$): $y_2 = 0.3704096$

$$\Delta_{\text{present}} = -9.119 \times 10^{-6}$$

$$\text{correction} = \frac{\Delta}{15} = -6.08 \times 10^{-7}$$

$$y_2 = 0.370409$$

$$\frac{dy}{dx} = -0.3$$

$$y_{\text{scale}} = 0.5 + |0.5(-0.3)| = 0.65$$

$$\Delta_{\text{new}} = 0.001(0.65) = 0.00065$$

Since $\Delta_{\text{present}} < \Delta_{\text{new}}$, therefore, increase step.

$$h_{\text{new}} = 0.5 \left| \frac{0.00065}{9.119 \times 10^{-6}} \right|^{0.2} = 1.1737$$

25.13 We will look at the first step only

$$\Delta_{\text{present}} = y_2 - y_1 = -0.24335$$

$$\frac{dy}{dx} = 4e^0 - 0.5(2) = 3$$

$$y_{\text{scale}} = 2 + (2(3)) = 8$$

$$\Delta_{\text{new}} = 0.001(8) = 0.008$$

Because $\Delta_{\text{present}} > \Delta_{\text{new}}$, decrease step.

25.14 The calculation of the k 's can be summarized in the following table:

	x	y	$f(x,y)$
k_1	0	2	3
k_2	0.2	2.6	3.394043
k_3	0.3	2.98866	3.590667
k_4	0.6	4.154161	4.387217
k_5	1	6.251995	5.776166
k_6	0.875	5.511094	5.299464

These can then be used to compute the 4th-order prediction

$$y_1 = 2 + \left(\frac{37}{378} 3 + \frac{250}{621} 3.590667 + \frac{125}{594} 4.387217 + \frac{512}{1771} 5.299464 \right) 1 = 6.194491$$

along with a fifth-order formula:

$$y_1 = 2 + \left(\frac{2825}{27,648} 3 + \frac{18,575}{48,384} 3.590667 + \frac{13,525}{55,296} 4.387217 + \frac{277}{14,336} 5.776166 + \frac{1}{4} 5.299464 \right) 1 = 6.194572$$

The error estimate is obtained by subtracting these two equations to give

$$E_a = 6.194572 - 6.194491 = 0.0000805$$

25.15

Option Explicit

```

Sub EulerTest()
    Dim i As Integer, m As Integer
    Dim xi As Double, yi As Double, xf As Double, dx As Double, xout As Double
    Dim xp(200) As Double, yp(200) As Double
    'Assign values
    yi = 1
    xi = 0
    xf = 4
    dx = 0.5
    xout = 0.5
    'Perform numerical Integration of ODE
    Call ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
    'Display results
    Sheets("Sheet1").Select
    Range("a5:b205").ClearContents
    Range("a5").Select
    For i = 0 To m
        ActiveCell.Value = xp(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yp(i)
        ActiveCell.Offset(1, -1).Select
    Next i

```

```

Range("a5").Select
End Sub

Sub ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
'Generate an array that holds the solution
Dim x As Double, y As Double, xend As Double
Dim h As Double
m = 0
xp(m) = xi
yp(m) = yi
x = xi
y = yi
Do           'Print loop
    xend = x + xout
    If (xend > xf) Then xend = xf  'Trim step if increment exceeds end
    h = dx
    Call Integrator(x, y, h, xend)
    m = m + 1
    xp(m) = x
    yp(m) = y
    If (x >= xf) Then Exit Do
Loop
End Sub

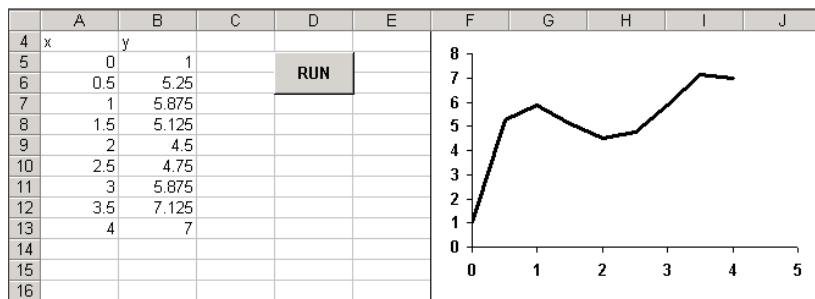
Sub Integrator(x, y, h, xend)
Dim ynew As Double
Do           'Calculation loop
    If (xend - x < h) Then h = xend - x  'Trim step if increment exceeds end
    Call Euler(x, y, h, ynew)
    y = ynew
    If (x >= xend) Then Exit Do
Loop
End Sub

Sub Euler(x, y, h, ynew)
Dim dydx As Double
'Implement Euler's method
Call Derivs(x, y, dydx)
ynew = y + dydx * h
x = x + h
End Sub

Sub Derivs(x, y, dydx)
'Define ODE
dydx = -2 * x ^ 3 + 12 * x ^ 2 - 20 * x + 8.5
End Sub

```

25.16 Example 25.1:

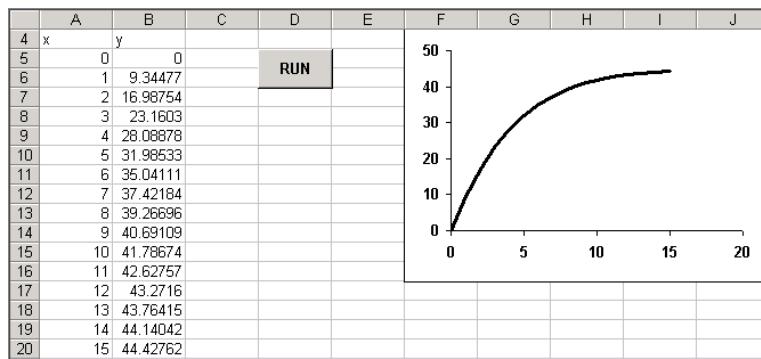


Example 25.4 (nonlinear model). Change time steps and initial conditions to

```
'Assign values
yi = 0
xi = 0
xf = 15
dx = 0.5
xout = 1
```

Change Derivs Sub to

```
Sub Derivs(t, v, dvdt)
'Define ODE
dvdt = 9.8 - 12.5 / 68.1 * (v + 8.3 * (v / 46) ^ 2.2)
End Sub
```



25.17

Option Explicit

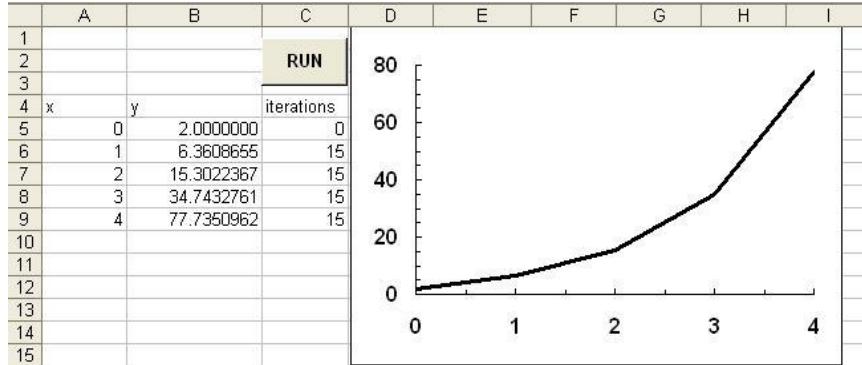
```
Sub Heun()
Dim maxit As Integer, es As Double
Dim n As Integer, m As Integer, i As Integer, iter As Integer
Dim xi As Double, xf As Double, yi As Double, h As Double
Dim x As Double, y As Double, y2 As Double, y2old As Double
Dim k1 As Double, k2 As Double, slope As Double
Dim xp(1000) As Double, yp(1000) As Double, itr(1000) As Integer
Dim ea As Double
maxit = 15: es = 0.0000001
xi = 0: xf = 4: yi = 2
n = 4
h = (xf - xi) / n
x = xi
y = yi
m = 0
xp(m) = x
yp(m) = y
For i = 1 To n
    Call Derivs(x, y, k1)
    y2 = y + k1 * h
    iter = 0
    Do
        y2old = y2
        Call Derivs(x + h, y2, k2)
        slope = (k1 + k2) / 2
        y2 = y + slope * h
        iter = iter + 1
    Loop While Abs(y2 - y2old) > es
    x = x + h
    y = y2
    m = m + 1
    xp(m) = x
    yp(m) = y
    If iter > maxit Then
        MsgBox "Maximum iterations reached"
        End
    End If
Next i
End Sub
```

```

ea = Abs((y2 - y2old) / y2) * 100
If ea < es Or iter >= maxit Then Exit Do
Loop
m = m + 1
x = x + h
xp(m) = x
yp(m) = y2
itr(m) = iter
y = y2
Next i
Sheets("Heun").Select
Range("a5:b1005").ClearContents
Range("a5").Select
For i = 0 To m
    ActiveCell.Value = xp(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = yp(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = itr(i)
    ActiveCell.Offset(1, -2).Select
Next i
Range("a5").Select
End Sub

Sub Derivs(x, y, dydx)
'Define ODE
dydx = 4 * Exp(0.8 * x) - 0.5 * y
End Sub

```



25.18

Option Explicit

```

Sub RK4Test()
Dim i As Integer, m As Integer
Dim xi As Double, yi As Double, xf As Double, dx As Double, xout As Double
Dim xp(200) As Double, yp(200) As Double
'Assign values
yi = 1
xi = 0
xf = 4
dx = 0.5
xout = 0.5
'Perform numerical Integration of ODE
Call ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
'Display results
Sheets("Sheet1").Select

```

```

Range("a5:b205").ClearContents
Range("a5").Select
For i = 0 To m
    ActiveCell.Value = xp(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = yp(i)
    ActiveCell.Offset(1, -1).Select
Next i
Range("a5").Select
End Sub

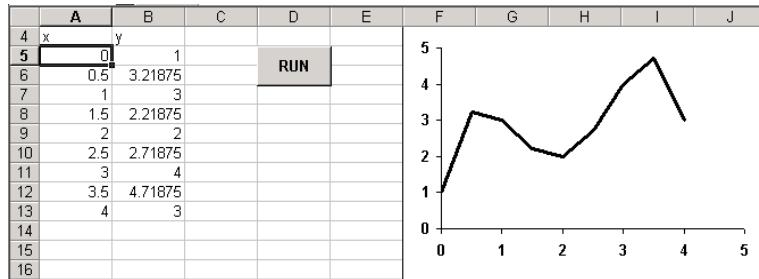
Sub ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
    'Generate an array that holds the solution
    Dim x As Double, y As Double, xend As Double
    Dim h As Double
    m = 0
    xp(m) = xi
    yp(m) = yi
    x = xi
    y = yi
    Do      'Print loop
        xend = x + xout
        If (xend > xf) Then xend = xf  'Trim step if increment exceeds end
        h = dx
        Call Integrator(x, y, h, xend)
        m = m + 1
        xp(m) = x
        yp(m) = y
        If (x >= xf) Then Exit Do
    Loop
End Sub

Sub Integrator(x, y, h, xend)
    Dim ynew As Double
    Do      'Calculation loop
        If (xend - x < h) Then h = xend - x  'Trim step if increment exceeds end
        Call RK4(x, y, h, ynew)
        y = ynew
        If (x >= xend) Then Exit Do
    Loop
End Sub

Sub RK4(x, y, h, ynew)
    'Implement RK4 method
    Dim k1 As Double, k2 As Double, k3 As Double, k4 As Double
    Dim ym As Double, ye As Double, slope As Double
    Call Derivs(x, y, k1)
    ym = y + k1 * h / 2
    Call Derivs(x + h / 2, ym, k2)
    ym = y + k2 * h / 2
    Call Derivs(x + h / 2, ym, k3)
    ye = y + k3 * h
    Call Derivs(x + h, ye, k4)
    slope = (k1 + 2 * (k2 + k3) + k4) / 6
    ynew = y + slope * h
    x = x + h
End Sub

Sub Derivs(x, y, dydx)
    'Define ODE
    dydx = -2 * x ^ 3 + 12 * x ^ 2 - 20 * x + 8.5
End Sub

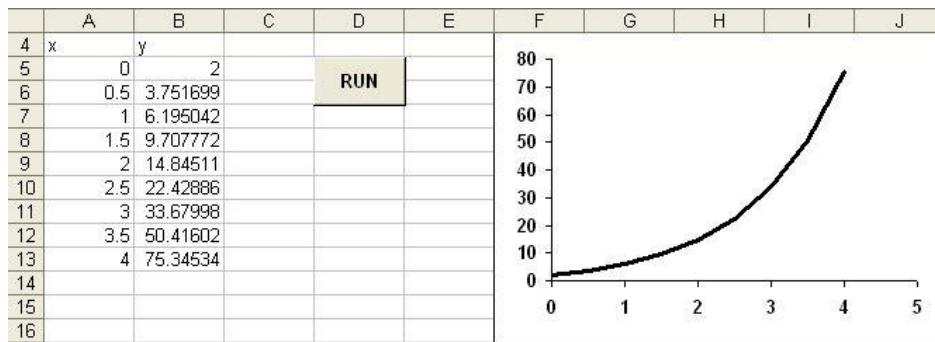
```

Example 25.7a:Example 25.7b: Change time steps and initial conditions to

```
'Assign values
yi = 2
xi = 0
xf = 4
dx = 0.5
xout = 0.5
```

Change Derivs Sub to

```
Sub Derivs(x, y, dydx)
'Define ODE
dydx = 4 * Exp(0.8 * x) - 0.5 * y
End Sub
```

**25.19**

Option Explicit

```
Sub RK4SysTest()
Dim i As Integer, m As Integer, n As Integer, j As Integer
Dim xi As Double, yi(10) As Double, xf As Double
Dim dx As Double, xout As Double
Dim xp(200) As Double, yp(200, 10) As Double
'Assign values
n = 2
xi = 0
xf = 2
yi(1) = 4
yi(2) = 6
dx = 0.5
```

```

xout = 0.5
'Perform numerical Integration of ODE
Call ODESolver(xi, yi, xf, dx, xout, xp, yp, m, n)
'Display results
Sheets("Sheet1").Select
Range("a5:n205").ClearContents
Range("a5").Select
For i = 0 To m
    ActiveCell.Value = xp(i)
    For j = 1 To n
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yp(i, j)
    Next j
    ActiveCell.Offset(1, -n).Select
Next i
Range("a5").Select
End Sub

Sub ODESolver(xi, yi, xf, dx, xout, xp, yp, m, n)
'Generate an array that holds the solution
Dim i As Integer
Dim x As Double, y(10) As Double, xend As Double
Dim h As Double
m = 0
x = xi
'set initial conditions
For i = 1 To n
    y(i) = yi(i)
Next i
'save output values
xp(m) = x
For i = 1 To n
    yp(m, i) = y(i)
Next i
Do      'Print loop
    xend = x + xout
    If (xend > xf) Then xend = xf  'Trim step if increment exceeds end
    h = dx
    Call Integrator(x, y, h, n, xend)
    m = m + 1
    'save output values
    xp(m) = x
    For i = 1 To n
        yp(m, i) = y(i)
    Next i
    If (x >= xf) Then Exit Do
Loop
End Sub

Sub Integrator(x, y, h, n, xend)
Dim j As Integer
Dim ynew(10) As Double
Do      'Calculation loop
    If (xend - x < h) Then h = xend - x  'Trim step if increment exceeds end
    Call RK4Sys(x, y, h, n, ynew)
    For j = 1 To n
        y(j) = ynew(j)
    Next j
    If (x >= xend) Then Exit Do
Loop
End Sub

```

```

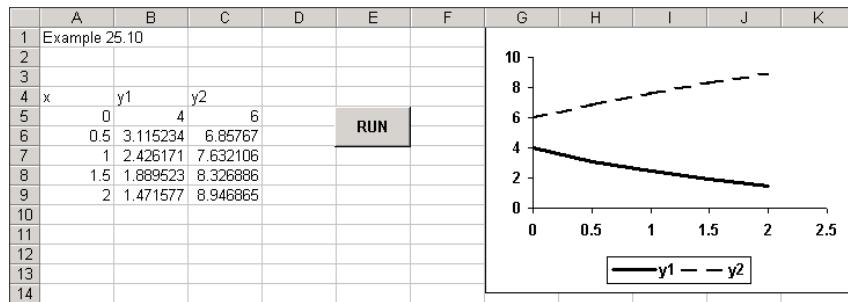
Sub RK4Sys(x, y, h, n, ynew)
Dim j As Integer
Dim ym(10) As Double, ye(10) As Double
Dim k1(10) As Double, k2(10) As Double, k3(10) As Double, k4(10) As Double
Dim slope(10)
'Implement RK4 method for systems of ODEs
Call Derivs(x, y, k1)
For j = 1 To n
    ym(j) = y(j) + k1(j) * h / 2
Next j
Call Derivs(x + h / 2, ym, k2)
For j = 1 To n
    ym(j) = y(j) + k2(j) * h / 2
Next j
Call Derivs(x + h / 2, ym, k3)
For j = 1 To n
    ye(j) = y(j) + k3(j) * h
Next j
Call Derivs(x + h, ye, k4)
For j = 1 To n
    slope(j) = (k1(j) + 2 * (k2(j) + k3(j)) + k4(j)) / 6
Next j
For j = 1 To n
    ynew(j) = y(j) + slope(j) * h
Next j
x = x + h

End Sub

Sub Derivs(x, y, dydx)
'Define ODE
dydx(1) = -0.5 * y(1)
dydx(2) = 4 - 0.3 * y(2) - 0.1 * y(1)
End Sub

```

Application to Example 25.10:



25.20 Main Program:

```

%Damped spring mass system
%mass: m=10 kg
%damping: c=5,40,200 N/(m/s)
%spring: k=40 N/m
% MATLAB 5 version
%Independent Variable t, tspan=[tstart tstop]
%initial conditions [x(1)=velocity, x(2)=displacement];

```

```
tspan=[0 15]; ic=[0 1];

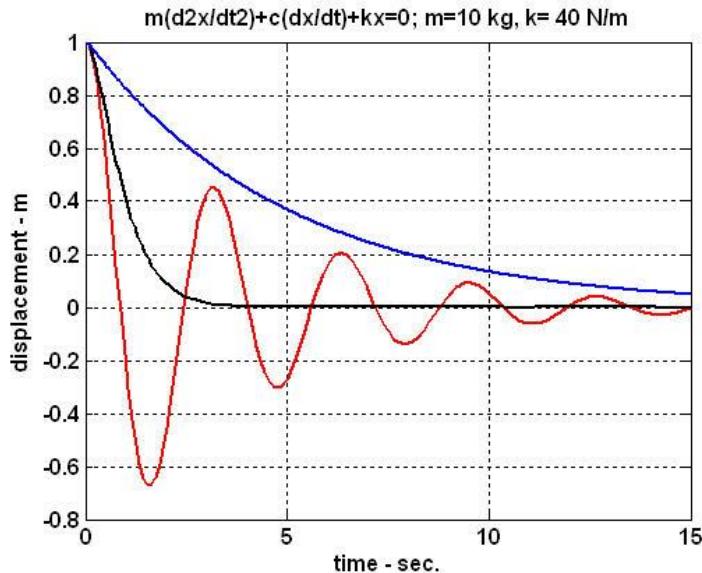
global cm km
m=10; c(1)=5; c(2)=40; c(3)=200; k=40;
km=k/m;

for n=1:3
    cm=c(n)/m;
    [t,x]=ode45('kc',tspan,ic);
    plot(t,x(:,2)); grid;
    xlabel('time - sec.');
    ylabel('displacement - m');
    title('m(d2x/dt2)+c(dx/dt)+kx=0; m=10 kg, k= 40 N/m')
    hold on
end
```

Function 'kc':

```
%Damped spring mass system - m d2x/dt2 + c dx/dt + k x =0
%mass: m=10 kg
%damping: c=5,40,200 N/ (m/s)
%spring: k=40 N/m
%x(1)=velocity, x(2)=displacement

function dx=kc(t,x);
global cm km
dx=[-cm*x(1)-km*x(2); x(1)];
```



- 25.21** The Matlab program shown below performs the Euler Method and displays the time just before the cylindrical tank empties ($y < 0$).

```
%prob2521.m
dt=0.5;
t=0;
y=3;
i=1;
```

```

while(1)
    y=y+dydt(t,y)*dt;
    if y<0, break, end
    t=t+dt;
    i=i+1;
end

function dy=dydt(t,y);
dy=-0.06*sqrt(y);

```

The result is 56 minutes as shown below

```
>> prob2521
```

```
t =
56
```

25.22

$$x = x(1)$$

$$v = \frac{dx}{dt} = x(2)$$

$$\frac{dv}{dt} = \frac{d}{dt} \left(\frac{dx}{dt} \right) = \frac{d^2x}{dt^2}$$

$$\frac{dx(1)}{dt} = x(2)$$

$$\frac{dx(2)}{dt} = -5x(1)x(2) - (x(1) + 7)\sin(t)$$

$$x(1)(t=0) = 6$$

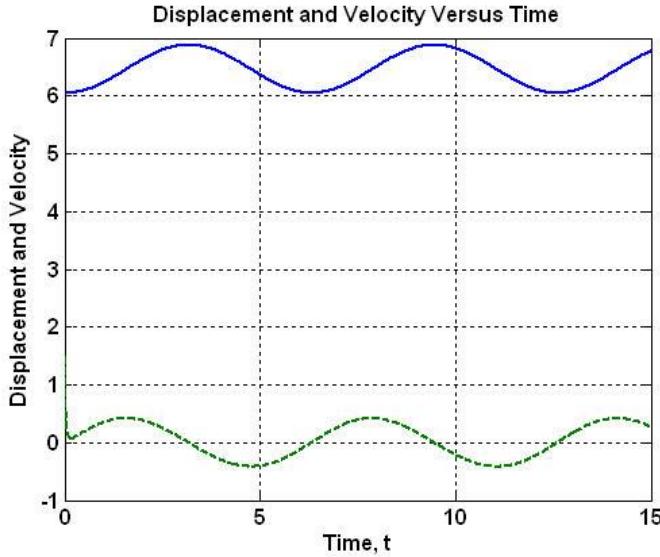
$$x(2)(t=0) = 1.5$$

```

tspan=[0,15]';
x0=[6,1.5]';
[t,x]=ode45('dxdt',tspan,x0);
plot(t,x(:,1),t,x(:,2),'--')
grid
title('Displacement and Velocity Versus Time')
xlabel('Time, t')
ylabel('Displacement and Velocity')

function dx=dxdt(t,x)
dx=[x(2);-5*x(1)*x(2)+(x(1)+7)*sin(1*t)];

```



25.23 The two differential equations to be solved are

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2$$

$$\frac{dx}{dt} = -v$$

(a) Here are the first few steps of Euler's method with a step size of $h = 0.2$.

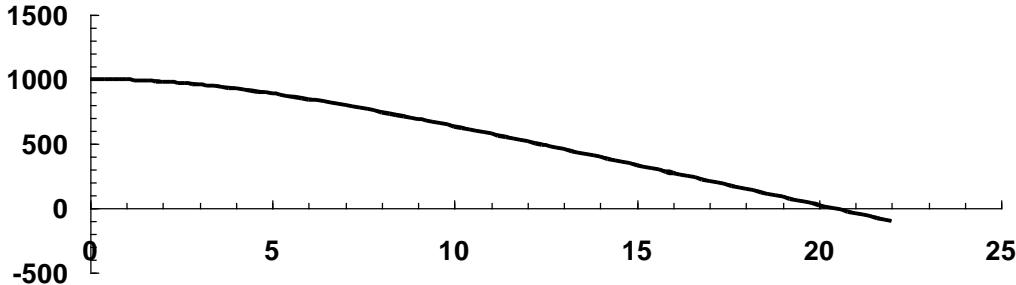
t	x	v	dx/dt	dv/dt
0	1000	0	0	9.81
0.2	1000	1.962	-1.962	9.800376
0.4	999.6076	3.922075	-3.92208	9.771543
0.6	998.8232	5.876384	-5.87638	9.72367
0.8	997.6479	7.821118	-7.82112	9.657075
1	996.0837	9.752533	-9.75253	9.57222

(b) Here are the results of the first few steps of the 4th-order RK method with a step size of $h = 0.2$.

t	x	v
0	1000	0
0.2	999.8038	1.961359
0.4	999.2157	3.918875
0.6	998.2368	5.868738
0.8	996.869	7.807195
1	995.1149	9.730582

The results for x of both methods are displayed graphically on the following plots. Because the step size is sufficiently small the results are in close agreement. Both indicate that the

parachutist would hit the ground at a little after 20 s. The more accurate 4th-order RK method indicates that the solution reaches the ground between $t = 20.2$ and 20.4 s.



25.24 The volume of the tank can be computed as

$$\frac{dV}{dt} = -CA\sqrt{2gH} \quad (1)$$

This equation cannot be solved because it has 2 unknowns: V and H . The volume is related to the depth of liquid by

$$V = \frac{\pi H^2 (3r - H)}{3} \quad (2)$$

Equation (2) can be differentiated to give

$$\frac{dV}{dt} = (2\pi rH - \pi H^2) \frac{dH}{dt} \quad (3)$$

This result can be substituted into Eq. (1) to give and equation with 1 unknown,

$$\frac{dH}{dt} = -\frac{CA\sqrt{2gH}}{2\pi rH - \pi H^2} \quad (4)$$

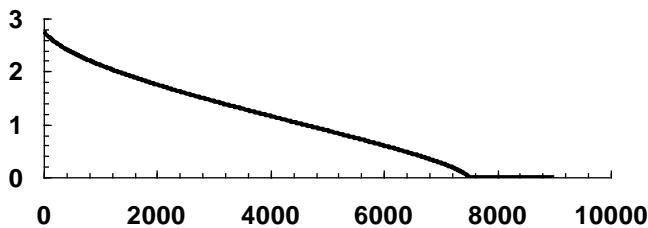
The area of the orifice can be computed as

$$A = \pi(0.015)^2 = 0.000707$$

Substituting this value along with the other parameters ($C = 0.55$, $g = 9.81$, $r = 1.5$) into Eq. (4) gives

$$\frac{dH}{dt} = -0.000548144 \frac{\sqrt{H}}{3H - H^2} \quad (5)$$

We can solve this equation with an initial condition of $H = 2.75$ m using the 4th-order RK method with a step size of 6 s. If this is done, the result can be plotted as shown,

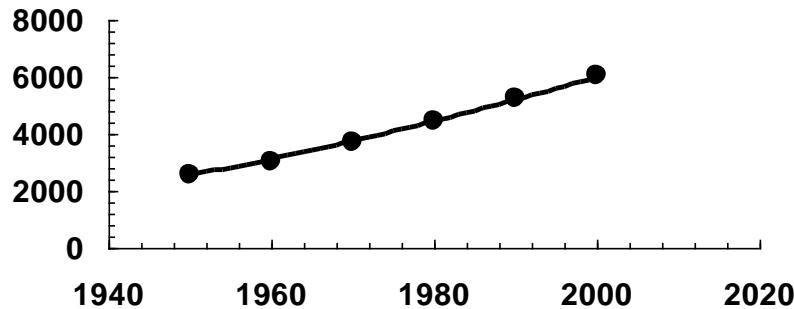


The results indicate that the tank empties at between $t = 7482$ and 7488 seconds.

25.25 The solution can be obtained with the 4th-order RK method with a step size of 1. Here are the results of the first few steps.

<i>x</i>	<i>y</i>
1950	2555
1951	2607.676
1952	2661.132
1953	2715.368
1954	2770.38
1955	2826.168

The entire solution along with the data can be plotted as



25.26 The equations to be integrated are

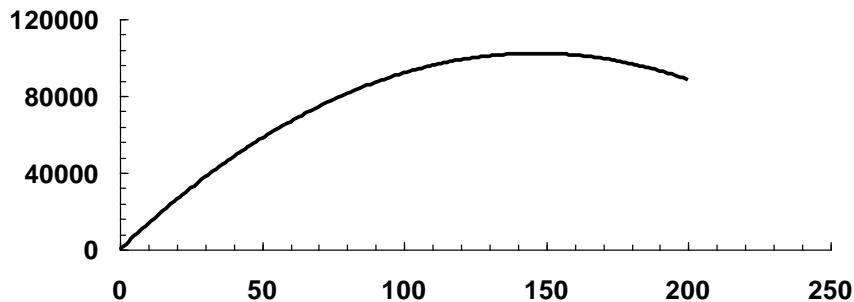
$$\frac{dv}{dt} = -9.81 \frac{(6.37 \times 10^6)^2}{(6.37 \times 10^6 + x)^2}$$

$$\frac{dx}{dt} = v$$

The solution can be obtained with Euler's method with a step size of 1. Here are the results of the first few steps.

<i>t</i>	<i>v</i>	<i>x</i>	<i>dv/dt</i>	<i>dx/dt</i>
0	1400	0	-9.81	1400
1	1390.19	1400	-9.80569	1390.19
2	1380.384	2790.19	-9.80141	1380.384
3	1370.583	4170.574	-9.79717	1370.583
4	1360.786	5541.157	-9.79296	1360.786
5	1350.993	6901.943	-9.78878	1350.993

The entire solution for height can be plotted as



The maximum height occurs at about 146 s.

25.27 First, we must recognize that the evaluation of the definite integral

$$I = \int_a^b f(x) dx$$

is equivalent to solving the differential equation

$$\frac{dy}{dx} = f(x)$$

for $y(b)$ given the initial condition $y(a) = 0$. Thus, we must solve the following initial-value problem:

$$\frac{dy}{dx} = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

where $y(0) = 0$. We can do this in a number of ways. One convenient approach is to use the MATLAB function `ode45` which implements an adaptive RK method. To do this, we must first set up an M-file to evaluate the right-hand side of the differential equation,

```
function dy = humpsODE(x,y)
dy = 1./((x-0.3).^2 + 0.01) + 1./((x-0.9).^2 + 0.04) - 6;
```

Then, the integral can be evaluated as

```
>> [x,y] = ode45(@humpsODE,[0 0.5 1],0);
>> disp([x,y])
          0           0
0.500000000000000  21.78356481821654
1.000000000000000  29.85525185285369
```

Thus, the integral estimate is approximately 29.86.

CHAPTER 26

26.1 (a) $h < 2/200,000 = 1 \times 10^{-5}$.

(b) The implicit Euler can be written for this problem as

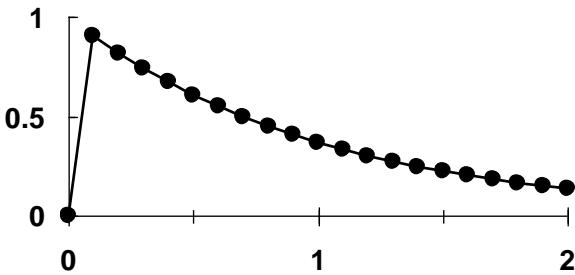
$$y_{i+1} = y_i + (-200,000 y_{i+1} + 200,000 e^{-x_{i+1}} - e^{-x_{i+1}})h$$

which can be solved for

$$y_{i+1} = \frac{y_i + 200,000 e^{-x_{i+1}} h - e^{-x_{i+1}} h}{1 + 200,000 h}$$

The results of applying this formula for the first few steps are shown below. A plot of the entire solution is also displayed

x	y
0	0
0.1	0.904788
0.2	0.818731
0.3	0.740818
0.4	0.670320
0.5	0.606531



26.2 The implicit Euler can be written for this problem as

$$y_{i+1} = y_i + (30(\cos t_{i+1} - y_{i+1}) + 3 \sin t_{i+1})h$$

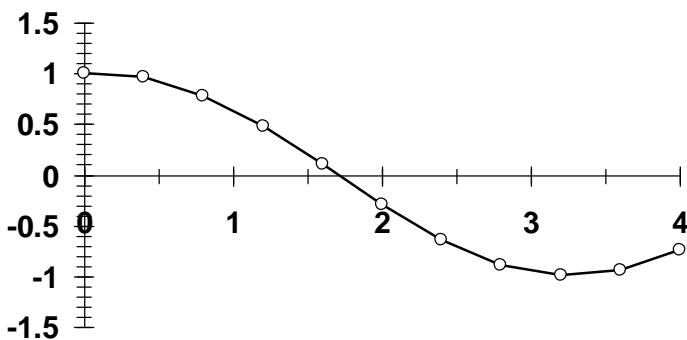
which can be solved for

$$y_{i+1} = \frac{y_i + 30 \cos t_{i+1} h + 3 \sin t_{i+1} h}{1 + 30h}$$

The results of applying this formula are tabulated and graphed below.

x	y
0	1

0.4	0.96308
0.8	0.783414
1.2	0.480781
1.6	0.102298
2	-0.29233
2.4	-0.64081
2.8	-0.88811
3.2	-0.99521
3.6	-0.94518
4	-0.74593



26.3 (a) The explicit Euler can be written for this problem as

$$\begin{aligned}x_{1,i+1} &= x_{1,i} + (1999x_{1,i} + 2999x_{2,i})h \\x_{2,i+1} &= x_{2,i} + (2000x_{1,i} - 3000x_{2,i})h\end{aligned}$$

Because the step-size is much too large for the stability requirements, the solution is unstable,

t	x₁	x₂	dx₁/dt	dx₂/dt
0	1	1	4998	-5000
0.05	250.9	-249	-245202	245200
0.1	-12009.2	12011	12014608	-1.2E+07
0.15	588721.2	-588720	-5.9E+08	5.89E+08
0.2	-2.9E+07	28847084	2.88E+10	-2.9E+10

(b) The implicit Euler can be written for this problem as

$$\begin{aligned}x_{1,i+1} &= x_{1,i} + (1999x_{1,i+1} + 2999x_{2,i+1})h \\x_{2,i+1} &= x_{2,i} + (-2000x_{1,i+1} - 3000x_{2,i+1})h\end{aligned}$$

or collecting terms

$$\begin{aligned}(1 - 1999h)x_{1,i+1} - 2999hx_{2,i+1} &= x_{1,i} \\2000hx_{1,i+1} + (1 + 3000h)x_{2,i+1} &= x_{2,i}\end{aligned}$$

or substituting $h = 0.05$ and expressing in matrix format

$$\begin{bmatrix} -98.95 & -149.95 \\ 100 & 151 \end{bmatrix} \begin{Bmatrix} x_{1,i+1} \\ x_{2,i+1} \end{Bmatrix} = \begin{Bmatrix} x_{1,i} \\ x_{2,i} \end{Bmatrix}$$

Thus, to solve for the first time step, we substitute the initial conditions for the right-hand side and solve the 2×2 system of equations. The best way to do this is with *LU* decomposition since we will have to solve the system repeatedly. For the present case, because it's easier to display, we will use the matrix inverse to obtain the solution. Thus, if the matrix is inverted, the solution for the first step amounts to the matrix multiplication,

$$\begin{Bmatrix} x_{1,i+1} \\ x_{2,i+1} \end{Bmatrix} = \begin{bmatrix} 2.819795 & 2.800187 \\ -1.86741 & -1.84781 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 5.619981 \\ -3.71522 \end{Bmatrix}$$

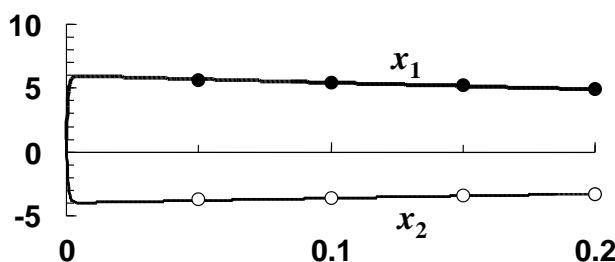
For the second step (from $x = 0.05$ to 0.1),

$$\begin{Bmatrix} x_{1,i+1} \\ x_{2,i+1} \end{Bmatrix} = \begin{bmatrix} 2.819795 & 2.800187 \\ -1.86741 & -1.84781 \end{bmatrix} \begin{Bmatrix} 5.619981 \\ -3.71522 \end{Bmatrix} = \begin{Bmatrix} 5.443885 \\ -3.62983 \end{Bmatrix}$$

The remaining steps can be implemented in a similar fashion to give

t	x₁	x₂
0	1	1
0.05	5.619981	-3.71522
0.1	5.443885	-3.62983
0.15	5.186447	-3.45877
0.2	4.939508	-3.2941

The results are plotted below, along with a solution with the explicit Euler using a step of 0.0005.



26.4 The analytical solution is

$$y = 10.625e^{-0.4t} - 0.625e^{-2t}$$

Therefore, the exact results are $y(2.5) = 3.904508$ and $y(3) = 3.198639$.

First step:

Predictor:

$$y_1^0 = 5.800007 + [-0.4(4.762673) + e^{-2(2)}]1 = 3.913253$$

Corrector:

$$y_1^1 = 4.762673 + \frac{-0.4(4.762673) + e^{-2(2)} - 0.4(3.913253) + e^{-2(2.5)}}{2} 0.5 = 3.901344$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \varepsilon_a , \%$
1	3.901344	
2	3.902534	0.0305

The true error can be computed as

$$\varepsilon_t = \left| \frac{3.904508 - 3.902534}{3.904508} \right| \times 100\% = 0.0506\%$$

Second step:

Predictor:

$$y_2^0 = 4.762673 + [-0.4(3.902534) + e^{-2(2.5)}]1 = 3.208397$$

Predictor Modifier:

$$y_2^0 = 3.208397 + 4/5(3.902534 - 3.913253) = 3.199822$$

Corrector:

$$y_2^1 = 3.902534 + \frac{-0.4(3.902534) + e^{-2(2.5)} - 0.4(3.199822) + e^{-2(3)}}{2} 0.5 = 3.194603$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \varepsilon_a , \%$
1	3.194603	
2	3.195125	0.0163

The true error can be computed as

$$\varepsilon_t = \left| \frac{3.198639 - 3.195125}{3.198639} \right| \times 100\% = 0.11\%$$

26.5 The analytical solution is

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$y = 10.625e^{-0.4t} - 0.625e^{-2t}$$

Therefore, the exact results are $y(2.5) = 3.904508$ and $y(3) = 3.198639$.

The Adams method can be implemented with the results

```

predictor =      3.8918505
Corrector Iteration
x           y           ea
2.5         3.905861533  3.59E-01
2.5         3.904810706  2.69E-02
2.5         3.904889518  2.02E-03

predictor =      3.194487763
Corrector Iteration
x           y           ea
3           3.1994013   1.54E-01
3           3.199032785  1.15E-02
3           3.199060424  8.64E-04

```

Therefore, the true percent relative errors can be computed as

$$\varepsilon_t = \left| \frac{3.904508 - 3.9048895}{3.904508} \right| \times 100\% = 0.0098\%$$

$$\varepsilon_t = \left| \frac{3.198639 - 3.19906}{3.198639} \right| \times 100\% = 0.0132\%$$

26.6 (a) Non-self-starting Heun

First step:

Predictor:

$$y_1^0 = 0.244898 + [-2(0.1875)/4]1 = 0.151148$$

Corrector:

$$y_1^1 = 0.1875 + \frac{-2(0.1875)/4 - 2(0.151148)/4.5}{2} 0.5 = 0.1472683$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \varepsilon_a , \%$
1	0.1472683	2.63
2	0.1476994	0.292

The true error can be computed as

$$\varepsilon_t = \left| \frac{0.148148 - 0.1476994}{0.148148} \right| \times 100\% = 0.3\%$$

Second step:

Predictor:

$$y_2^0 = 0.1875 + [-2(0.1476994)/4.5]1 = 0.1218558$$

Predictor Modifier:

$$y_2^0 = 0.1218558 + 4/5(0.1476994 - 0.151148) = 0.1190969$$

Corrector:

$$y_2^1 = 0.1476994 + \frac{-2(0.1476994)/4.5 - 2(0.1190969)/5}{2} 0.5 = 0.1193786$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \varepsilon_a , \%$
1	0.1193786	0.236

The true error can be computed as

$$\varepsilon_t = \left| \frac{0.12 - 0.1193786}{0.12} \right| \times 100\% = 0.52\%$$

(b) Adams method

```

predictor =      0.152793519
Corrector Iteration
x          y                  ea          et
4.5        0.147605464    3.51E+00    0.366%
4.5        0.148037802    2.92E-01    0.074%
4.5        0.148001774    2.43E-02    0.099%
4.5        0.148004776    0.002028547   0.097%

predictor =      0.121661277
Corrector Iteration
x          y                  ea          et
5          0.119692476    1.64E+00    0.256%
5          0.119840136    1.23E-01    0.133%
5          0.119829061    9.24E-03    0.142%

```

26.7 The 4th-order RK method can be used to compute $y(0.25) = 0.7828723$. Then, the non-self-starting Heun method can be implemented as

First step:

Predictor:

$$y_1^0 = 1 + f(0.25, 0.7828723)(0.5) = 0.6330286$$

Corrector:

$$y_1^1 = 0.7828723 + \frac{f(0.25, 0.7828723) + f(0.5, 0.6330286)}{2} 0.25 = 0.6317830$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \epsilon_a , \%$
1	0.6317830	0.197
2	0.6318998	0.018
3	0.6318888	0.0017
4	0.6318898	0.00016
5	0.6318897	0.000015

Second step:

Predictor:

$$y_2^0 = 0.7828723 + f(0.5, 0.6318897)(0.5) = 0.5458282$$

Predictor Modifier:

$$y_2^0 = 0.5458282 + 4/5(0.6318897 - 0.6330286) = 0.5449171$$

Corrector:

$$y_2^1 = 0.6318897 + \frac{f(0.5, 0.6318897) + f(0.75, 0.5449171)}{2} 0.25 = 0.5430563$$

The corrector can be iterated to yield

j	y_{i+1}^j	$ \epsilon_a , \%$
1	0.5430563	0.343
2	0.5431581	0.0187
3	0.5431525	0.001
4	0.5431528	0.000056

26.8 The fourth-order RK method can be used to determine the following values:

t	y
0	2
0.5	0.893333
1	0.50288
1.5	0.321905

The 4th-order Adams method can then be implemented with the results summarized below:

```
predictor = 0.476800394
Corrector Iteration
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

x          y          ea
2          0.187936982  1.54E+02
2          0.224044908  1.61E+01
2          0.219531418  2.06E+00
2          0.220095604  0.256336947
2          0.220025081  0.032052389
2          0.220033896  0.004006388

predictor = 0.204189336
Corrector Iteration
x          y          ea
2.5        0.156440735  3.05E+01
2.5        0.161556656  3.166642375
2.5        0.161008522  0.340438161
2.5        0.16106725   0.036462217
2.5        0.161060958  0.003906819

```

26.9

Option Explicit

```

Sub SimpImplTest()
Dim i As Integer, m As Integer
Dim xi As Double, yi As Double, xf As Double, dx As Double, xout As Double
Dim xp(200) As Double, yp(200) As Double
'Assign values
yi = 0
xi = 0
xf = 2
dx = 0.1
xout = 0.1
'Perform numerical Integration of ODE
Call ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
'Display results
Sheets("Sheet1").Select
Range("a5:b205").ClearContents
Range("a5").Select
For i = 0 To m
    ActiveCell.Value = xp(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = yp(i)
    ActiveCell.Offset(1, -1).Select
Next i
Range("a5").Select
End Sub

Sub ODESolver(xi, yi, xf, dx, xout, xp, yp, m)
'Generate an array that holds the solution
Dim x As Double, y As Double, xend As Double
Dim h As Double
m = 0
xp(m) = xi
yp(m) = yi
x = xi
y = yi
Do      'Print loop
    xend = x + xout
    If (xend > xf) Then xend = xf  'Trim step if increment exceeds end
    h = dx
    Call Integrator(x, y, h, xend)

```

```

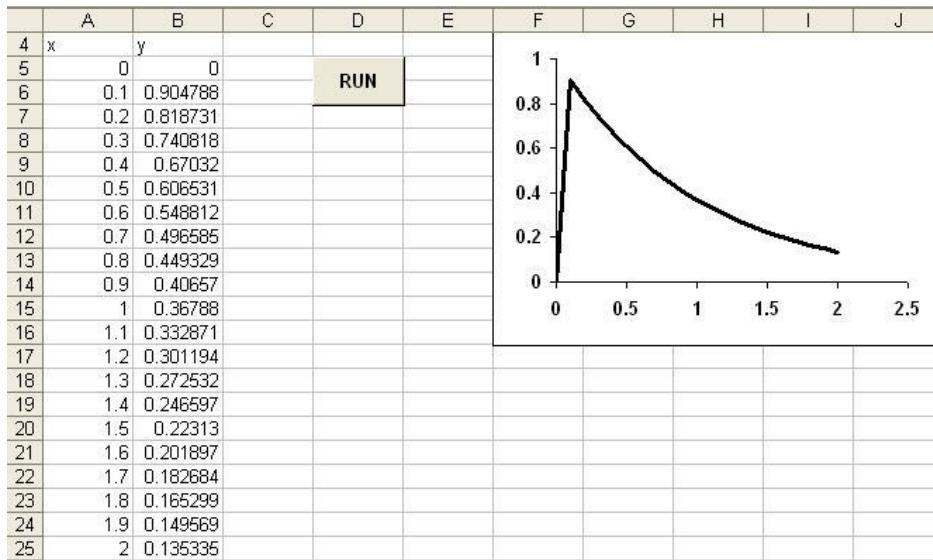
m = m + 1
xp(m) = x
yp(m) = y
If (x >= xf) Then Exit Do
Loop
End Sub

Sub Integrator(x, y, h, xend)
Dim ynew As Double
Do          'Calculation loop
    If (xend - x < h) Then h = xend - x  'Trim step if increment exceeds end
    Call SimpImpl(x, y, h, ynew)
    y = ynew
    If (x >= xend) Then Exit Do
Loop
End Sub

Sub SimpImpl(x, y, h, ynew)
'Implement implicit Euler's method
ynew = (y + h * FF(x + h)) / (1 + 200000 * h)
x = x + h
End Sub

Function FF(x)
'Define Forcing Function
FF = 200000 * Exp(-x) - Exp(-x)
End Function

```



26.10 All linear systems are of the form

$$\frac{dy_1}{dt} = a_{11}y_1 + a_{12}y_2 + F_1$$

$$\frac{dy_2}{dt} = a_{21}y_1 + a_{22}y_2 + F_2$$

As shown in the book (p. 730), the implicit approach amounts to solving

$$\begin{bmatrix} 1-a_{11}h & -a_{12} \\ -a_{21} & 1-a_{22}h \end{bmatrix} \begin{Bmatrix} y_{1,i+1} \\ y_{2,i+1} \end{Bmatrix} = \begin{Bmatrix} y_{1,i} + F_1 h \\ y_{2,i} + F_2 h \end{Bmatrix}$$

Therefore, for Eq. 26.6: $a_{11} = -5$, $a_{12} = 3$, $a_{21} = 100$, $a_{22} = -301$, $F_1 =$, and $F_2 = 0$,

$$\begin{bmatrix} 1+5h & -3 \\ -100 & 1+301h \end{bmatrix} \begin{Bmatrix} y_{1,i+1} \\ y_{2,i+1} \end{Bmatrix} = \begin{Bmatrix} y_{1,i} \\ y_{2,i} \end{Bmatrix}$$

A VBA program written in these terms is

```

Option Explicit

Sub StiffSysTest()
Dim i As Integer, m As Integer, n As Integer, j As Integer
Dim xi As Single, yi(10) As Single, xf As Single, dx As Single, xout As
Single
Dim xp(200) As Single, yp(200, 10) As Single

'Assign values
n = 2
xi = 0
xf = 0.4
yi(1) = 52.29
yi(2) = 83.82
dx = 0.05
xout = 0.05

'Perform numerical Integration of ODE
Call ODESolver(xi, yi(), xf, dx, xout, xp(), yp(), m, n)

'Display results
Sheets("Sheet1").Select
Range("a5:n205").ClearContents
Range("a5").Select
For i = 0 To m
    ActiveCell.Value = xp(i)
    For j = 1 To n
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yp(i, j)
    Next j
    ActiveCell.Offset(1, -n).Select
Next i
Range("a5").Select
End Sub

Sub ODESolver(xi, yi, xf, dx, xout, xp, yp, m, n)
'Generate an array that holds the solution
Dim i As Integer
Dim x As Single, y(10) As Single, xend As Single
Dim h As Single
m = 0
x = xi
'set initial conditions
For i = 1 To n
    y(i) = yi(i)
Next i

```

```

'save output values
xp(m) = x
For i = 1 To n
    yp(m, i) = y(i)
Next i
Do      'Print loop
    xend = x + xout
    If (xend > xf) Then xend = xf  'Trim step if increment exceeds end
    h = dx
    Call Integrator(x, y(), h, n, xend)
    m = m + 1
    'save output values
    xp(m) = x
    For i = 1 To n
        yp(m, i) = y(i)
    Next i
    If (x >= xf) Then Exit Do
Loop
End Sub

Sub Integrator(x, y, h, n, xend)
Dim j As Integer
Dim ynew(10) As Single
Do      'Calculation loop
    If (xend - x < h) Then h = xend - x  'Trim step if increment exceeds end
    Call StiffSys(x, y, h, n, ynew())
    For j = 1 To n
        y(j) = ynew(j)
    Next j
    If (x >= xend) Then Exit Do
Loop
End Sub

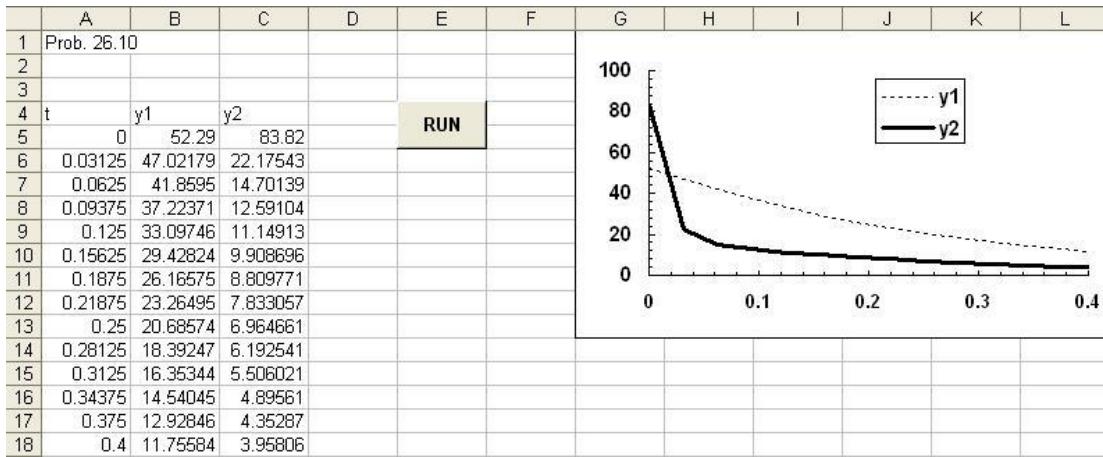
Sub StiffSys(x, y, h, n, ynew)
Dim j As Integer
Dim FF(2) As Single, b(2, 2) As Single, c(2) As Single, den As Single
Call Force(x, FF())
'MsgBox "pause"

b(1, 1) = 1 + 5 * h
b(1, 2) = -3 * h
b(2, 1) = -100 * h
b(2, 2) = 1 + 301 * h
For j = 1 To n
    c(j) = y(j) + FF(j) * h
Next j
den = b(1, 1) * b(2, 2) - b(1, 2) * b(2, 1)
ynew(1) = (c(1) * b(2, 2) - c(2) * b(1, 2)) / den
ynew(2) = (c(2) * b(1, 1) - c(1) * b(2, 1)) / den
x = x + h
End Sub

Sub Force(t, FF)
'Define Forcing Function
FF(0) = 0
FF(1) = 0
End Sub

```

The result compares well with the analytical solution. If a smaller step size were used, the solution would improve



26.11

Option Explicit

```

Sub NonSelfStartHeun()
    Dim n As Integer, m As Integer, i As Integer, iter As Integer
    Dim xi As Double, xf As Double, yi As Double, h As Double
    Dim x As Double, y As Double
    Dim xp(1000) As Double, yp(1000) As Double
    xi = -1
    xf = 4
    yi = -0.3929953
    h = 1
    n = (xf - xi) / h
    x = xi
    y = yi
    m = 0
    xp(m) = x
    yp(m) = y
    Call RK4(x, y, h)
    m = m + 1
    xp(m) = x
    yp(m) = y
    For i = 2 To n
        Call NSSHeun(xp(i - 2), yp(i - 2), xp(i - 1), yp(i - 1), x, y, h, iter)
        m = m + 1
        xp(m) = x
        yp(m) = y
    Next i
    Sheets("NSS Heun").Select
    Range("a5:b1005").ClearContents
    Range("a5").Select
    For i = 0 To m
        ActiveCell.Value = xp(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yp(i)
        ActiveCell.Offset(1, -1).Select
    Next i
    Range("a5").Select
End Sub

Sub RK4(x, y, h)
    ' Implement RK4 method

```

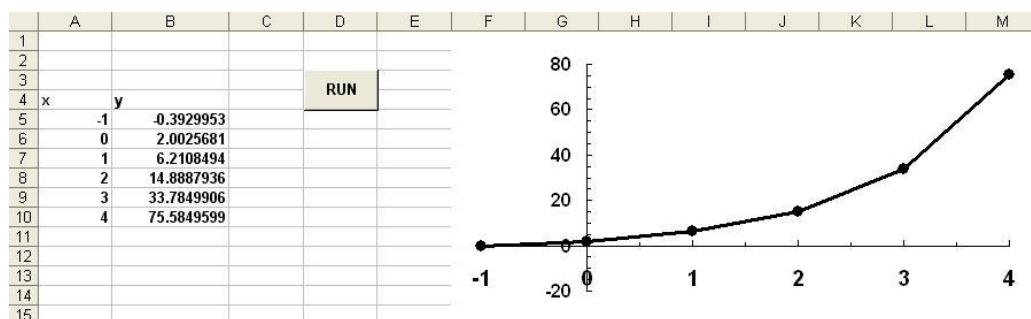
```

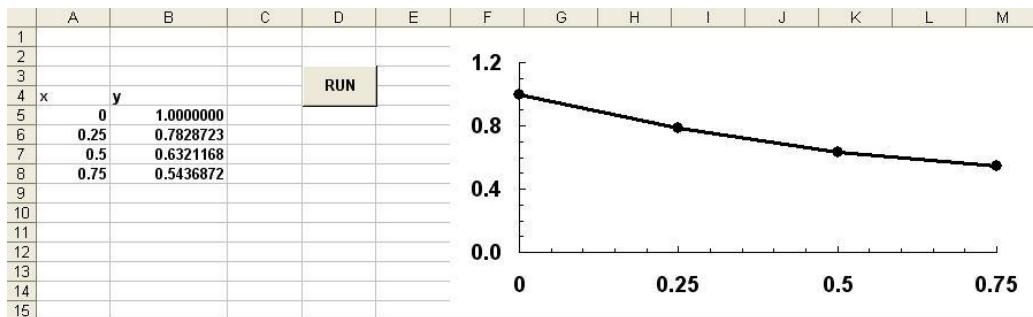
Dim k1 As Double, k2 As Double, k3 As Double, k4 As Double
Dim ym As Double, ye As Double, slope As Double
Call Derivs(x, y, k1)
ym = y + k1 * h / 2
Call Derivs(x + h / 2, ym, k2)
ym = y + k2 * h / 2
Call Derivs(x + h / 2, ym, k3)
ye = y + k3 * h
Call Derivs(x + h, ye, k4)
slope = (k1 + 2 * (k2 + k3) + k4) / 6
y = y + slope * h
x = x + h
End Sub

Sub NSSHeun(x0, y0, x1, y1, x, y, h, iter)
'Implement Non Self-Starting Heun
Dim i As Integer
Dim y2 As Double
Dim slope As Double, k1 As Double, k2 As Double
Dim ea As Double
Dim y2p As Double
Static y2old As Double, y2pold As Double
Call Derivs(x1, y1, k1)
y2 = y0 + k1 * 2 * h
y2p = y2
If iter > 0 Then
    y2 = y2 + 4 * (y2old - y2pold) / 5
End If
x = x + h
iter = 0
Do
    y2old = y2
    Call Derivs(x, y2, k2)
    slope = (k1 + k2) / 2
    y2 = y1 + slope * h
    iter = iter + 1
    ea = Abs((y2 - y2old) / y2) * 100
    If ea < 0.01 Then Exit Do
Loop
y = y2 - (y2 - y2p) / 5
y2old = y2
y2pold = y2p
End Sub

Sub Derivs(x, y, dydx)
'Define ODE
dydx = 4 * Exp(0.8 * x) - 0.5 * y
End Sub

```



26.12

26.13 The second-order equation can be composed into a pair of first-order equations as

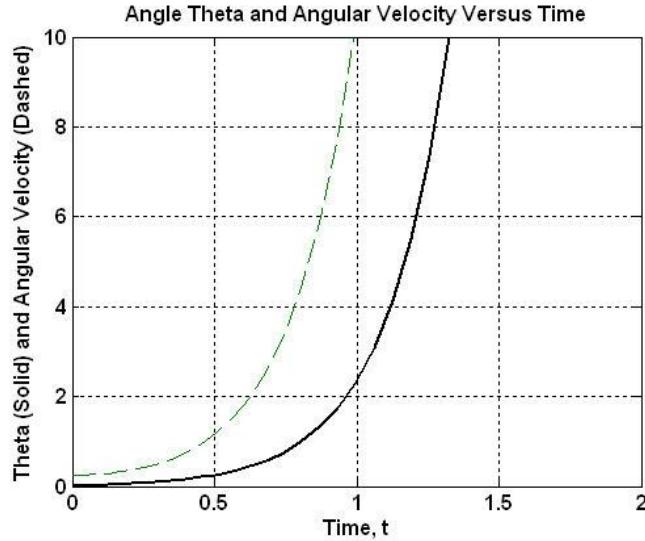
$$\frac{d\theta}{dt} = x$$

$$\frac{dx}{dt} = \frac{g}{l} \theta$$

We can use MATLAB to solve this system of equations.

```
tspan=[0,5]';
x0=[0,0.25]';
[t,x]=ode45('dxdt',tspan,x0);
plot(t,x(:,1),t,x(:,2),'--')
grid
title('Angle Theta and Angular Velocity Versus Time')
xlabel('Time, t')
ylabel('Theta (Solid) and Angular Velocity (Dashed)')
axis([0 2 0 10])
zoom

function dx=dxdt(t,x)
dx=[x(2); (9.81/0.5)*x(1)];
```



26.14 Analytic solution: Take Laplace transform

$$sX - x(0) = -700X - \frac{1000}{s+1}$$

$$sX + 700X = x(0) - \frac{1000}{s+1}$$

$$X = \frac{x(0)}{s+700} - \frac{1000}{(s+1)(s+700)}$$

Substituting the initial condition and expanding the last term with partial fractions gives,

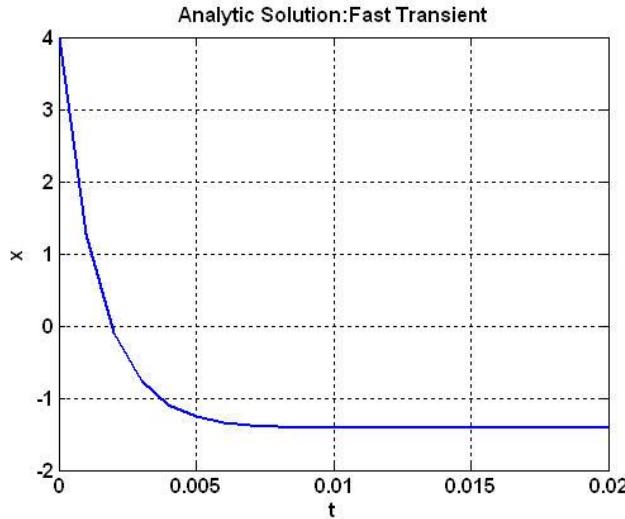
$$X = \frac{4}{s+700} - \frac{1.430615}{s+1} + \frac{1.430615}{s+700}$$

Taking inverse transforms yields

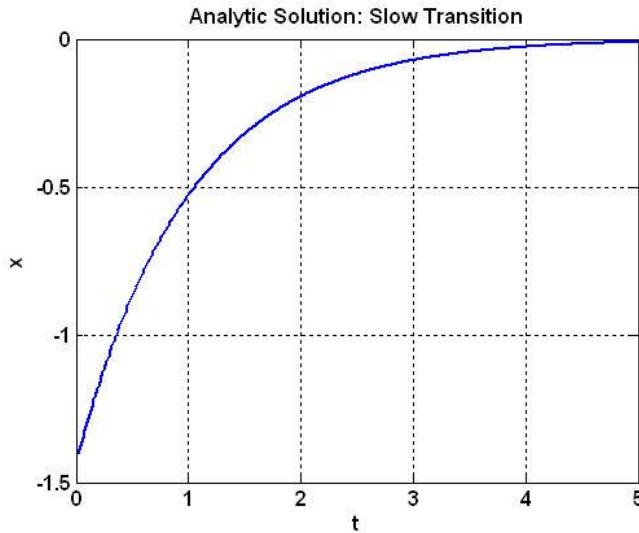
$$x = 5.430615e^{-700t} - 1.430615e^{-t}$$

MATLAB can be used to plot both the fast transient and the slow phases of the analytical solution.

```
t=[0:.001:.02];
x=5.430615*exp(-700*t)-1.430615*exp(-t);
plot(t,x)
grid
xlabel('t')
ylabel('x')
title('Analytic Solution:Fast Transient')
```



```
>> t=[0.02::01:5];
>> x=5.430615*exp(-700*t)-1.430615*exp(-t);
>> plot(t,x)
>> grid
>> xlabel('t')
>> ylabel('x')
>> title('Analytic Solution: Slow Transition')
```



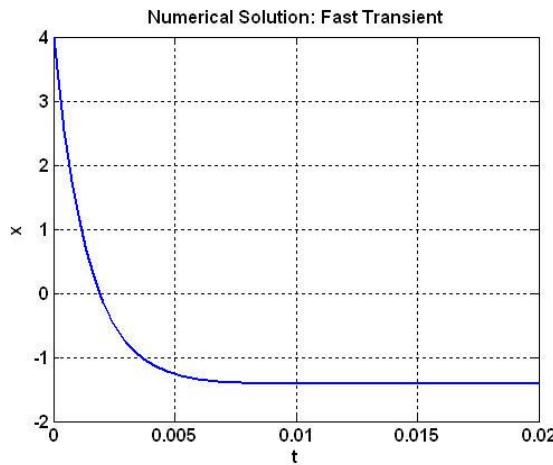
Numerical solution: First set up the function holding the differential equation:

```
function dx=dxdt(t,x)
dx=-700*x-1000*exp(-t);
```

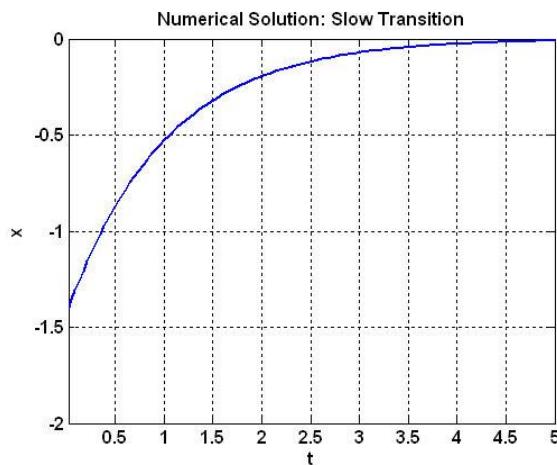
The following MATLAB code then generates the solutions and plots:

```
tspan=[0 5];
x0=[4];
```

```
[t,x]=ode23s(@dxdt,tspan,x0);
plot(t,x)
grid
xlabel('t')
ylabel('x')
title('Numerical Solution: Fast Transient')
axis([0 .02 -2 4])
```



```
tspan=[0 5];
x0=[4];
[t,x]=ode23s(@dxdt,tspan,x0);
plot(t,x)
grid
xlabel('t')
ylabel('x')
title('Numerical Solution: Slow Transition')
%axis([0.02 5 -2 0])
```



26.15 (a) Analytic solution:

$$y = \frac{1}{999} (1000e^{-x} - e^{-1000x})$$

(b) The second-order differential equation can be expressed as the following pair of first-order ODEs,

$$\frac{dy}{dx} = w$$

$$\frac{dw}{dx} = -1000y - 1001w$$

where $w = y'$. Using the same approach as described in Sec. 26.1, the following simultaneous equations need to be solved to advance each time step,

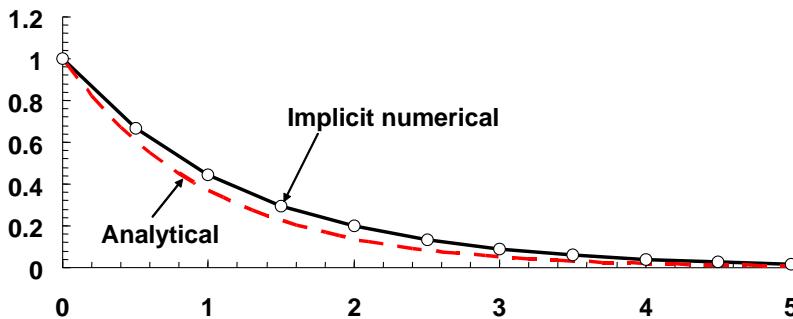
$$y_{i+1} - hw_{i+1} = y_i$$

$$1000hy_{i+1} + (1001h + 1)w_{i+1} = w_i$$

If these are implemented with a step size of 0.5, the following values are simulated

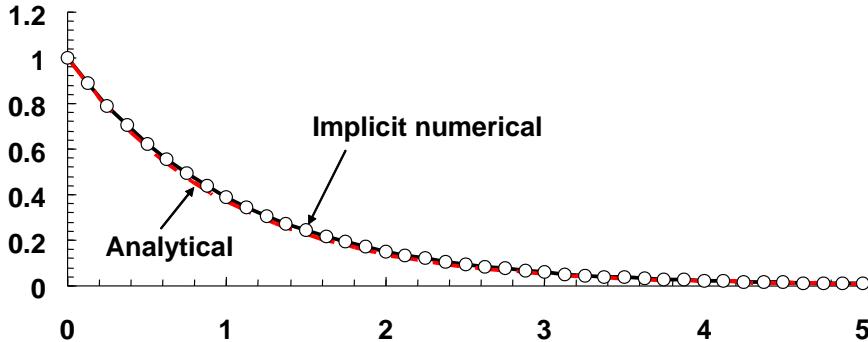
x	y	w
0	1	0
0.5	0.667332	-0.66534
1	0.444889	-0.44489
1.5	0.296593	-0.29659
2	0.197729	-0.19773
2.5	0.131819	-0.13182
3	0.087879	-0.08788
3.5	0.058586	-0.05859
4	0.039057	-0.03906
4.5	0.026038	-0.02604
5	0.017359	-0.01736

The results for y along with the analytical solution are displayed below:



Note that because we are using an implicit method the results are stable. However, also notice that the results are somewhat inaccurate. This is due to the large step size. If we use a smaller

step size, the results will converge on the analytical solution. For example, if we use $h = 0.125$, the results are:

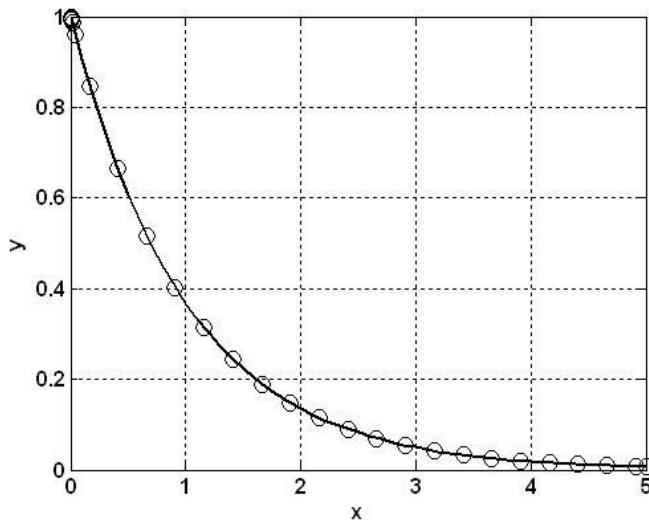


Finally, we can also solve this problem using one of the MATLAB routines expressly designed for stiff systems. To do this, we first develop a function to hold the pair of ODEs,

```
function dy = dydx(x, y)
dy = [y(2); -1000*y(1)-1001*y(2)];
```

Then the following session generates a plot of both the analytical and numerical solutions. As can be seen, the results are indistinguishable.

```
x=[0:.1:5];
y=1/999*(1000*exp(-x)-exp(-1000*x));
xspan=[0 5];
x0=[1 0];
[xx,yy]=ode23s(@dydx,xspan,x0);
plot(x,y,xx,yy(:,1),'o')
grid
xlabel('x')
ylabel('y')
```



26.16 (a) Analytic solution:

$$y = 1e^{-10t}$$

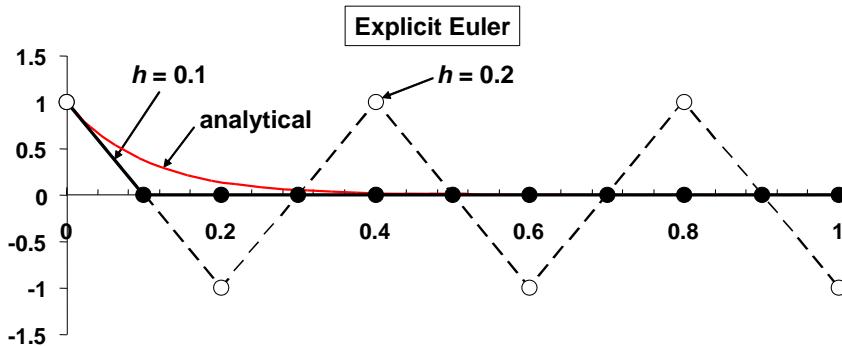
(b) Explicit Euler

$$y_{i+1} = y_i + (-10y_i)h$$

Here are the results for $h = 0.2$. Notice that the solution oscillates:

t	y	dy/dt
0	1	-10
0.2	-1	10
0.4	1	-10
0.6	-1	10
0.8	1	-10
1	-1	10

For $h = 0.1$, the solution plunges abruptly to 0 at $t = 0.1$ and then stays at zero thereafter. Both results are displayed along with the analytical solution below:



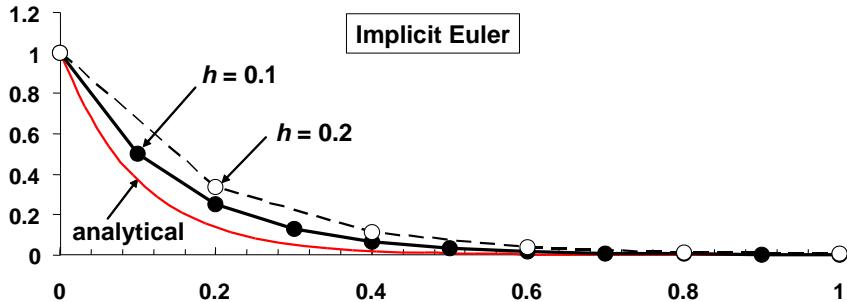
(c) Implicit Euler

$$y_{i+1} = \frac{y_i}{1 + 10h}$$

Here are the results for $h = 0.2$. Notice that although the solution is not very accurate, it is stable and declines monotonically in a similar fashion to the analytical solution.

t	y
0	1
0.2	0.333333333
0.4	0.111111111
0.6	0.037037037
0.8	0.012345679
1	0.004115226

For $h = 0.1$, the solution is also stable and tracks closer to the analytical solution. Both results are displayed along with the analytical solution below:



CHAPTER 27

27.1 The solution can be assumed to be $T = e^{\lambda x}$. This, along with the second derivative $T'' = \lambda^2 e^{\lambda x}$, can be substituted into the differential equation to give

$$\lambda^2 e^{\lambda x} - 0.15 e^{\lambda x} = 0$$

which can be used to solve for

$$\begin{aligned}\lambda^2 - 0.15 &= 0 \\ \lambda &= \pm\sqrt{0.15}\end{aligned}$$

Therefore, the general solution is

$$T = Ae^{\sqrt{0.15}x} + Be^{-\sqrt{0.15}x}$$

The constants can be evaluated by substituting each of the boundary conditions to generate two equations with two unknowns,

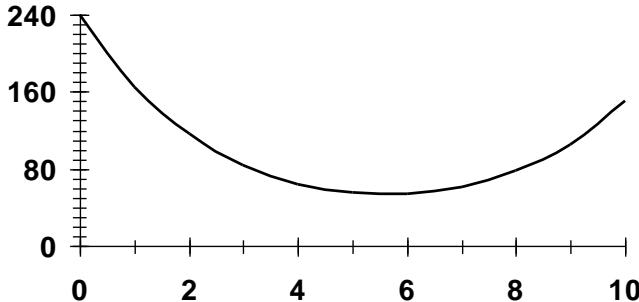
$$\begin{aligned}240 &= A + B \\ 150 &= 48.08563A + 0.020796B\end{aligned}$$

which can be solved for $A = 3.016944$ and $B = 236.9831$. The final solution is, therefore,

$$T = 3.016944e^{\sqrt{0.15}x} + 236.9831e^{-\sqrt{0.15}x}$$

which can be used to generate the values below:

x	T
0	240
1	165.329
2	115.7689
3	83.79237
4	64.54254
5	55.09572
6	54.01709
7	61.1428
8	77.55515
9	105.7469
10	150



27.2 Reexpress the second-order equation as a pair of ODEs:

$$\frac{dT}{dx} = z$$

$$\frac{dz}{dx} = 0.15T$$

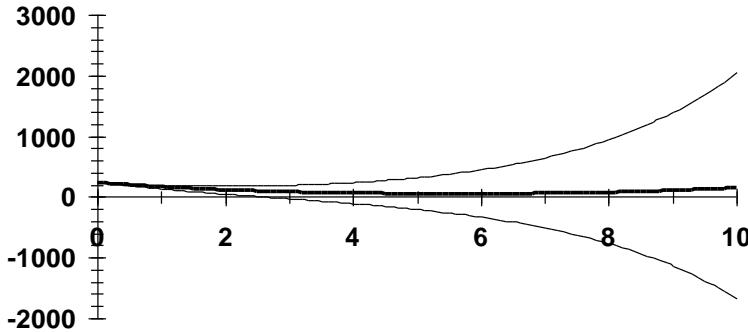
The solution was then generated on the Excel spreadsheet using the Heun method (without iteration) with a step-size of 0.01. An initial condition of $z = -120$ was chosen for the first shot. The first few calculation results are shown below.

x	T	z	k_{11}	k_{12}	T_{end}	z_{end}	k_{21}	k_{22}	ϕ_1	ϕ_2
0	240.000	-120.000	-120.000	36.000	228.000	-116.400	-116.400	34.200	-118.200	35.100
0.1	228.180	-116.490	-116.490	34.227	216.531	-113.067	-113.067	32.480	-114.779	33.353
0.2	216.702	-113.155	-113.155	32.505	205.387	-109.904	-109.904	30.808	-111.529	31.657
0.3	205.549	-109.989	-109.989	30.832	194.550	-106.906	-106.906	29.183	-108.447	30.007
0.4	194.704	-106.988	-106.988	29.206	184.006	-104.068	-104.068	27.601	-105.528	28.403
0.5	184.152	-104.148	-104.148	27.623	173.737	-101.386	-101.386	26.061	-102.767	26.842

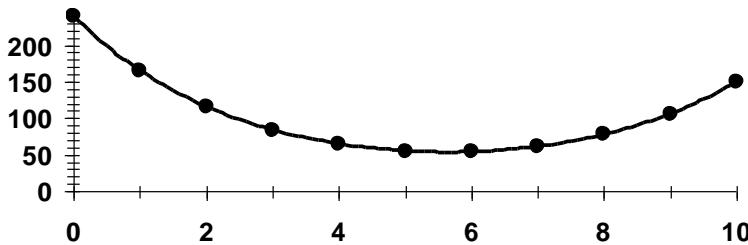
The resulting value at $x = 10$ was $T(10) = -1671.817$. A second shot using an initial condition of $z(0) = -60$ was attempted with the result at $x = 10$ of $T(10) = 2047.766$. These values can then be used to derive the correct initial condition,

$$z(0) = -120 + \frac{-60 + 120}{2047.766 - (-1671.817)} (150 - (-1671.817)) = -90.6126$$

The resulting fit, along with the two “shots” are displayed below:



The final shot along with the analytical solution (displayed as filled circles) shows close agreement:



27.3 A centered finite difference can be substituted for the second derivative to give,

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} - 0.15T_i = 0$$

or for $h = 1$,

$$-T_{i-1} + 2.15T_i - T_{i+1} = 0$$

The first node would be

$$2.15T_1 - T_2 = 240$$

and the last node would be

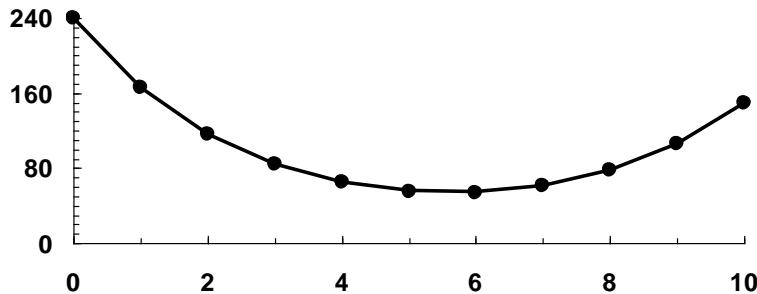
$$-T_9 + 2.15T_{10} = 150$$

The tridiagonal system can be solved with the Thomas algorithm or Gauss-Seidel for (the analytical solution is also included)

x	T	Analytical
0	240	240
1	165.7573	165.3290
2	116.3782	115.7689

3	84.4558	83.7924
4	65.2018	64.5425
5	55.7281	55.0957
6	54.6136	54.0171
7	61.6911	61.1428
8	78.0223	77.5552
9	106.0569	105.7469
10	150	150

The following plot of the results (with the analytical shown as filled circles) indicates close agreement.



27.4 The second-order ODE can be expressed as the following pair of first-order ODEs,

$$\begin{aligned}\frac{dy}{dx} &= z \\ \frac{dz}{dx} &= \frac{2z + y - x}{7}\end{aligned}$$

These can be solved for two guesses for the initial condition of z . For our cases we used -1 and -0.5 . We solved the ODEs with the Heun method without iteration using a step size of 0.125. The results are

$z(0)$	-1	-0.5
$y(20)$	-11,837.64486	22,712.34615

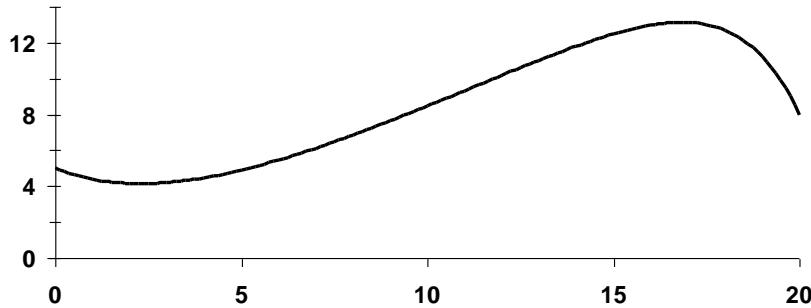
Clearly, the solution is quite sensitive to the initial conditions. These values can then be used to derive the correct initial condition,

$$z(0) = -1 + \frac{-0.5 + 1}{22712.34615 - (-11837.64486)} (8 - (-11837.64486)) = -0.82857239$$

The resulting fit is displayed below:

x	y
0	5
2	4.151601

4	4.461229
6	5.456047
8	6.852243
10	8.471474
12	10.17813
14	11.80277
16	12.97942
18	12.69896
20	8



27.5 Centered finite differences can be substituted for the second and first derivatives to give,

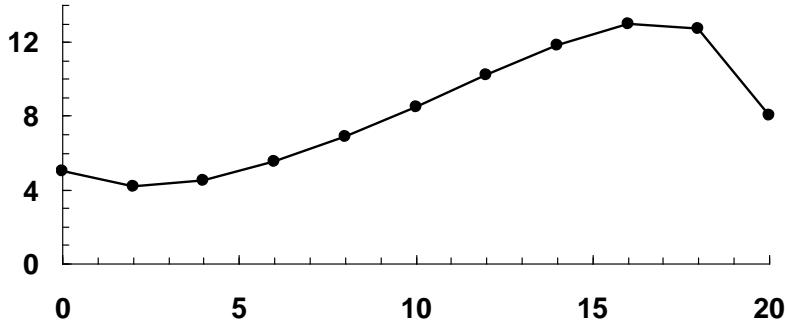
$$7 \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} - 2 \frac{y_{i+1} - y_{i-1}}{2\Delta x} - y_i + x_i = 0$$

or substituting $\Delta x = 2$ and collecting terms yields

$$-2.25y_{i-1} + 4.5y_i - 1.25y_{i+1} = x_i$$

This equation can be written for each node and solved with methods such as the Tridiagonal solver, the Gauss-Seidel method or LU Decomposition. The following solution was computed using Excel's Minverse and Mmult functions:

x	y
0	5
2	4.199592
4	4.518531
6	5.507445
8	6.893447
10	8.503007
12	10.20262
14	11.82402
16	13.00176
18	12.7231
20	8



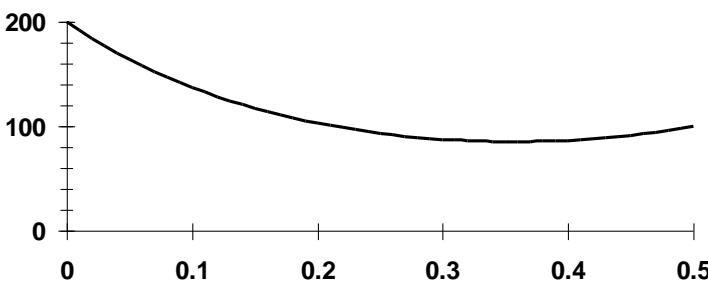
27.6 The second-order ODE can be expressed as the following pair of first-order ODEs,

$$\frac{dT}{dx} = z$$

$$\frac{dz}{dx} = 1 \times 10^{-7} (T + 273)^4 - 4(150 - T)$$

The solution was then generated on the Excel spreadsheet using the Heun method (without iteration) with a step-size of 0.01. The Excel Solver was used to adjust the initial condition of z until the value of $T(0.5) = 100$. Part of the resulting spreadsheet is shown below along with a graph of the final solution.

x	T	z	k_{11}	k_{12}	T_{end}	z_{end}	k_{21}	k_{22}	ϕ_1	ϕ_2
0	200.000	-839.391	-839.391	5205.467	191.606	-787.336	-787.336	4825.927	-813.364	5015.697
0.01	191.866	-789.234	-789.234	4837.418	183.974	-740.860	-740.860	4496.695	-765.047	4667.057
0.02	184.216	-742.564	-742.564	4506.902	176.790	-697.495	-697.495	4200.146	-720.029	4353.524
0.03	177.016	-699.028	-699.028	4209.256	170.025	-656.936	-656.936	3932.349	-677.982	4070.802
0.04	170.236	-658.320	-658.320	3940.516	163.653	-618.915	-618.915	3689.943	-638.618	3815.229
0.05	163.850	-620.168	-620.168	3697.296	157.648	-583.195	-583.195	3470.045	-601.682	3583.671



27.7 The second-order ODE can be linearized as in

$$\frac{d^2T}{dx^2} - 1 \times 10^{-7} (T_b + 273)^4 - 4 \times 10^{-7} (T_b + 273)^3 (T - T_b) + 4(150 - T) = 0$$

Substituting $T_b = 150$ and collecting terms gives

$$\frac{d^2T}{dx^2} - 34.27479T + 1939.659 = 0$$

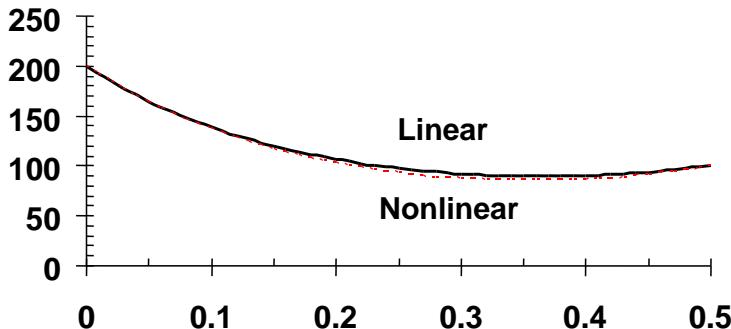
Substituting a centered-difference approximation of the second derivative gives

$$-T_{i-1} + (2 + 34.27479 \Delta x^2)T_i - T_{i+1} = 1939.659 \Delta x^2$$

We used the Gauss-Seidel method to solve these equations. The results for a few selected points are:

x	0	0.1	0.2	0.3	0.4	0.5
T	200	138.8337	106.6616	92.14149	90.15448	100

A graph of the entire solution along with the nonlinear result from Prob. 27.7 is shown below:



27.8 For three springs

$$\begin{aligned} \left(\frac{2k}{m_1} - \omega^2 \right) A_1 - \frac{k}{m_1} A_2 &= 0 \\ -\frac{k}{m_2} A_1 + \left(\frac{2k}{m_2} - \omega^2 \right) A_2 - \frac{k}{m_2} A_3 &= 0 \\ -\frac{k}{m_3} A_2 + \left(\frac{2k}{m_3} - \omega^2 \right) A_3 &= 0 \end{aligned}$$

Substituting $m = 40$ kg and $k = 240$ gives

$$\begin{aligned} (12 - \omega^2) A_1 - 6A_2 &= 0 \\ -6A_1 + (12 - \omega^2) A_2 - 6A_3 &= 0 \\ -6A_2 + (12 - \omega^2) A_3 &= 0 \end{aligned}$$

The determinant is

$$-\omega^6 + 36\omega^4 - 360\omega^2 + 864 = 0$$

which can be solved for $\omega^2 = 20.4853, 12$, and 3.5147 s^{-2} . Therefore the frequencies are $\omega = 4.526, 3.464$, and 1.875 s^{-1} . Substituting these values into the original equations yields for $\omega^2 = 20.4853$,

$$A_1 = -0.707A_2 = A_3$$

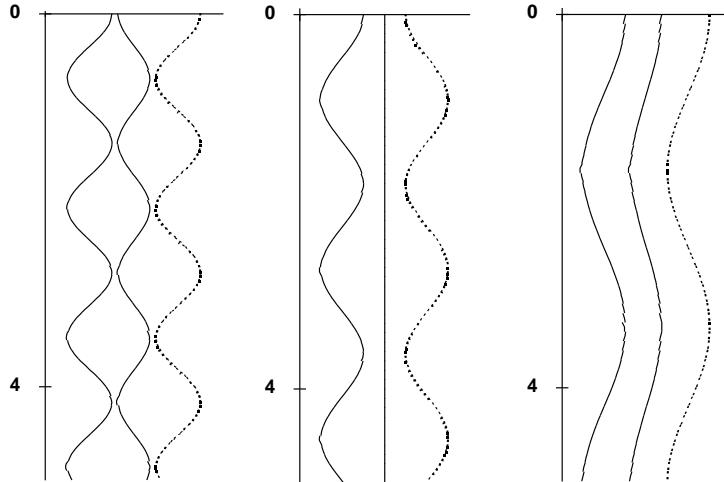
for $\omega^2 = 12$

$$A_1 = -A_3, \text{ and } A_2 = 0$$

for $\omega^2 = 3.5147$

$$A_1 = 0.707A_2 = A_3$$

Plots:



- 27.9** For 5 interior points ($h = 3/6 = 0.5$), the result is Eq. (27.19) with $2 - 0.25p^2$ on the diagonal. Dividing by 0.25 gives,

$$\begin{bmatrix} 8-p^2 & -4 & & & \\ -4 & 8-p^2 & -4 & & \\ & -4 & 8-p^2 & -4 & \\ & & -4 & 8-p^2 & -4 \\ & & & -4 & 8-p^2 \end{bmatrix} = 0$$

The determinant can be expanded (e.g., with Fadeev-Leverrier or the MATLAB **poly** function) to give

$$0 = -p^{10} + 40p^8 - 576p^6 + 3,584p^4 - 8960p^2 + 6,144$$

The roots of this polynomial can be determined as (e.g., with Bairstow's methods or the MATLAB **roots** function) $p^2 = 1.072, 4, 8, 12, 14.94$. The square root of these roots yields $p = 1.035, 2, 2.828, 3.464$, and 3.864 .

27.10 Minors:

$$(2-\lambda) \begin{vmatrix} 4-\lambda & 5 \\ 5 & 7-\lambda \end{vmatrix} - 8 \begin{vmatrix} 8 & 5 \\ 10 & 7-\lambda \end{vmatrix} + 10 \begin{vmatrix} 8 & 4-\lambda \\ 10 & 5 \end{vmatrix} = -\lambda^3 + 13\lambda^2 + 139\lambda - 42$$

27.11 Although the following computation can be implemented on a pocket calculator, a spreadsheet or with a program, we've used MATLAB.

```
>> a=[2 8 10;8 4 5;10 5 7]
a =
    2      8      10
    8      4      5
   10      5      7
>> x=[1 1 1] '
x =
    1
    1
    1
```

First iteration:

```
>> x=a*x
x =
    20
    17
    22
>> e=max(x)
e =
    22
>> x=x/e
x =
    0.9091
    0.7727
    1.0000
```

Second iteration:

```
>> x=a*x
x =
    18.0000
    15.3636
    19.9545
>> e=max(x)
e =
    19.9545
>> x=x/e
x =
    0.9021
    0.7699
    1.0000
```

Third iteration:

```
>> x=a*x
x =
    17.9636
    15.2961
    19.8702
>> e=max(x)
e =
    19.8702
>> x=x/e
x =
    0.9040
    0.7698
    1.0000
```

Fourth iteration:

```
>> x=a*x
x =
    17.9665
    15.3116
    19.8895
>> e=max(x)
e =
    19.8895
>> x=x/e
x =
    0.9033
    0.7698
    1.0000
```

Thus, after four iterations, the result is converging on a highest eigenvalue of 19.8842 with a corresponding eigenvector of [0.9035 0.7698 1].

27.12 As in Example 27.10, the computation can be laid out as

2	8	10					
8	4	5					
10	5	7					
First iteration:						eigenvalue	eigenvector
-0.07143	0.142857	0	1		0.071429		0.1363636
0.142857	2.047619	-1.66667	1	=	0.52381	0.52381	1
0	-1.66667	1.333333	1		-0.33333		-0.6363636
Second iteration:							
-0.07143	0.142857	0	0.136364		0.133117		0.0425606
0.142857	2.047619	-1.66667	1	=	3.127706	3.127706	1
0	-1.66667	1.333333	-0.63636		-2.51515		-0.8041522
Third iteration:							
-0.07143	0.142857	0	0.042561		0.139817		0.0411959
0.142857	2.047619	-1.66667	1	=	3.393953	3.393953	1
0	-1.66667	1.333333	-0.80415		-2.73887		-0.8069852
Fourth iteration:							
-0.07143	0.142857	0	0.041196		0.139915		0.0411698
0.142857	2.047619	-1.66667	1	=	3.398479	3.398479	1

0	-1.66667	1.333333	-0.80699		-2.74265		-0.8070218
Fifth iteration:							
-0.07143	0.142857	0	0.04117		0.139916		0.0411696
0.142857	2.047619	-1.66667	1	=	3.398537	3.398537	1
0	-1.66667	1.333333	-0.80702		-2.7427		-0.8070225

Thus, after four iterations, the estimate of the lowest eigenvalue is $1/(3.398537) = 0.294244$ with an eigenvector of $[0.0411696 \ 1 - 0.8070225]$.

27.13 Here is VBA Code to implement the shooting method:

```

Option Explicit

Sub Shoot()
Dim n As Integer, m As Integer, i As Integer, j As Integer
Dim x0 As Double, xf As Double
Dim x As Double, y(2) As Double, h As Double, dx As Double, xend As Double
Dim xp(200) As Double, yp(2, 200) As Double, xout As Double
Dim z01 As Double, z02 As Double, T01 As Double, T02 As Double
Dim T0 As Double, Tf As Double
Dim Tf1 As Double, Tf2 As Double
'set parameters
n = 2
x0 = 0
T0 = 40
xf = 10
Tf = 200
dx = 2
xend = xf
xout = 2
'first shot
x = x0
y(1) = T0
y(2) = 10
Call RKsystems(x, y, n, dx, xf, xout, xp, yp, m)
z01 = yp(2, 0)
Tf1 = yp(1, m)
'second shot
x = x0
y(1) = T0
y(2) = 20
Call RKsystems(x, y, n, dx, xf, xout, xp, yp, m)
z02 = yp(2, 0)
Tf2 = yp(1, m)
'last shot
x = x0
y(1) = T0
'linear interpolation
y(2) = z01 + (z02 - z01) / (Tf2 - Tf1) * (Tf - Tf1)
Call RKsystems(x, y, n, dx, xf, xout, xp, yp, m)
'output results
Range("A4:C1004").ClearContents
Range("A4").Select
For j = 0 To m
    ActiveCell.Value = xp(j)
    For i = 1 To n
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = yp(i, j)
    Next i
Next j
End Sub

```

```

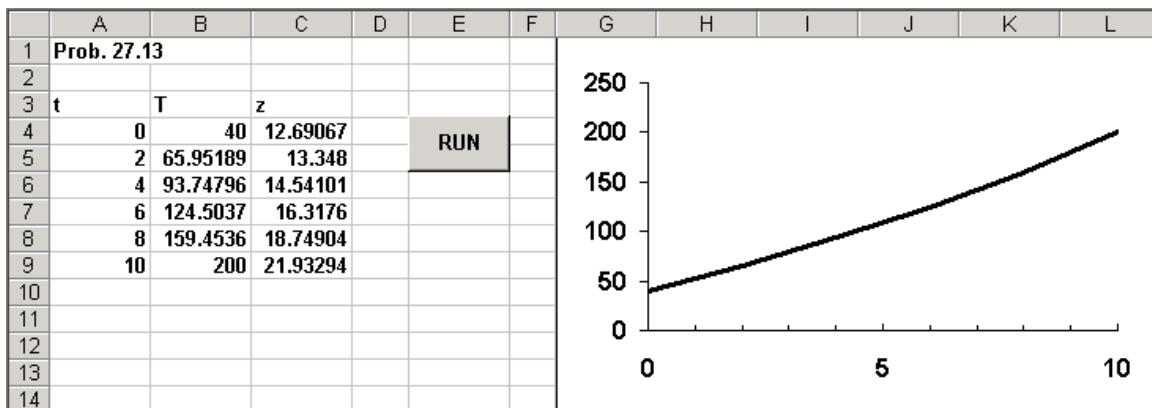
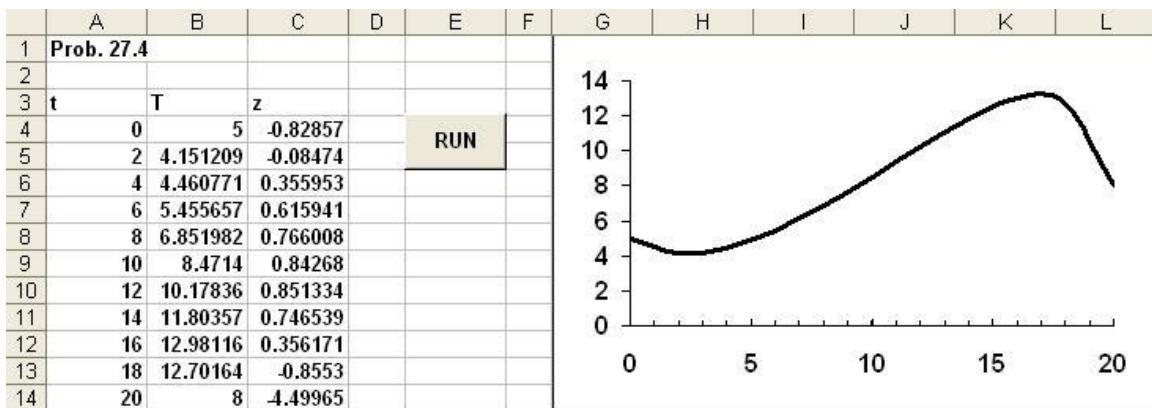
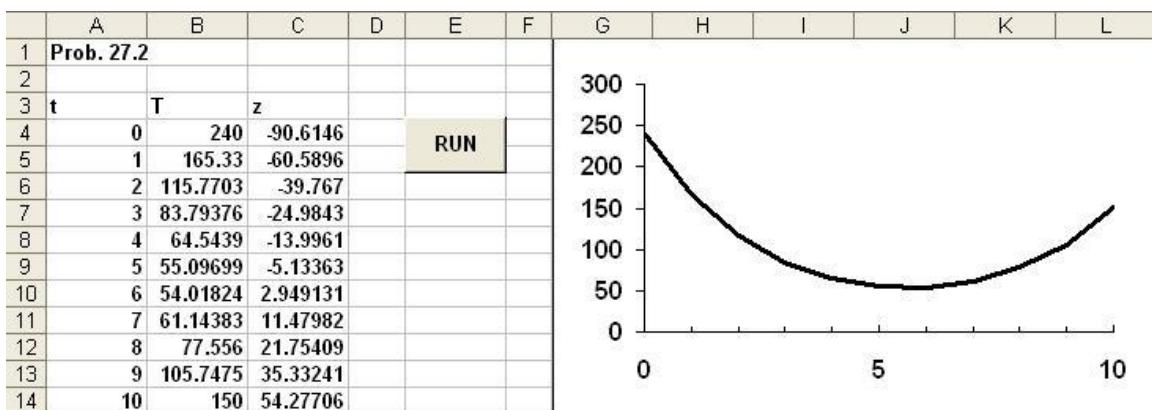
Next i
    ActiveCell.Offset(1, -n).Select
Next j
Range("A4").Select
End Sub

Sub RKsystems(x, y, n, dx, xf, xout, xp, yp, m)
Dim i As Integer
Dim xend As Double, h As Double
m = 0
For i = 1 To n
    yp(i, m) = y(i)
Next i
Do
    xend = x + xout
    If xend > xf Then xend = xf
    h = dx
    Do
        If xend - x < h Then h = xend - x
        Call RK4(x, y, n, h)
        If x >= xend Then Exit Do
    Loop
    m = m + 1
    xp(m) = x
    For i = 1 To n
        yp(i, m) = y(i)
    Next i
    If x >= xf Then Exit Do
Loop
End Sub

Sub RK4(x, y, n, h)
Dim i
Dim ynew, dydx(10), ym(10), ye(10)
Dim k1(10), k2(10), k3(10), k4(10)
Dim slope(10)
Call Derivs(x, y, k1)
For i = 1 To n
    ym(i) = y(i) + k1(i) * h / 2
Next i
Call Derivs(x + h / 2, ym, k2)
For i = 1 To n
    ym(i) = y(i) + k2(i) * h / 2
Next i
Call Derivs(x + h / 2, ym, k3)
For i = 1 To n
    ye(i) = y(i) + k3(i) * h
Next i
Call Derivs(x + h, ye, k4)
For i = 1 To n
    slope(i) = (k1(i) + 2 * (k2(i) + k3(i)) + k4(i)) / 6
Next i
For i = 1 To n
    y(i) = y(i) + slope(i) * h
Next i
x = x + h
End Sub

Sub Derivs(x, y, dydx)
dydx(1) = y(2)
dydx(2) = 0.01 * (y(1) - 20)
End Sub

```

**27.14**

27.15 A general formulation that describes Example 27.3 as well as Probs. 27.3 and 27.5 is

$$a \frac{d^2y}{dx^2} + b \frac{dy}{dx} + cy + f(x) = 0$$

Finite difference approximations can be substituted for the derivatives:

$$a \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} + b \frac{y_{i+1} - y_{i-1}}{2\Delta x} + cy_i + f(x_i) = 0$$

Collecting terms

$$-(a - 0.5b\Delta x)y_{i-1} + (2a - c\Delta x^2)y_i - (a + 0.5b\Delta x)y_{i+1} = f(x_i)\Delta x^2$$

Dividing by Δx^2 ,

$$-(a/\Delta x^2 - 0.5b/\Delta x)y_{i-1} + (2a/\Delta x^2 - c)y_i - (a/\Delta x^2 + 0.5b/\Delta x)y_{i+1} = f(x_i)$$

For Example 27.3, $a = 1$, $b = 0$, $c = -h'$ and $f(x) = h'T_a$. The following VBA code implements Example 27.3.

```

Public hp As Double
Option Explicit
Sub FDBoundaryValue()
    Dim ns As Integer, i As Integer
    Dim a As Double, b As Double, c As Double
    Dim e(100) As Double, f(100) As Double, g(100) As Double, r(100) As Double,
    y(100) As Double
    Dim Lx As Double, xx As Double, x(100) As Double, dx As Double
    Lx = 10
    dx = 2
    ns = Lx / dx
    xx = 0
    For i = 0 To ns
        x(i) = xx
        xx = xx + dx
    Next i
    hp = 0.01
    a = 1
    b = 0
    c = -hp
    y(0) = 40
    y(ns) = 200
    f(1) = 2 * a / dx ^ 2 - c
    g(1) = -(a / dx ^ 2 + b / (2 * dx))
    r(1) = ff(x(1)) + (a / dx ^ 2 - b / (2 * dx)) * y(0)
    For i = 2 To ns - 2
        e(i) = -(a / dx ^ 2 - b / (2 * dx))
        f(i) = 2 * a / dx ^ 2 - c
        g(i) = -(a / dx ^ 2 + b / (2 * dx))
        r(i) = ff(x(i))
    Next i
    e(ns - 1) = -(a / dx ^ 2 - b / (2 * dx))
    f(ns - 1) = 2 * a / dx ^ 2 - c
    r(ns - 1) = ff(x(ns - 1)) + (a / dx ^ 2 + b / (2 * dx)) * y(ns)
    Sheets("Sheet2").Select
    Range("a5:d105").ClearContents
    Range("a5").Select
    For i = 1 To ns - 1
        ActiveCell.Value = e(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = f(i)
        ActiveCell.Offset(0, 1).Select
    Next i

```

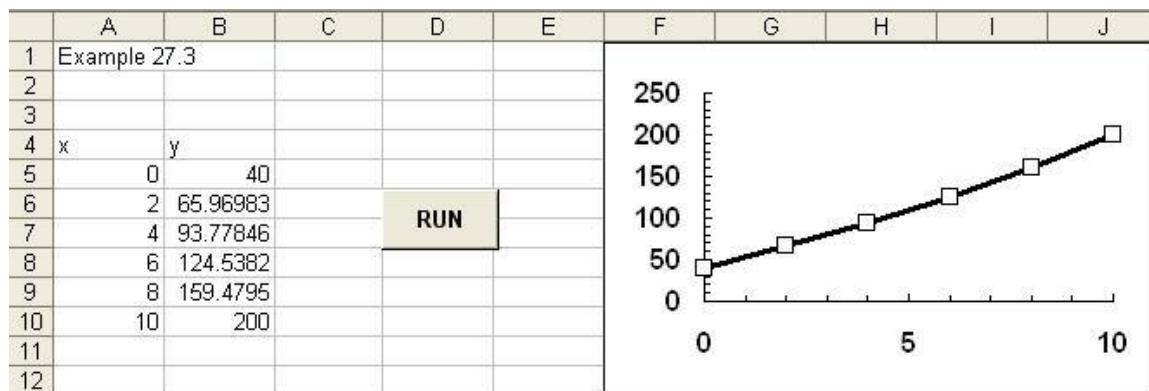
```

ActiveCell.Value = g(i)
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = r(i)
ActiveCell.Offset(1, -3).Select
Next i
Range("a5").Select
Call Tridiag(e, f, g, r, ns - 1, y)
Sheets("Sheet1").Select
Range("a5:b105").ClearContents
Range("a5").Select
For i = 0 To ns
    ActiveCell.Value = x(i)
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = y(i)
    ActiveCell.Offset(1, -1).Select
Next i
Range("a5").Select
End Sub

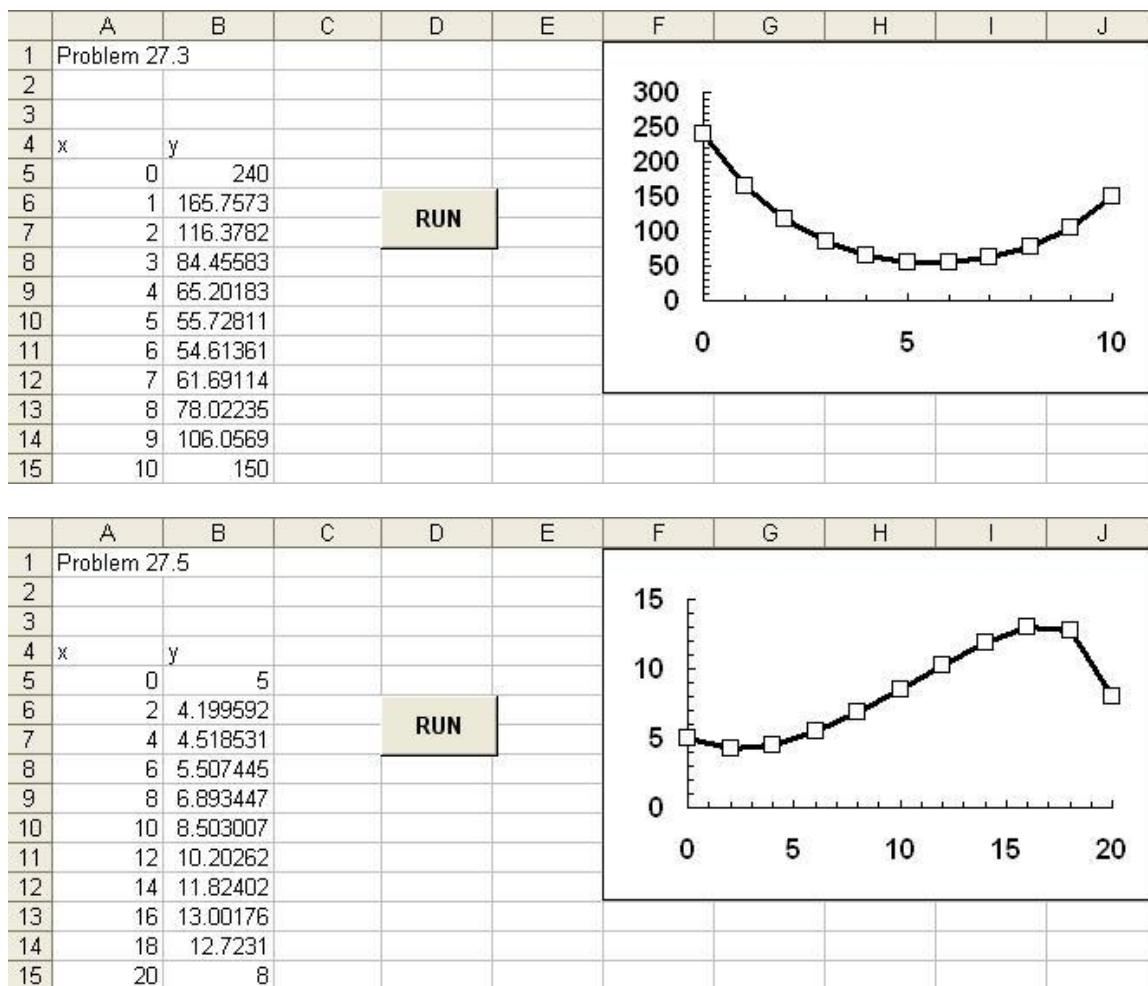
Sub Tridiag(e, f, g, r, n, x)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

Function ff(x)
ff = hp * 20
End Function

```



27.16



27.17 The following two codes can be used to solve this problem. The first is written in VBA/Excel. The second is an M-file implemented in MATLAB.

VBA/Excel:

```

Option Explicit
Sub Power()
    Dim n As Integer, i As Integer, iter As Integer
    Dim aa As Double, bb As Double
    Dim a(10, 10) As Double, c(10) As Double
    Dim lam As Double, lamold As Double, v(10) As Double
    Dim es As Double, ea As Double
    es = 0.001
    n = 3
    aa = 2 / 0.5625
    bb = -1 / 0.5625
    a(1, 1) = aa
    a(1, 2) = bb
    For i = 2 To n - 1
        a(i, i - 1) = bb
        a(i, i) = aa
        a(i, i + 1) = bb
    Next i

```

```

a(i, i - 1) = bb
a(i, i) = aa
lam = 1
For i = 1 To n
    v(i) = lam
Next i
Sheets("sheet1").Select
Range("a3:b1000").ClearContents
Range("a3").Select
Do
    iter = iter + 1
    Call Mmult(a, (v), v, n, n, 1)
    lam = Abs(v(1))
    For i = 2 To n
        If Abs(v(i)) > lam Then lam = Abs(v(i))
    Next i
    ActiveCell.Value = "iteration: "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = iter
    ActiveCell.Offset(1, -1).Select
    ActiveCell.Value = "eigenvalue: "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = lam
    ActiveCell.Offset(1, -1).Select
    For i = 1 To n
        v(i) = v(i) / lam
    Next i
    ActiveCell.Value = "eigenvector:"
    ActiveCell.Offset(0, 1).Select
    For i = 1 To n
        ActiveCell.Value = v(i)
        ActiveCell.Offset(1, 0).Select
    Next i
    ActiveCell.Offset(1, -1).Select
    ea = Abs((lam - lamold) / lam) * 100
    lamold = lam
    If ea <= es Then Exit Do
Loop
End Sub

Sub Mmult(a, b, c, m, n, l)
Dim i As Integer, j As Integer, k As Integer
Dim sum As Double
For i = 1 To n
    sum = 0
    For k = 1 To m
        sum = sum + a(i, k) * b(k)
    Next k
    c(i) = sum
Next i
End Sub

```

	A	B	C	D	E
1	Example 27.7				
2					
3	iteration:	1			
4	eigenvalue:	1.777778		RUN	
5	eigenvector:	1			
6		0			
7		1			
8					
9	iteration:	2			
10	eigenvalue:	3.555556			
11	eigenvector:	1			
12		-1			
13		1			
14					
15	iteration:	3			
16	eigenvalue:	7.111111			
17	eigenvector:	0.75			
18		-1			
19		0.75			

•
•
•

57	iteration:	10			
58	eigenvalue:	6.069717			
59	eigenvector:	0.707107			
60		-1			
61		0.707107			

MATLAB:

```

function [e, v] = powmax(A)
% [e, v] = powmax(A):
%   uses the power method to find the highest eigenvalue and
%   the corresponding eigenvector
% input:
%   A = matrix to be analyzed
% output:
%   e = eigenvalue
%   v = eigenvector

es = 0.0001;
maxit = 100;
n = size(A);
for i=1:n
    v(i)=1;
end
v = v';
e = 1;
iter = 0;

```

```

while (1)
    eold = e;
    x = A*v;
    [e,i] = max(abs(x));
    e = sign(x(i))*e;
    v = x/e;
    iter = iter + 1;
    ea = abs((e - eold)/e) * 100;
    if ea <= es | iter >= maxit, break, end
end

```

Application to solve Example 27.7,

```

>> A=[3.556 -1.778 0;-1.778 3.556 -1.778;0 -1.778 3.556];
>> [e,v]=powmax(A)
e =
    6.0705
v =
   -0.7071
    1.0000
   -0.7071

```

27.18 The following two codes can be used to solve this problem. The first is written in VBA/Excel. The second is an M-file implemented in MATLAB.

VBA/Excel:

```

Option Explicit

Sub Power()
Dim n As Integer, i As Integer, iter As Integer, j As Integer
Dim aa As Double, bb As Double
Dim a(10, 10) As Double, c(10) As Double
Dim lam As Double, lamold As Double, v(10) As Double
Dim es As Double, ea As Double
Dim x(10) As Double, ai(10, 10) As Double
es = 0.0000011
n = 3
aa = 2 / 0.5625
bb = -1 / 0.5625
a(1, 1) = aa
a(1, 2) = bb
For i = 2 To n - 1
    a(i, i - 1) = bb
    a(i, i) = aa
    a(i, i + 1) = bb
Next i
a(i, i - 1) = bb
a(i, i) = aa
Call LUDminv(a, n, x)
lam = 1
For i = 1 To n
    v(i) = lam
Next i
Sheets("sheet1").Select
Range("a3:j1000").ClearContents
Range("a3").Select

```

```

ActiveCell.Value = "Matrix inverse:"
ActiveCell.Offset(1, 0).Select
For i = 1 To n
    For j = 1 To n
        ActiveCell.Value = a(i, j)
        ActiveCell.Offset(0, 1).Select
    Next j
    ActiveCell.Offset(1, -n).Select
Next i
ActiveCell.Offset(1, 0).Select
Do
    iter = iter + 1
    Call Mmult(a, (v), v, n, n, 1)
    lam = Abs(v(1))
    For i = 2 To n
        If Abs(v(i)) > lam Then lam = Abs(v(i))
    Next i
    ActiveCell.Value = "iteration: "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = iter
    ActiveCell.Offset(1, -1).Select
    ActiveCell.Value = "eigenvalue: "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = lam
    ActiveCell.Offset(1, -1).Select
    For i = 1 To n
        v(i) = v(i) / lam
    Next i
    ActiveCell.Value = "eigenvector: "
    ActiveCell.Offset(0, 1).Select
    For i = 1 To n
        ActiveCell.Value = v(i)
        ActiveCell.Offset(1, 0).Select
    Next i
    ActiveCell.Offset(1, -1).Select
    ea = Abs((lam - lamold) / lam) * 100
    lamold = lam
    If ea <= es Then Exit Do
Loop

End Sub

Sub Mmult(a, b, c, m, n, 1)
Dim i As Integer, j As Integer, k As Integer
Dim sum As Double
For i = 1 To n
    sum = 0
    For k = 1 To m
        sum = sum + a(i, k) * b(k)
    Next k
    c(i) = sum
Next i
End Sub

Sub LUDminv(a, n, x)
Dim i As Integer, j As Integer, er As Integer
Dim o(3) As Double, s(3) As Double, b(3) As Double
Dim ai(10, 10) As Double, tol As Double
tol = 0.00001
Call Decompose(a, n, tol, o, s, er)
If er = 0 Then
    For i = 1 To n

```

```

For j = 1 To n
    If i = j Then
        b(j) = 1
    Else
        b(j) = 0
    End If
Next j
Call Substitute(a, o, n, b, x)
For j = 1 To n
    ai(j, i) = x(j)
Next j
Next i
End If
For i = 1 To n
    For j = 1 To n
        a(i, j) = ai(i, j)
    Next j
Next i
End Sub

Sub Decompose(a, n, tol, o, s, er)
Dim i As Integer, j As Integer, k As Integer
Dim factor As Double
For i = 1 To n
    o(i) = i
    s(i) = Abs(a(i, 1))
    For j = 2 To n
        If Abs(a(i, j)) > s(i) Then s(i) = Abs(a(i, j))
    Next j
Next i
For k = 1 To n - 1
    Call Pivot(a, o, s, n, k)
    If Abs(a(o(k), k) / s(o(k))) < tol Then
        er = -1
        Exit For
    End If
    For i = k + 1 To n
        factor = a(o(i), k) / a(o(k), k)
        a(o(i), k) = factor
        For j = k + 1 To n
            a(o(i), j) = a(o(i), j) - factor * a(o(k), j)
        Next j
    Next i
Next k
If (Abs(a(o(k), k) / s(o(k))) < tol) Then er = -1
End Sub

Sub Pivot(a, o, s, n, k)
Dim ii As Integer, p As Integer
Dim big As Double, dummy As Double
p = k
big = Abs(a(o(k), k) / s(o(k)))
For ii = k + 1 To n
    dummy = Abs(a(o(ii), k) / s(o(ii)))
    If dummy > big Then
        big = dummy
        p = ii
    End If
Next ii
dummy = o(p)
o(p) = o(k)
o(k) = dummy

```

```

End Sub

Sub Substitute(a, o, n, b, x)
Dim k As Integer, i As Integer, j As Integer
Dim sum As Double, factor As Double
For k = 1 To n - 1
    For i = k + 1 To n
        factor = a(o(i), k)
        b(o(i)) = b(o(i)) - factor * b(o(k))
    Next i
Next k
x(n) = b(o(n)) / a(o(n), n)
For i = n - 1 To 1 Step -1
    sum = 0
    For j = i + 1 To n
        sum = sum + a(o(i), j) * x(j)
    Next j
    x(i) = (b(o(i)) - sum) / a(o(i), i)
Next i
End Sub

```

	A	B	C	D	E	F
1	Example 27.8					
2						
3	Matrix inverse:				RUN	
4	0.421875	0.28125	0.140625			
5	0.28125	0.5625	0.28125			
6	0.140624985	0.28125	0.421875			
7						
8	iteration:	1.0000				
9	eigenvalue:	1.125				
10	eigenvector:	0.75				
11		1				
12		0.75				
13						
14	iteration:	2				
15	eigenvalue:	0.984375				
16	eigenvector:	0.714286				
17		1				
18		0.714286				

•

•

•

50	iteration:	8				
51	eigenvalue:	0.960248				
52	eigenvector:	0.707107				
53		1				
54		0.707107				

MATLAB:

```

function [e, v] = powmin(A)
% [e, v] = powmin(A):

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

% uses the power method to find the lowest eigenvalue and
% the corresponding eigenvector
% input:
% A = matrix to be analyzed
% output:
% e = eigenvalue
% v = eigenvector

es = 0.0001;
maxit = 100;
n = size(A);
for i=1:n
    v(i)=1;
end
v = v';
e = 1;
Ai = inv(A);
iter = 0;
while (1)
    eold = e;
    x = Ai*v;
    [e,i] = max(abs(x));
    e = sign(x(i))*e;
    v = x/e;
    iter = iter + 1;
    ea = abs((e - eold)/e) * 100;
    if ea <= es | iter >= maxit, break, end
end
e = 1./e;

```

Application to solve Example 27.8,

```

>> A=[3.556 -1.778 0;-1.778 3.556 -1.778;0 -1.778 3.556];
>> [e,v]=powmin(A)
e =
    1.0415
v =
    0.7071
    1.0000
    0.7071

```

27.19 This problem can be solved by recognizing that the solution corresponds to driving the differential equation to zero. To do this, a finite difference approximation can be substituted for the second derivative to give

$$R = \frac{T_{i-1} - 2T_i + T_{i+1}}{(\Delta x)^2} - 1 \times 10^{-7} (T_i + 273)^4 + 4(150 - T_i)$$

where R = the residual, which is equal to zero when the equation is satisfied. Next, a spreadsheet can be set up as below. Guesses for T can be entered in cells B11:B14. Then, the residual equation can be written in cells C11:C14 and referenced to the temperatures in column B. The square of the R 's can then be entered in column D and summed (D17).

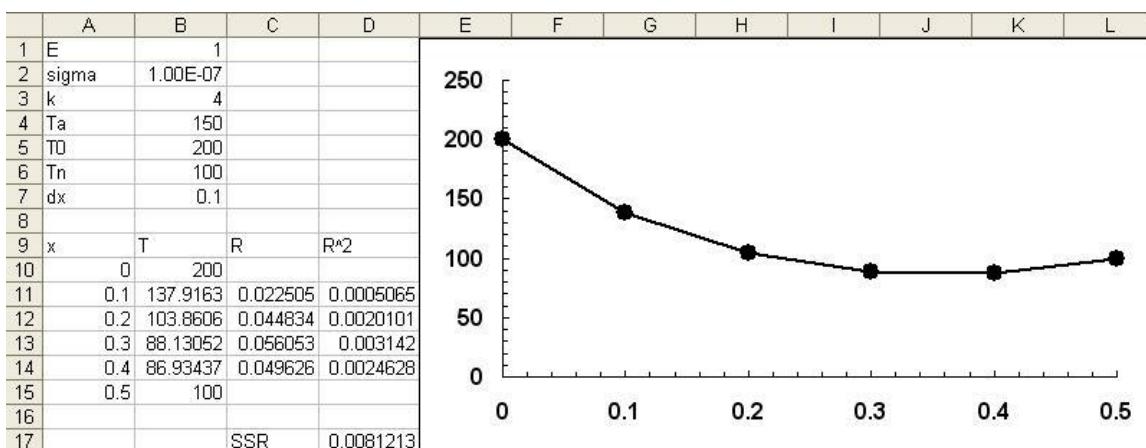
	A	B	C	D
1	E	1		
2	sigma	1.00E-07		
3	k	4		
4	Ta	150		
5	T0	200		
6	Tn	100		
7	dx	0.1		
8				
9	x	T	R	R^2
10		0	200	
11	0.1	0	20044.54	401783697
12	0.2	0	44.54282	1984.0624
13	0.3	0	44.54282	1984.0624
14	0.4	0	10044.54	100892840
15	0.5	100		
16		SSR	502680505	
17				= sum(D11:D14)

$=(B10-2*B11+B12)/\$B\$7^2-\$B\$2*(B11+273)^4+\$B\$3*(\$B\$4-B11)$

Solver can then be invoked to drive cell D17 to zero by varying B11:B14.



The result is as shown in the spreadsheet along with a plot.



27.20 First, an M-file containing the system of ODEs can be created and saved (in this case as `predprey.m`),

```
function dy = predprey(t,y)
dy=[0.35*y(1)-1.6*y(1)*y(2);-0.15*y(2)+0.04*y(1)*y(2)];
```

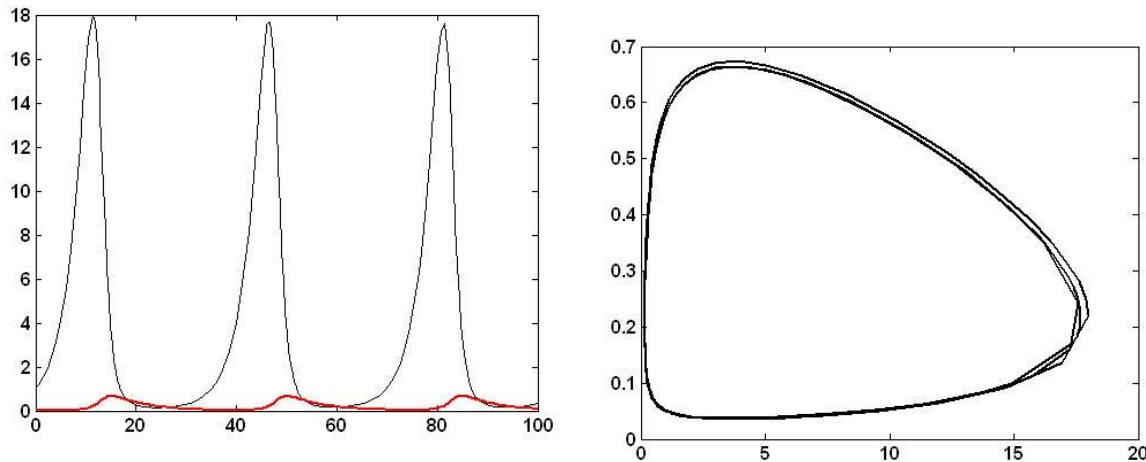
Then, the following MATLAB session is used to generate the solution:

```
>> [t,y]=ode45(@predprey,[0 100],[1;.05]);
```

A plot of the solution along with the state-space plot are generated with

```
>> plot(t,y)
>> plot(y(:,1),y(:,2))
```

These plots are displayed below



27.21 (a) First, the 2nd-order ODE can be reexpressed as the following system of 1st-order ODE's

$$\begin{aligned}\frac{dx}{dt} &= z \\ \frac{dz}{dt} &= -8z - 1200x\end{aligned}$$

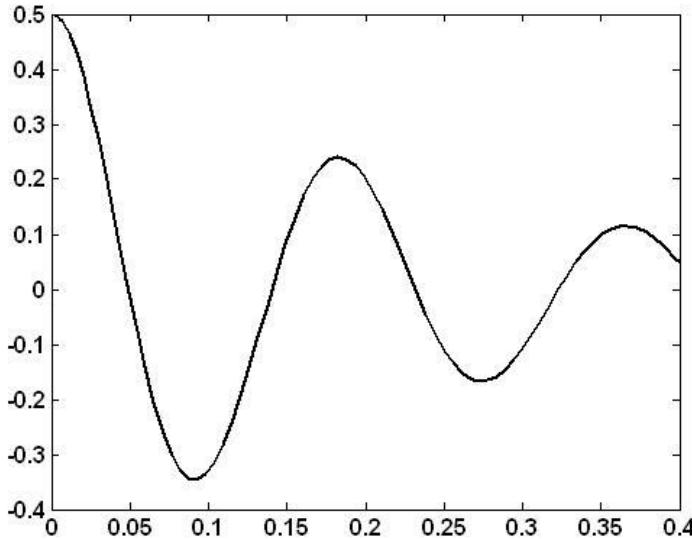
Next, we create an M-file to hold the ODEs:

```
function dx=spring(t,y)
dx=[y(2);-8*y(2)-1200*y(1)]
```

Then we enter the following commands into MATLAB

```
[t,y]=ode45('spring',[0 .4],[0.5;0]);
plot(t,y(:,1));
```

The following plot results:



(b) The eigenvalues and eigenvectors can be determined with the following commands:

```
>> a=[0 -1;8 1200];
>> format short e
>> [v,d]=eig(a)
v =
-9.9998e-001 8.3334e-004
 6.6666e-003 -1.0000e+000
d =
 6.6667e-003          0
      0   1.2000e+003
```

27.22 This problem is solved in an identical fashion to that employed in Example 27.11. For part (a), the program can be written as

```
Program PredPrey
USE msimsl
INTEGER :: mxparm, n
PARAMETER (mxparm=50, n=2)
INTEGER :: ido, nout
REAL:: param(mxparm), t, tend, tol, y(n)
EXTERNAL fcn
CALL UMACH (2, nout)
t = 0.0
y(1) = 2.0
y(2) = 1.0
tol = 0.0005
CALL SSET(mxparm, 0.0, param, 1)
param(10) = 1.0
PRINT '(7X, "Time", 9X, "Y1", 11X, "Y2")'
ido = 1
tend = 0
WRITE(nout,'(3F12.3)') t, y
DO
```

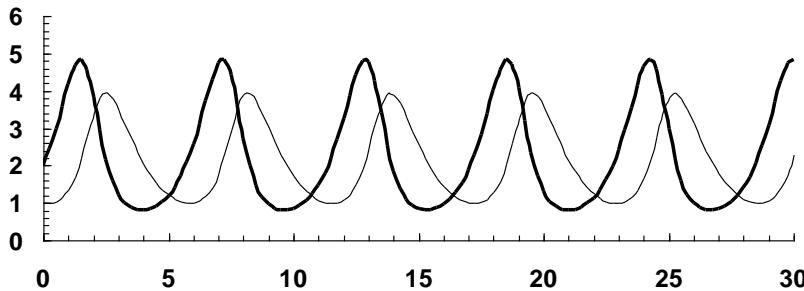
```

tend = tend + 0.25
CALL IVPRK (ido, n, fcn, t, tend, tol, param, y)
IF (tend .GT. 30) EXIT
WRITE(nout,'(3F12.3)') t, y
IF (tend .EQ. 30) ido = 3
END DO
END PROGRAM

SUBROUTINE fcn(n, t, y, yprime)
IMPLICIT NONE
INTEGER :: n
REAL :: t, y(n), yprime(n)
yprime(1) = 1.5*y(1) - 0.7*y(1)*y(2)
yprime(2) = -0.9*y(2) + 0.4*y(1)*y(2)
END SUBROUTINE

```

The results are displayed in the following plot:



(b) The program can be written as

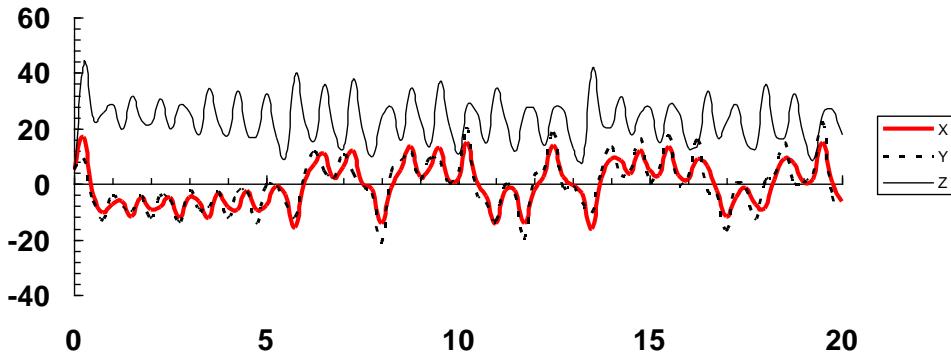
```

Program Lorenz
USE msimsl
INTEGER :: mxparm, n
PARAMETER (mxparm=50, n=3)
INTEGER :: ido, nout
REAL:: param(mxparm), t, tend, tol, y(n)
EXTERNAL fcn
CALL UMACH (3, nout)
t = 0.0
y(1) = 5.
y(2) = 5.
y(3) = 5.
tol = 0.0005
CALL SSET(mxparm, 0.0, param, 1)
param(10) = 1.0
PRINT '(7X, "Time", 9X, "X", 11X, "Y", 11X, "Z")'
ido = 1
tend = 0
WRITE(nout,'(4F12.3)') t, y
DO
    tend = tend + 0.25
    CALL IVPRK (ido, n, fcn, t, tend, tol, param, y)
    IF (tend .GT. 20.) EXIT
    WRITE(nout,'(4F12.3)') t, y
    IF (tend .EQ. 20.) ido = 3
END DO
END PROGRAM

```

```
SUBROUTINE fcn(n, t, y, yprime)
IMPLICIT NONE
INTEGER :: n
REAL :: t, y(n), yprime(n)
yprime(1) = -10.*y(1)+10.*y(2)
yprime(2) = 28.*y(1)-y(2)-y(1)*y(3)
yprime(3) = -2.666667*y(3)+y(1)*y(2)
END SUBROUTINE
```

The results are displayed in the following plot:



Note that students can be directed to Sec. 28.2 for additional information on these equations and their solution.

27.23 Boundary Value Problem

1. x -spacing

at $x = 0, i = 1$; and at $x = 2, i = n$

$$\Delta x = \frac{2 - 0}{n - 1}$$

2. Finite Difference Equation

$$\frac{d^2 u}{dx^2} + 6 \frac{du}{dx} - u = 2$$

Substitute finite difference approximations:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + 6 \frac{u_{i+1} - u_{i-1}}{2\Delta x} - u_i = 2$$

$$[1 - 3(\Delta x)]u_{i-1} + [-2 - \Delta x^2]u_i + [1 + 3(\Delta x)]u_{i+1} = 2\Delta x^2$$

Coefficients:

$$a_i = 1 - 3\Delta x \quad b_i = -2 - \Delta x^2 \quad c_i = 1 + 3\Delta x \quad d_i = 2\Delta x^2$$

3. End point equations

i = 2:

$$[1 - 3(\Delta x)]10 + [-2 - \Delta x^2]u_2 + [1 + 3(\Delta x)]u_3 = 2\Delta x^2$$

Coefficients:

$$a_2 = 0 \quad b_2 = -2 - \Delta x^2 \quad c_2 = 1 + 3\Delta x \quad d_2 = 2\Delta x^2 - 10(1 - 3(\Delta x))$$

i = n - 1:

$$[1 - 3(\Delta x)]u_{n-2} + [-2 - \Delta x^2]u_{n-1} + [1 + 3(\Delta x)]1 = 2\Delta x^2$$

Coefficients:

$$a_2 = 1 - 3\Delta x \quad b_2 = -2 - \Delta x^2 \quad c_2 = 0 \quad d_2 = 2\Delta x^2 - (1 - 3(\Delta x))$$

```
% Boundary Value Problem
% u''+6u'+u=2
% BC: u(x=0)=10 u(x=2)=1
% i=spatial index from 1 to n
% numbering for points is i=1 to i=21 for 20 dx spaces
% u(1)=10 and u(n)=1

n=41; xspan=2.0;

% Constants
dx=xspan/(n-1);
dx2=dx*dx;

% Sizing matrices
u=zeros(1,n); x=zeros(1,n);
a=zeros(1,n); b=zeros(1,n); c=zeros(1,n); d=zeros(1,n);
ba=zeros(1,n); ga=zeros(1,n);

% Coefficients and Boundary Conditions
x=0:dx:2;
u(1)=10; u(n)=1;
b(2)=-2-dx2;
c(2)=1+3*dx;
d(2)=2*dx2-(1-3*dx)*10;
for i=3:n-2
    a(i)=1-3*dx;
    b(i)=-2-dx2;
    c(i)=1+3*dx;
    d(i)=2*dx2;
end
a(n-1)=1-3*dx;
b(n-1)=-2-dx2;
d(n-1)=2*dx2-(1+3*dx);

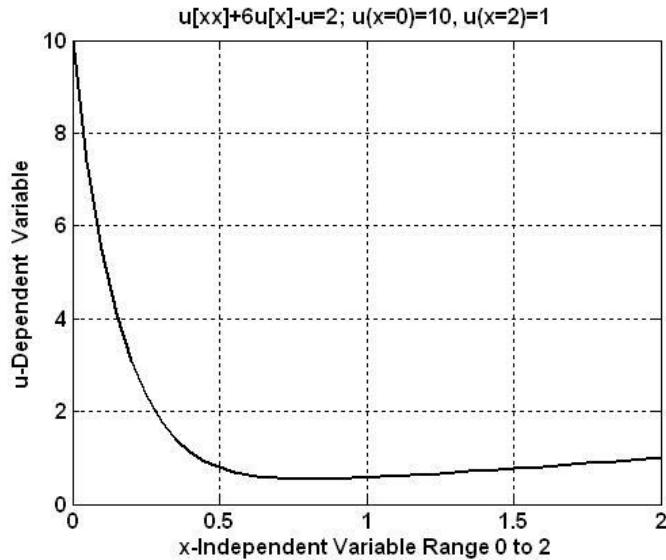
% Solution by Thomas Algorithm
```

```

ba(2)=b(2);
ga(2)=d(2)/b(2);
for i=3:n-1
    ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
    ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
end
% back substitution
u(n-1)=ga(n-1);
for i=n-2:-1:2
    u(i)=ga(i)-c(i)*u(i+1)/ba(i);
end

% Plot
plot(x,u)
title('u[xx]+6u[x]-u=2; u(x=0)=10, u(x=2)=1')
xlabel('x-Independent Variable Range 0 to 2'); ylabel('u-Dependent Variable')
grid

```



27.24

- Divide the radial coordinate into n finite points.

$$\Delta r = \frac{1}{n-1}$$

- The finite difference approximations for the general point i

$$\frac{d^2T}{dr^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta r^2}$$

$$\frac{dT}{dr} = \frac{T_{i+1} - T_{i-1}}{2\Delta r}$$

$$r = \Delta r(i - 1)$$

3. Substituting in the finite difference approximations for the derivatives

$$\frac{d^2T}{dr^2} + \frac{1}{r} \frac{dT}{dr} + S = 0$$

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta r^2} + \frac{1}{\Delta r(i-1)} \frac{T_{i+1} - T_{i-1}}{2\Delta r} + S = 0$$

4. Collecting like terms results in the general finite difference equation at point i

$$-\left[1 - \frac{1}{2(i-1)}\right]T_{i-1} + 2T_i - \left[1 + \frac{1}{2(i-1)}\right]T_{i+1} = \Delta r^2 S$$

5. End point equation at $i = 1$

$$\left. \frac{dT}{dr} \right|_{r=0} = 0$$

Substituting in the FD approximation gives

$$\frac{T_2 - T_0}{2\Delta r} = 0$$

where T_0 is a fictitious point. Thus, we see that $T_0 = T_2$ for zero slope at $r = 0$. Writing out the general equation at point $i = 1$ gives:

$$-\left[1 - \frac{1}{2(i-1)}\right]T_0 + 2T_1 - \left[1 + \frac{1}{2(i-1)}\right]T_2 = \Delta r^2 S$$

Substituting $T_0 = T_2$ and collecting terms gives

$$2T_1 - 2T_2 = \Delta r^2 S$$

6. End point equation at $i = n - 1$

$$T(r=1) = 1$$

$$-\left[1 - \frac{1}{2(n-1)}\right]T_{n-2} + 2T_{n-1} = \Delta r^2 S + \left[1 + \frac{1}{2(n-1)}\right]$$

7. Solve the resulting tridiagonal system of algebraic equations using the Thomas Algorithm.

8. Following program in MATLAB.

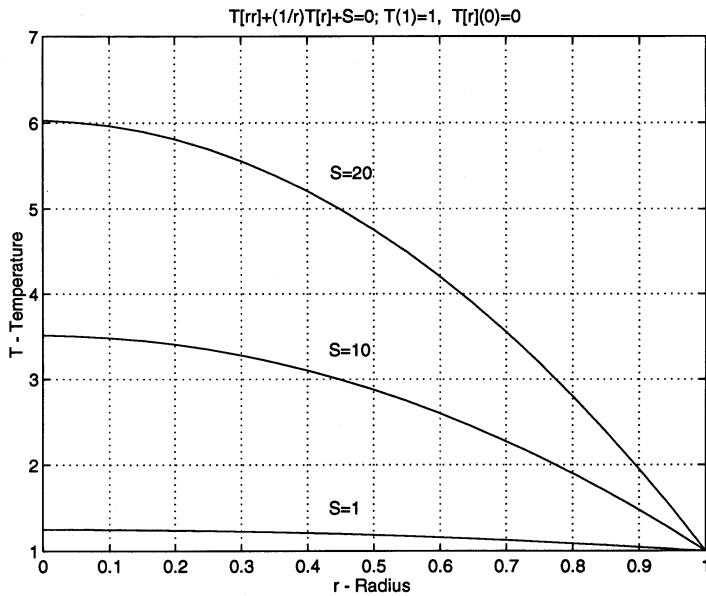
```
% Solution of the ODE Boundary Value Problem
% T[rr]+(1/r)T[r]+S=0
% BC: T(r=1)=1 T[r](r=0)=0
% i=spatial index from 1 to n
% numbering for points: i=1 to i=21 for 20 dr spaces
% i=1 (r=0), and i=n (r=1)
% T(n)=1 and T'(0)=0

% Constants
n=6;
dr=1/(n-1);
dr2=dr*dr;
S=1;

% Sizing Matrices
rad=0:dr:1;
T=zeros(1,n);
e=zeros(1,n); f=zeros(1,n); g=zeros(1,n); r=zeros(1,n);
ba=zeros(1,n); ga=zeros(1,n);
% Coefficients and Boundary Conditions
f(1)=2;
g(1)=-2;
r(1)=dr2*S;
for i=2:n-2
    e(i)=-1+1/(2*(i-1));
    f(i)=2;
    g(i)=-1-1/(2*(i-1));
    r(i)=dr2*S;
end
e(n-1)=-1+1/(2*(n-2));
f(n-1)=2;
r(n-1)=dr2*S+(1+1/(2*(n-2)));
T(n)=1;
% Solution by Thomas Algorithm
for i=2:n-1
    e(i)=e(i)/f(i-1);
    f(i)=f(i)-e(i)*g(i-1);
end
for i=2:n-1
    r(i)=r(i)-e(i)*r(i-1);
end
T(n-1)=r(n-1)/f(n-1);
for i=n-2:-1:1
    T(i)=(r(i)-g(i)*T(i+1))/f(i);
end

%Plot
plot(rad,T)
title('T[rr]+(1/r)T[r]+S=0; T(1)=1, T[r](0)=0')
xlabel('r - Radius'); ylabel('T - Temperature')
grid
```

Here is a plot of all the results for the 3 cases:



27.25 By summing forces on each mass and equating that to the mass times acceleration, the resulting differential equations can be written

$$\ddot{x}_1 + \left(\frac{k_1 + k_2}{m_1} \right) x_1 - \left(\frac{k_2}{m_1} \right) x_2 = 0$$

$$\ddot{x}_2 - \left(\frac{k_2}{m_2} \right) x_1 + \left(\frac{k_2 + k_3}{m_2} \right) x_2 - \left(\frac{k_3}{m_2} \right) x_3 = 0$$

$$\ddot{x}_3 - \left(\frac{k_3}{m_3} \right) x_2 + \left(\frac{k_3 + k_4}{m_3} \right) x_3 = 0$$

In matrix form

$$\begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{Bmatrix} + \begin{bmatrix} \frac{k_1 + k_2}{m_1} & -\frac{k_2}{m_1} & 0 \\ -\frac{k_2}{m_2} & \frac{k_2 + k_3}{m_2} & -\frac{k_3}{m_2} \\ 0 & -\frac{k_3}{m_3} & \frac{k_3 + k_4}{m_3} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

The k/m matrix becomes with: $k_1 = k_4 = 15 \text{ N/m}$, $k_2 = k_3 = 35 \text{ N/m}$, and $m_1 = m_2 = m_3 = 1.5 \text{ kg}$

$$\begin{bmatrix} \frac{k}{m} \\ m \end{bmatrix} = \begin{bmatrix} 33.33333 & -23.33333 & 0 \\ -23.33333 & 46.66667 & -23.33333 \\ 0 & -23.33333 & 33.33333 \end{bmatrix}$$

Solve for the eigenvalues/natural frequencies using MATLAB:

```
>> k1=15;k4=15;k2=35;k3=35;
>> m1=1.5;m2=1.5;m3=1.5;
>> a=[(k1+k2)/m1 -k2/m1 0;
-k2/m2 (k2+k3)/m2 -k3/m2;
0 -k3/m3 (k3+k4)/m3]

a =
    33.3333   -23.3333            0
   -23.3333    46.6667   -23.3333
        0   -23.3333    33.3333

>> w2=eig(a)

w2 =
    6.3350
   33.3333
   73.6650

>> w=sqrt(w2)

w =
    2.5169
    5.7735
    8.5828
```

27.26 Here is a MATLAB session that uses `eig` to determine the eigenvalues and the natural frequencies:

```
>> k=2;
>> kmw2=[2*k,-k,-k;-k,2*k,-k;-k,-k,2*k];
>> [v,d]=eig(kmw2)

v =
    0.5774    0.2673    0.7715
    0.5774   -0.8018   -0.1543
    0.5774    0.5345   -0.6172

d =
    -0.0000         0         0
        0    6.0000         0
        0         0    6.0000
```

Therefore, the eigenvalues are 0, 6, and 6. Setting these eigenvalues equal to $m\omega^2$, the three frequencies can be obtained.

$$m\omega_1^2 = 0 \Rightarrow \omega_1 = 0 \text{ (Hz) } 1^{\text{st}} \text{ mode of oscillation}$$

$$m\omega_2^2 = 6 \Rightarrow \omega_2 = \sqrt{6} \text{ (Hz) } 2^{\text{nd}} \text{ mode}$$

$$m\omega_3^2 = 6 \Rightarrow \omega_3 = \sqrt{6} \text{ (Hz) } 3^{\text{rd}} \text{ mode}$$

27.27 (a) The exact solution is

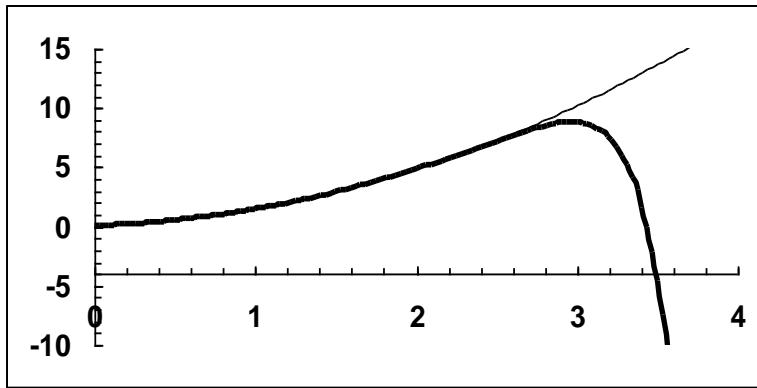
$$y = Ae^{5t} + t^2 + 0.4t + 0.08$$

If the initial condition at $t = 0$ is 0.8, $A = 0$,

$$y = t^2 + 0.4t + 0.08$$

Note that even though the choice of the initial condition removes the positive exponential terms, it still lurks in the background. Very tiny round off errors in the numerical solutions bring it to the fore. Hence all of the following solutions eventually diverge from the analytical solution.

(b) 4th order RK. The plot shows the numerical solution (bold line) along with the exact solution (fine line).



(c)

```
function yp=dy(t,y)
yp=5*(y-t^2);

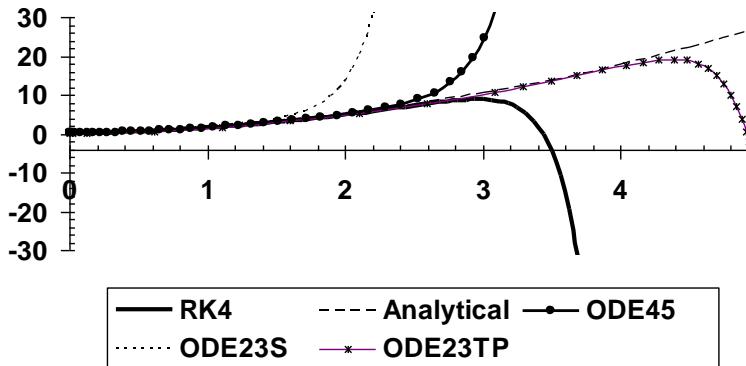
>> tspan=[0,5];
>> y0=0.08;
>> [t,y]=ode45('dy1',tspan,y0);
```

(d)

```
>> [t,y]=ode23S('dy1',tspan,y0);
```

(e)

```
>> [t,y]=ode23TB('dy1',tspan,y0);
```



27.28 First, the 2nd-order ODE can be reexpressed as the following system of 1st-order ODE's

$$\begin{aligned}\frac{dT}{dx} &= z \\ \frac{dz}{dx} &= -25\end{aligned}$$

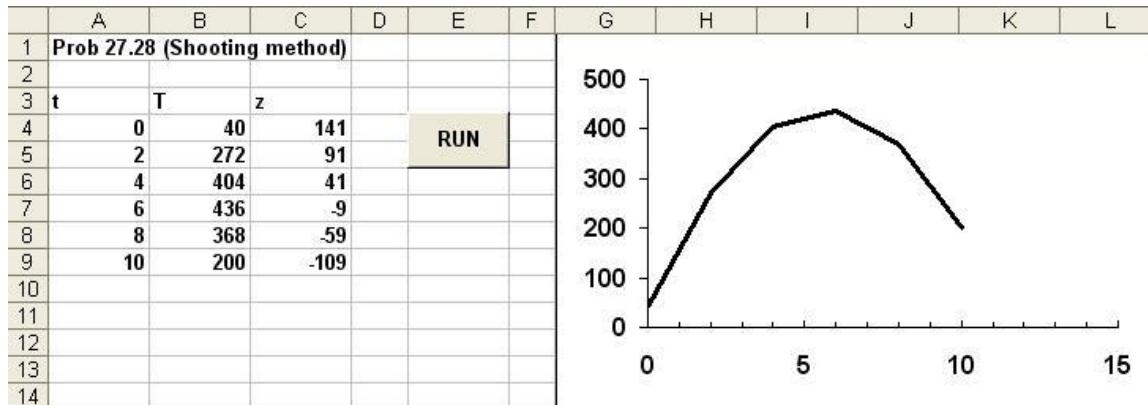
(a) Shooting method: These can be solved for two guesses for the initial condition of z . For our cases we used -1 and -0.5 . We solved the ODEs with the 4th-order RK method using a step size of 0.125. The results are

$z(0)$	-1	-0.5
$T(10)$	-1220	-1215

These values can then be used to derive the correct initial condition,

$$z(0) = -1 + \frac{-0.5 + 1}{-1215 - (-1220)} (200 - (-1220)) = 141$$

The resulting fit is displayed below:



(b) Finite difference: Centered finite differences can be substituted for the second and first derivatives to give,

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + 25 = 0$$

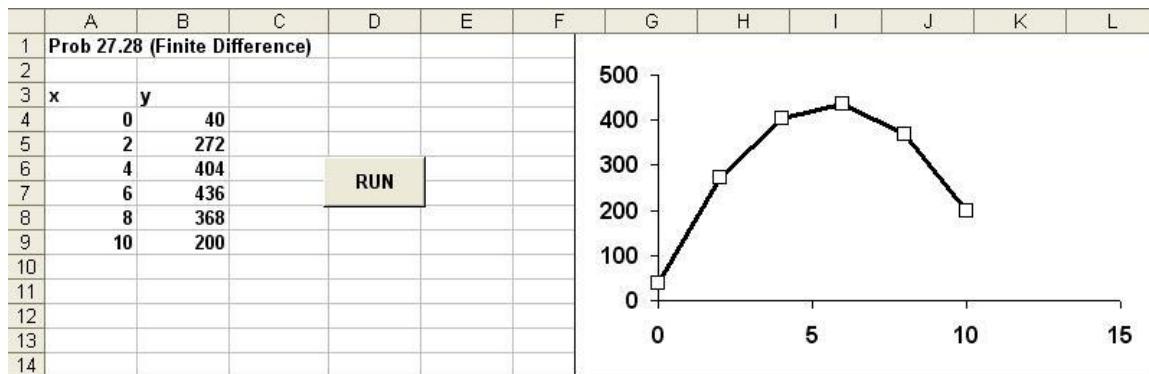
or substituting $\Delta x = 2$ and collecting terms yields

$$-T_{i+1} + 2T_i - T_{i-1} = 100$$

This equation can be written for each node with the result

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 140 \\ 100 \\ 100 \\ 300 \end{Bmatrix}$$

These equations can be solved with methods such as the tridiagonal solver, the Gauss-Seidel method or *LU* Decomposition. The following solution was computed using Excel's Minverse and Mmult functions:



27.29 First, the 2nd-order ODE can be reexpressed as the following system of 1st-order ODE's

$$\begin{aligned} \frac{dT}{dx} &= z \\ \frac{dz}{dx} &= -(0.12x^3 - 2.4x^2 + 12x) \end{aligned}$$

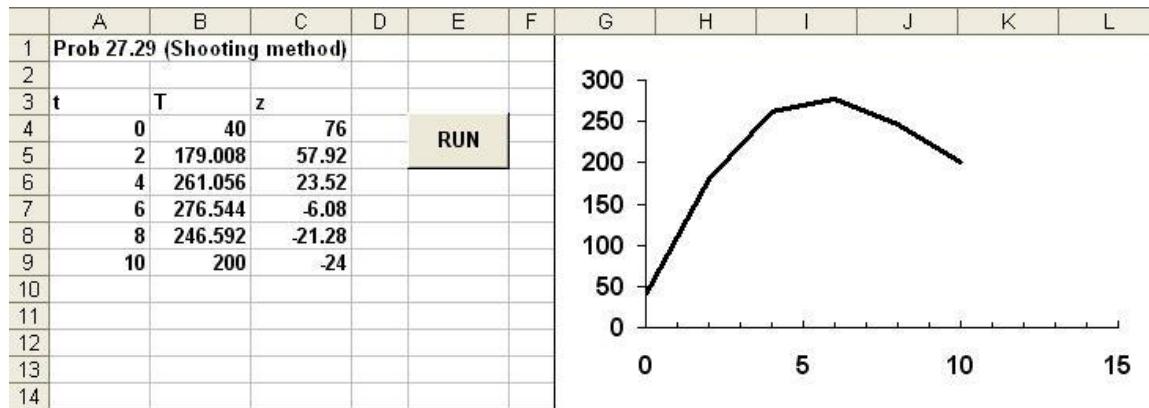
(a) Shooting method: These can be solved for two guesses for the initial condition of z . For our cases we used -1 and -0.5 . We solved the ODEs with the 4th-order RK method using a step size of 0.125. The results are

$z(0)$	-1	-0.5
$T(10)$	-570	-565

These values can then be used to derive the correct initial condition,

$$z(0) = -1 + \frac{-0.5 + 1}{-565 - (-570)} (200 - (-570)) = 76$$

The resulting fit is displayed below:



(b) Finite difference: Centered finite differences can be substituted for the second and first derivatives to give,

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + 0.12x_i^3 - 2.4x_i^2 + 12x_i = 0$$

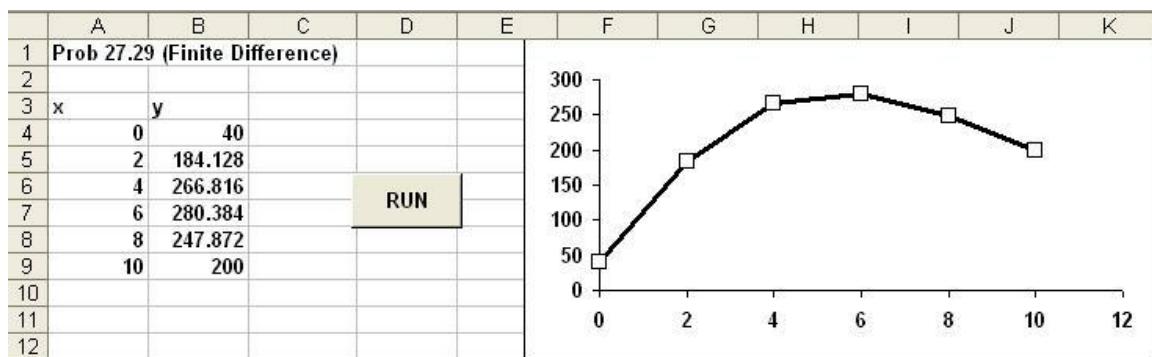
or substituting $\Delta x = 2$ and collecting terms yields

$$-T_{i+1} + 2T_i - T_{i-1} = \Delta x^2 (0.12x_i^3 - 2.4x_i^2 + 12x_i)$$

This equation can be written for each node with the result

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 101.44 \\ 69.12 \\ 46.08 \\ 215.36 \end{Bmatrix}$$

These equations can be solved with methods such as the tridiagonal solver, the Gauss-Seidel method or *LU* Decomposition. The following solution was computed using Excel's Minverse and Mmult functions:



CHAPTER 28

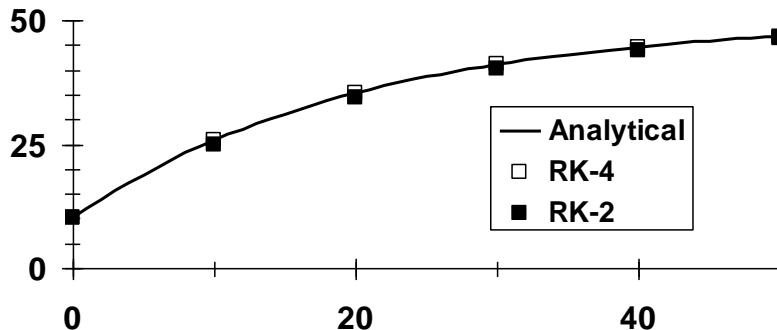
28.1 The solution with the 2nd-order RK (Heun without corrector) can be laid out as

<i>t</i>	<i>c</i>	<i>k</i> ₁	<i>c</i>	<i>k</i> ₂	<i>ϕ</i>
0	10	2	30	1	1.5
10	25	1.25	37.5	0.625	0.9375
20	34.375	0.78125	42.1875	0.390625	0.585938
30	40.23438	0.488281	45.11719	0.244141	0.366211
40	43.89648	0.305176	46.94824	0.152588	0.228882
50	46.1853	0.190735	48.09265	0.095367	0.143051

For the 4th-order RK, the solution is

<i>t</i>	<i>c</i>	<i>k</i> ₁	<i>c</i> _{mid}	<i>k</i> ₂	<i>c</i> _{mid}	<i>k</i> ₃	<i>c</i> _{end}	<i>k</i> ₄	<i>ϕ</i>
0	10	2	20	1.5	17.5	1.625	26.25	1.1875	1.572917
10	25.72917	1.213542	31.79688	0.910156	30.27995	0.986003	35.58919	0.72054	0.9544
20	35.27317	0.736342	38.95487	0.552256	38.03445	0.598278	41.25594	0.437203	0.579102
30	41.06419	0.446791	43.29814	0.335093	42.73965	0.363017	44.69436	0.265282	0.351382
40	44.57801	0.2711	45.93351	0.203325	45.59463	0.220268	46.78069	0.160965	0.213208
50	46.71009	0.164495	47.53257	0.123371	47.32695	0.133652	48.04662	0.097669	0.129369

A plot of both solutions along with the analytical result is displayed below:



28.2 The mass-balance equations can be written as

$$\frac{dc_1}{dt} = -0.16c_1 + 0.06c_3 + 1$$

$$\frac{dc_2}{dt} = 0.2c_1 - 0.2c_2$$

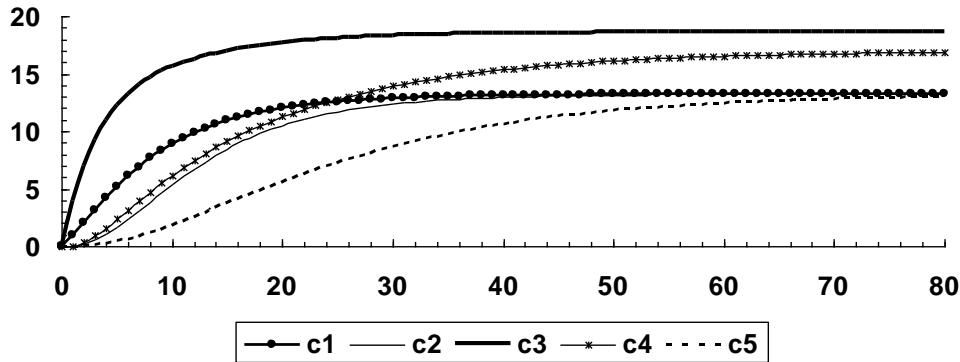
$$\frac{dc_3}{dt} = 0.05c_2 - 0.25c_3 + 4$$

$$\frac{dc_4}{dt} = 0.0875c_3 - 0.125c_4 + 0.0375c_5$$

$$\frac{dc_5}{dt} = 0.04c_1 + 0.02c_2 - 0.06c_5$$

Selected solution results (Euler's method) are displayed below, along with a plot of the results.

t	c₁	c₂	c₃	c₄	c₅	dc₁/dt	dc₂/dt	dc₃/dt	dc₄/dt	dc₅/dt
0	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	4.0000	0.0000	0.0000
1	1.0000	0.0000	4.0000	0.0000	0.0000	1.0800	0.2000	3.0000	0.3500	0.0400
2	2.0800	0.2000	7.0000	0.3500	0.0400	1.0872	0.3760	2.2600	0.5703	0.0848
3	3.1672	0.5760	9.2600	0.9203	0.1248	1.0488	0.5182	1.7138	0.6999	0.1307
4	4.2160	1.0942	10.9738	1.6201	0.2555	0.9839	0.6244	1.3113	0.7673	0.1752
5	5.1999	1.7186	12.2851	2.3874	0.4307	0.9051	0.6963	1.0147	0.7927	0.2165
•										
•										
•										
76	13.2416	13.2395	18.6473	16.8515	12.9430	0.0002	0.0004	0.0001	0.0106	0.0179
77	13.2418	13.2399	18.6475	16.8621	12.9609	0.0002	0.0004	0.0001	0.0099	0.0168
78	13.2419	13.2403	18.6476	16.8720	12.9777	0.0001	0.0003	0.0001	0.0093	0.0158
79	13.2421	13.2406	18.6477	16.8813	12.9935	0.0001	0.0003	0.0001	0.0088	0.0149
80	13.2422	13.2409	18.6478	16.8901	13.0084	0.0001	0.0003	0.0001	0.0082	0.0140



Finally, MATLAB can be used to determine the eigenvalues and eigenvectors:

```
>> a=[.16 -.06 0 0 0;-.2 .2 0 0 0;0 -.05 .25 0 0;0 0 -.0875 .125 -.0375;-.04
-.02 0 0 .06]
a =
    0.1600   -0.0600         0         0         0
   -0.2000    0.2000         0         0         0
      0   -0.0500    0.2500         0         0
      0         0   -0.0875    0.1250   -0.0375
   -0.0400   -0.0200         0         0    0.0600

>> [v,d]=eig(a)
v =
      0         0         0   -0.1039    0.2604
      0         0         0   -0.1582   -0.5701
      0    0.8192         0   -0.0436    0.6892
```

```

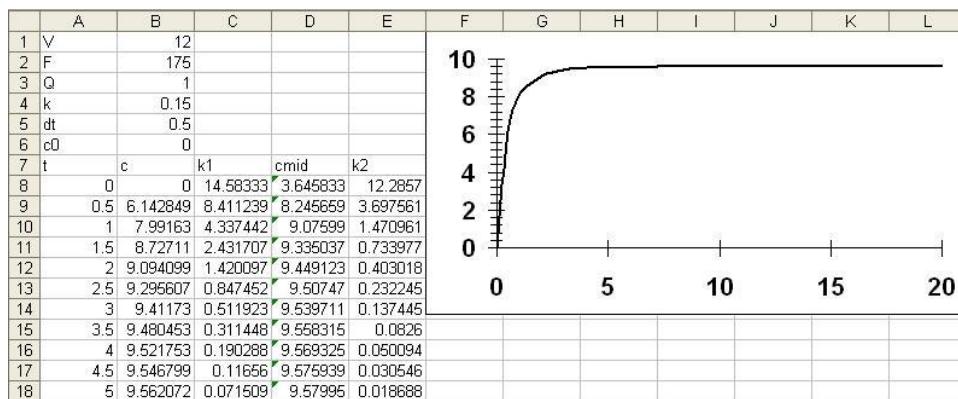
1.0000   -0.5735    0.4997    0.4956   -0.3635
          0           0       0.8662    0.8466    0.0043
d =
0.1250      0         0         0         0
0       0.2500      0         0         0
0       0       0.0600      0         0
0       0         0       0.0686      0
0       0         0         0     0.2914

```

28.3 Substituting the parameters into the differential equation gives

$$\frac{dc}{dt} = 14.583333 - 0.0833333c - 0.15c^2$$

The mid-point method can be applied with the result:



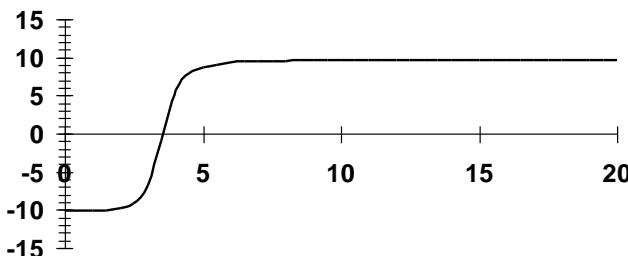
The results are approaching a steady-state value of 9.586267.

Challenge question:

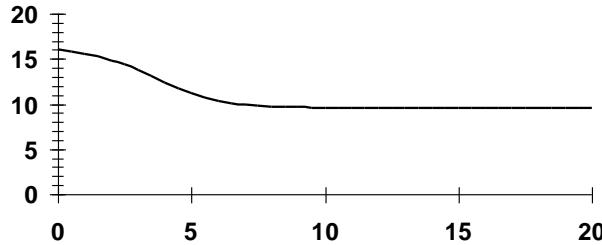
The steady state form (i.e., $dc/dt = 0$) of the equation is $0 = 14.58333 - 0.083333c - 0.15c^2$, which can be solved for 9.586267 and -10.1418. Thus, there is a negative root.

If we put in the initial y value as -10.1418 (or higher precision), the solution will stay at the negative root for awhile until roundoff errors may lead to the solution going unstable. If we use an initial value less than the negative root, the solution will also go unstable.

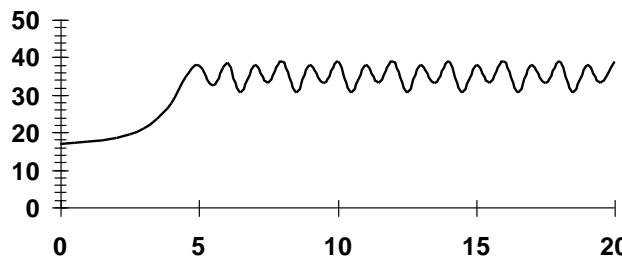
However, if we pick a value that is slightly higher (as per machine precision), it will gravitate towards the positive root. For example if we use -10.14



This kind of behavior will also occur for higher initial conditions. For example, using an initial condition of 16 gives,



However, if we start to use even higher initial conditions, the solution will again go unstable. For example, if we use an initial condition of 17, the result is

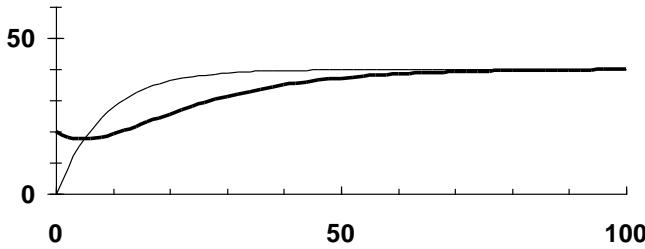


Therefore, it looks like initial conditions roughly in the range from -10.14 up to about 16.5 will yield stable solutions that converge on the steady-state solution of 9.586 . Note that this range depends on our choice of step size.

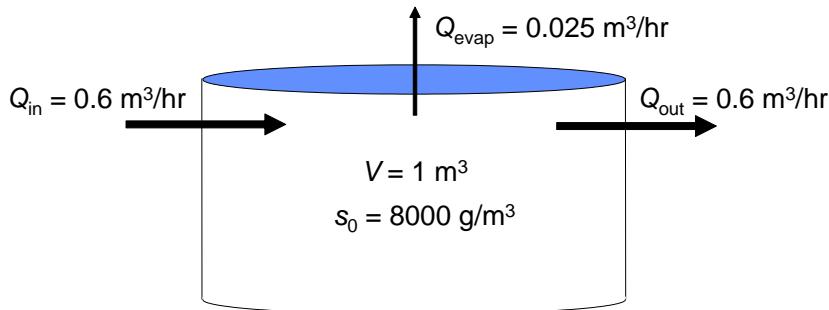
28.4 The first steps of the solution are shown below along with a plot. Notice that a value of the inflow concentration at the end of the interval (c_{in-end}) is required to calculate the k_2 's correctly.

t	c_{in}	c	k_1	c_{end}	c_{in-end}	k_2	ϕ
0	0	20	-1.2	17.6	8.534886	-0.54391	-0.87195
2	8.534886	18.25609	-0.58327	17.08955	15.24866	-0.11045	-0.34686
4	15.24866	17.56237	-0.13882	17.28472	20.52991	0.194711	0.027944
6	20.52991	17.61826	0.174699	17.96766	24.68428	0.402998	0.288848
8	24.68428	18.19595	0.3893	18.97455	27.95223	0.538661	0.46398
10	27.95223	19.12391	0.529699	20.18331	30.52289	0.620375	0.575037

In the following plot, the inflow concentration (thin line) and the outflow concentration (heavy line) are both shown.



28.5 The system is as depicted below:



(a) The mass of water in the tank can be modeled with a simple mass balance

$$\frac{dV}{dt} = Q_{\text{in}} - Q_{\text{out}} - Q_{\text{evap}} = 0.6 - 0.6 - 0.025 = -0.025$$

With the initial condition that $V = 1 \text{ m}^3$ at $t = 0$, this equation can be integrated to yield,

$$V = 1 - 0.025t$$

Thus, the time to empty the tank ($M = 0$) can be calculated as $t = 1/0.025 = 40 \text{ hrs}$.

(b) The concentration in the tank over this period can be computed in several ways. The simplest is to compute the mass of salt in the tank over time by solving the following differential equation:

$$\frac{dm}{dt} = Q_{\text{in}}s_{\text{in}} - Q_{\text{out}}s \quad (1)$$

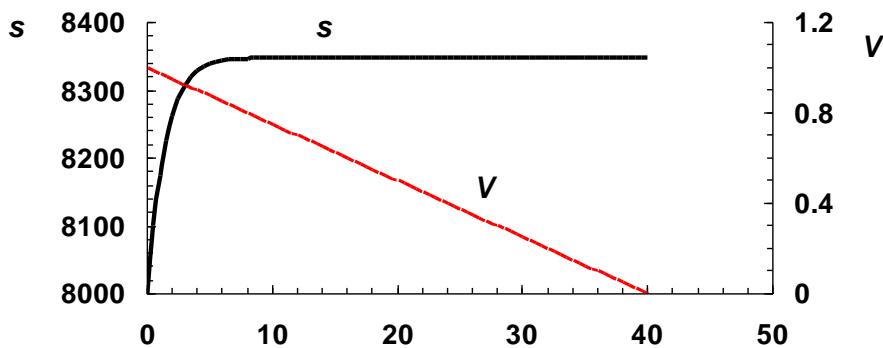
where m = the mass of salt in the tank. The salt concentration in the tank, s , is the ratio of the mass of salt to the volume of water

$$s = \frac{m}{V} = \frac{m}{1 - 0.025t} \quad (2)$$

Thus, we can integrate Eq. 1 to determine the mass at each time step and then use Eq. 2 to determine the corresponding salt concentration. The first few steps of the solution of the ODE using Euler's method are tabulated below. In addition, a graph of the entire solution is also

displayed. Note that the concentration asymptotically approaches a constant value of 8,347.826 g/m³ as the solution evolves.

t	Vol	m	s	dm/dt
0	1	8000	8000	0
0.5	0.9875	8000	8101.266	-60.7595
1	0.975	7969.62	8173.969	-104.382
1.5	0.9625	7917.429	8225.901	-135.54
2	0.95	7849.659	8262.799	-157.679
2.5	0.9375	7770.819	8288.874	-173.324



Recognize that a singularity occurs at $t = 40$, because the tank would be totally empty at this point.

28.6 A heat balance for the sphere can be written as

$$\frac{dH}{dt} = hA(T_a - T)$$

The heat gain can be transformed into a volume loss by considering the latent heat of fusion.

$$\frac{dV}{dt} = -\frac{hA}{\rho L_f}(T_a - T) \quad (1)$$

where ρ = density $\cong 0.917 \text{ kg/m}^3$ and L_f = latent heat of fusion $\cong 333 \text{ kJ/kg}$. The volume and area of a sphere are computed by

$$V = \frac{4}{3}\pi r^3 \quad A = 4\pi r^2 \quad (2)$$

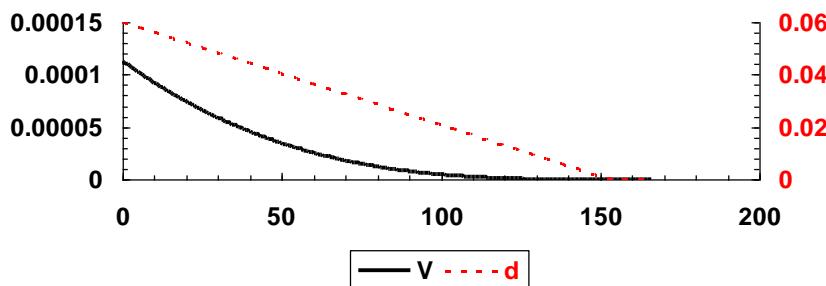
These can be combined with (1) to yield,

$$\frac{dV}{dt} = \frac{h4\pi \left(\frac{3V}{4\pi} \right)^{2/3}}{\rho L_f} (T - T_a)$$

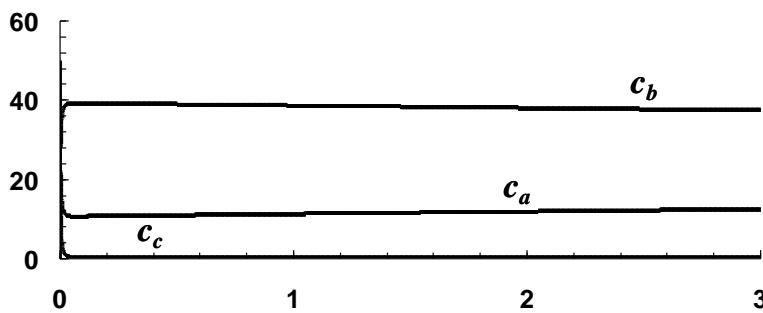
where $T = 0^\circ\text{C}$. This equation can be integrated along with the initial condition,

$$V_0 = \frac{4}{3}\pi(0.03)^3 = 0.000113 \text{ m}^3$$

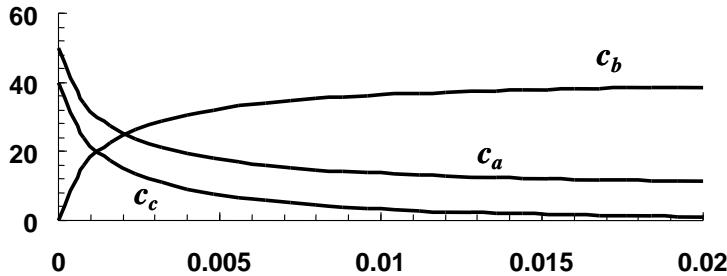
to yield the resulting volume as a function of time. This result can be converted into diameter using (2). Both results are shown on the following plot which indicates that it takes about 150 s for the ice to melt.



28.7 The system for this problem is stiff. Thus, the use of a simple explicit Runge-Kutta scheme would involve using a very small time step in order to maintain a stable solution. A solver designed for stiff systems was used to generate the solution shown below. Two views of the solution are given. The first is for the entire solution domain.



In addition, we can enlarge the initial part of the solution to illustrate the fast transients that occur as the solution moves from its initial conditions to its dominant trajectories.



28.8 Several methods could be used to obtain a solution for this problem (e.g., finite-difference, shooting method, finite-element). The finite-difference approach is straightforward:

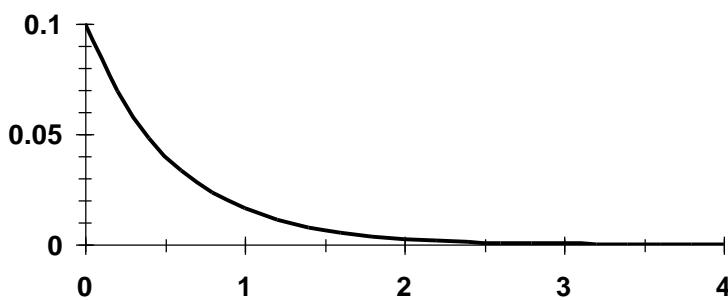
$$D \frac{A_{i-1} - 2A_i + A_{i+1}}{\Delta x^2} - kA_i = 0$$

Substituting parameter values and collecting terms gives

$$-1.5 \times 10^{-6} A_{i-1} + (3 \times 10^{-6} + 5 \times 10^{-6} \Delta x^2) A_i - 1.5 \times 10^{-6} A_{i+1} = 0$$

Using a $\Delta x = 0.2$ cm this equation can be written for all the interior nodes. The resulting linear system can be solved with an approach like the Gauss-Seidel method. The following table and graph summarize the results.

x	A	x	A	x	A	x	A
0	0.1						
0.2	0.069544	1.2	0.011267	2.2	0.001779	3.2	0.000257
0.4	0.048359	1.4	0.007814	2.4	0.001224	3.4	0.000166
0.6	0.033621	1.6	0.005415	2.6	0.000840	3.6	9.93E-05
0.8	0.023368	1.8	0.003748	2.8	0.000574	3.8	4.65E-05
1	0.016235	2	0.002591	3	0.000389	4	0



28.9 (a) The temperature of the body can be determined by integrating Newton's law of cooling to give,

$$T(t) = T_o e^{-Kt} + T_a (1 - e^{-Kt})$$

This equation can be solved for K ,

$$K = -\frac{1}{t} \ln \frac{T(t) - T_a}{T_o - T_a}$$

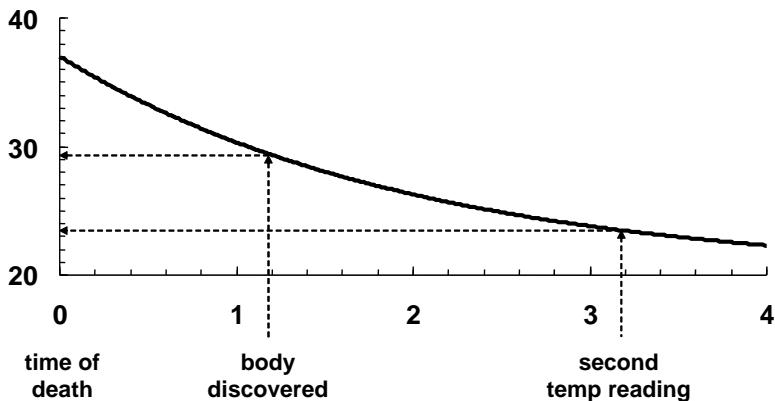
Substituting the values yields

$$K = -\frac{1}{2} \ln \frac{23.5 - 20}{29.5 - 20} = 0.499264 / \text{hr}$$

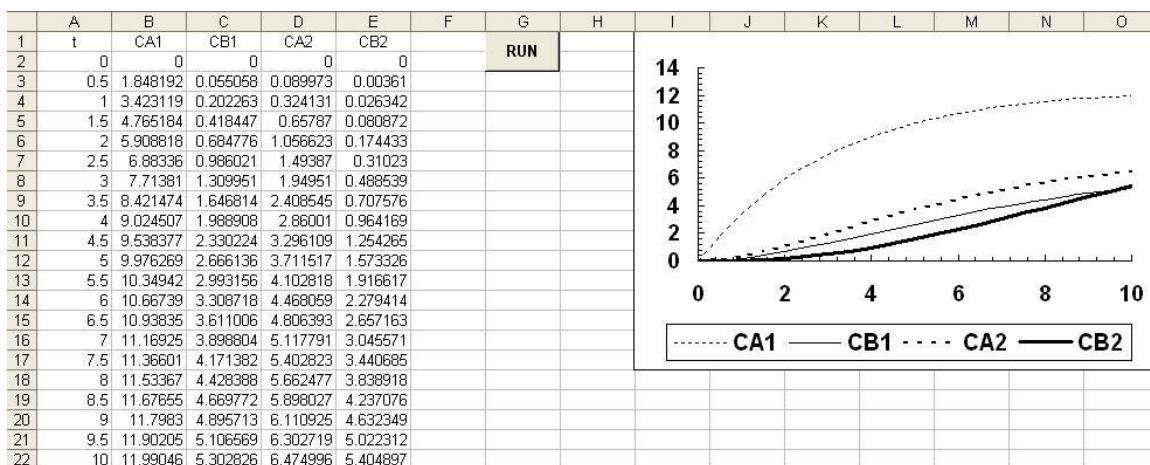
The time of death can then be computed as

$$t_d = -\frac{1}{K} \ln \frac{T(t_d) - T_a}{T_o - T_a} = -\frac{1}{0.499264} \ln \frac{37 - 20}{29.5 - 20} = -1.16556 \text{ hr}$$

Thus, the person died 1.166 hrs prior to being discovered. The non-self-starting Heun yielded the following time series of temperature. For convenience, we have redefined the time of death as $t = 0$:

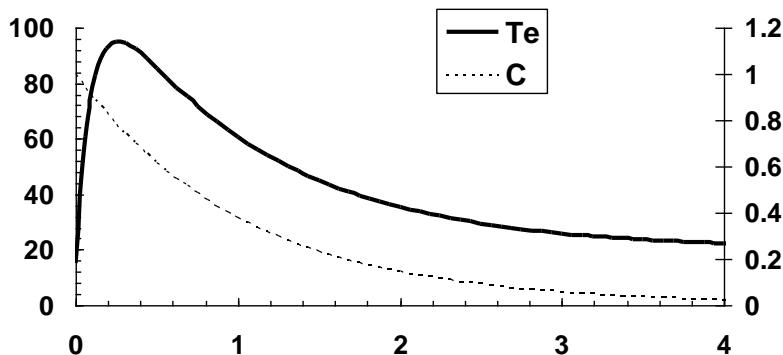


28.10 The classical 4th order RK method yields



28.11 The classical 4th order RK method yields

<i>t</i>	<i>C</i>	<i>Te</i>
0	1	15
0.0625	0.941261	60.79675
0.125	0.885808	82.90298
0.1875	0.83356	92.37647
0.25	0.784373	95.1878
0.3125	0.738086	94.54859
0.375	0.694535	92.18087
0.4375	0.653562	89.00406
0.5	0.615016	85.50575
0.5625	0.578753	81.94143
0.625	0.544638	78.4421
0.6875	0.512542	75.07218
0.75	0.482346	71.86061
0.8125	0.453936	68.8176
0.875	0.427205	65.94362
0.9375	0.402055	63.23421
1	0.378391	60.68253
•		
•		
•		



28.12 In MATLAB, the first step is to set up a function to hold the differential equations:

```
function dc = dcdtstiff(t, c)
dc = [-0.013*c(1)-1000*c(1)*c(3); -2500*c(2)*c(3); -0.013*c(1)-
1000*c(1)*c(3)-2500*c(2)*c(3);
```

Then, an ODE solver like the function `ode45` can be implemented as in

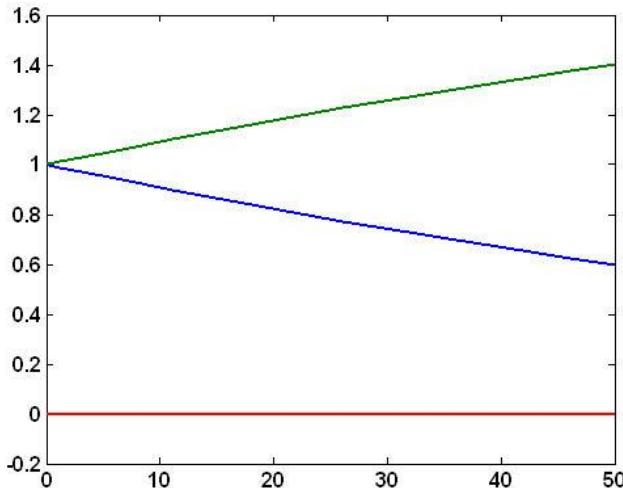
```
>> tspan=[0,50];
>> y0=[1,1,0];
>> [t,y]=ode45(@dcdtstiff,tspan,y0);
```

If this is done, the solution will take a relatively long time to compute the results. In contrast, because it is expressly designed to handle stiff systems, a function like `ode23s` yields results almost instantaneously.

```
>> [t,y]=ode23s(@dcdtstiff,tspan,y0);
```

In either case, a plot of the results can be developed as

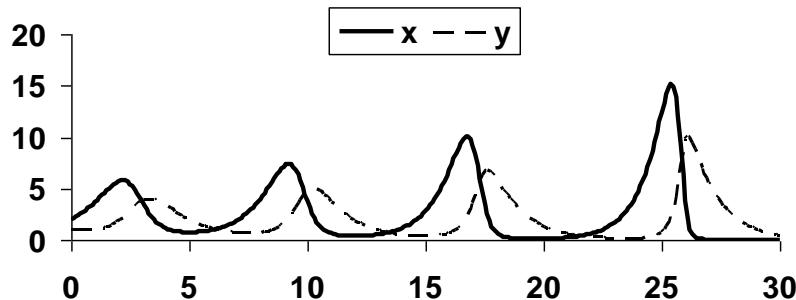
```
>> plot(t,y)
```



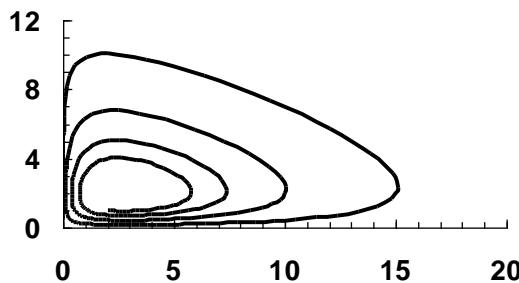
28.13 (a) The first few steps of Euler's method are shown in the following table

<i>t</i>	<i>x</i>	<i>y</i>
0	2	1
0.1	2.12	0.98
0.2	2.249744	0.963928
0.3	2.389598	0.951871
0.4	2.539874	0.943959
0.5	2.700807	0.940369

A plot of the entire simulation is shown below:



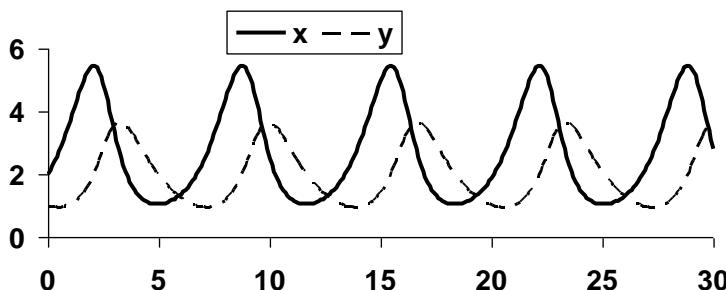
Notice that because the Euler method is lower order, the peaks are increasing, rather than repeating in a stable manner as time progresses. This result is reinforced when a state-space plot of the calculation is displayed.



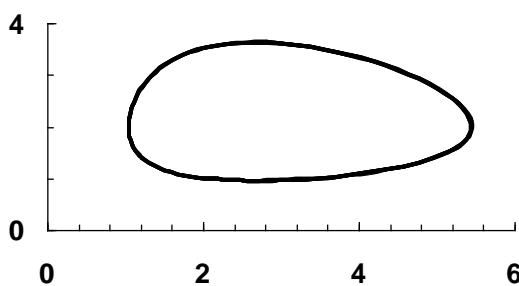
(b) The first few steps of the Heun method is shown in the following table

t	x	y
0	2	1
0.1	2.124872	0.981964
0.2	2.259707	0.968014
0.3	2.404809	0.958274
0.4	2.560393	0.95292
0.5	2.72655	0.952178

A plot of the entire simulation is shown below:



Notice that in contrast to the Euler method, the peaks are stable manner as time progresses. This result is also reinforced when a state-space plot of the calculation is displayed.



(c) The first few steps of the 4th-order RK method is shown in the following table

<i>t</i>	<i>x</i>	<i>y</i>
0	2	1
0.1	2.124862	0.982012
0.2	2.259682	0.968111
0.3	2.404758	0.958421
0.4	2.560303	0.953116
0.5	2.726403	0.952425

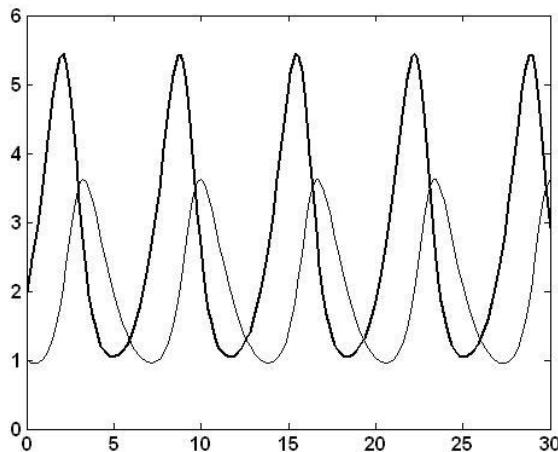
The results are quite close to those obtained with the Heun method in part (b). In fact, both the time series and state-space plots are indistinguishable from each other.

(d) In order to solve the problem in MATLAB, a function is first developed to hold the ODEs,

```
function yp=predprey(t,y)
yp=[1.2*y(1)-0.6*y(1)*y(2);-0.8*y(2)+0.3*y(1)*y(2)];
```

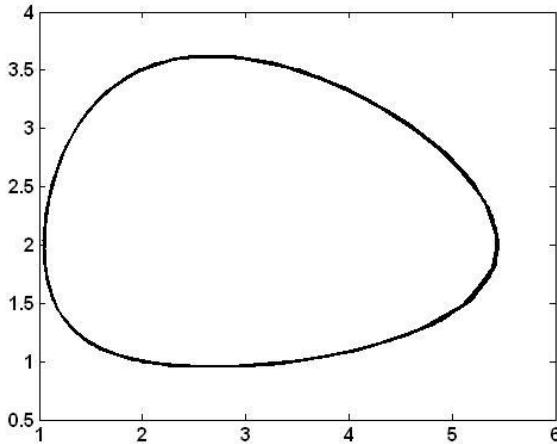
Then, an ODE solver like the function ode45 can be implemented as in

```
>> tspan=[0,30];
>> y0=[2,1];
>> [t,y]=ode45(@predprey,tspan,y0);
>> plot(t,y)
```



The state-space plot can be generated as

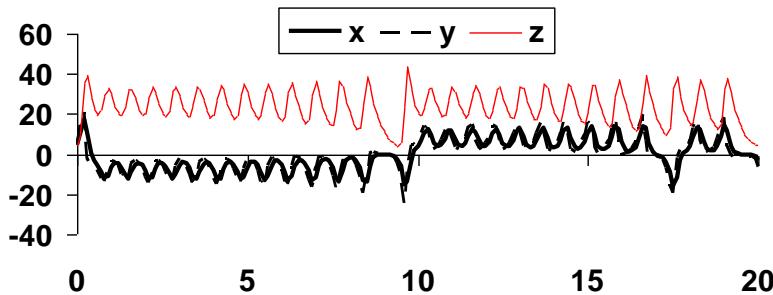
```
>> plot(y(:,1),y(:,2))
```



28.14 Using the step size of 0.1, (a) and (b) both give unstable results.

(c) The 4th-order RK method yields a stable solution. The first few values are shown in the following table. A time series plot of the results is also shown below. Notice how after about $t = 6$, this solution diverges from the double precision version in Fig. 28.9.

<i>t</i>	<i>x</i>	<i>y</i>	<i>z</i>
0	5	5	5
0.1	9.781469	17.07946	10.43947
0.2	17.70297	20.8741	35.89687
0.3	10.81088	-2.52924	39.30745
0.4	0.549577	-5.54419	28.07462
0.5	-3.16461	-5.84129	22.36889



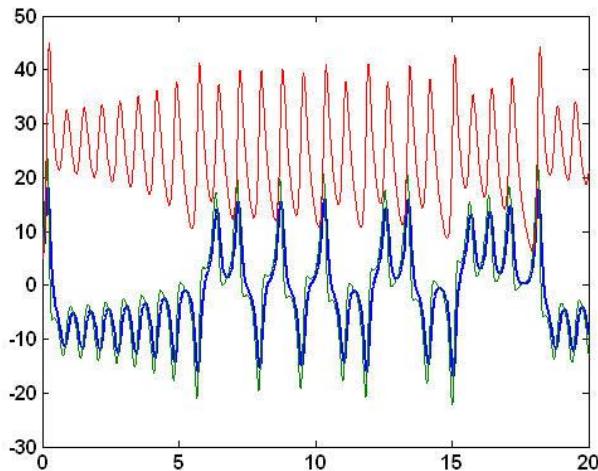
(d) In order to solve the problem in MATLAB, a function is first developed to hold the ODEs,

```
function yp=lorenz(t,y)
yp=[-10*y(1)+10*y(2);28*y(1)-y(2)-y(1)*y(3);-
2.666667*y(3)+y(1)*y(2)];
```

Then, an ODE solver like the function ode45 can be implemented as in

```
>> tspan=[0, 20];
```

```
>> y0=[5,5,5];
>> [t,y]=ode45(@lorenz,tspan,y0);
>> plot(t,y)
```

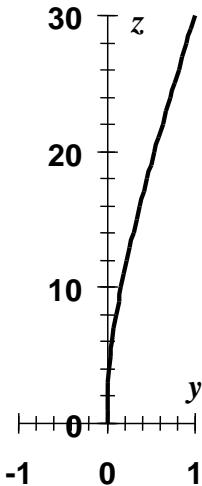


28.15 The second-order equation can be expressed as a pair of first-order equations,

$$\begin{aligned}\frac{dy}{dz} &= w \\ \frac{dw}{dz} &= \frac{f}{2EI}(L-z)^2\end{aligned}$$

We used Euler's method with $h = 1$ to obtain the solution:

<i>z</i>	<i>y</i>	<i>w</i>	<i>dy/dz</i>	<i>dw/dz</i>
0	0	0	0	0.00432
1	0	0.00432	0.00432	0.004037
2	0.00432	0.008357	0.008357	0.003763
3	0.012677	0.01212	0.01212	0.003499
4	0.024797	0.015619	0.015619	0.003245
5	0.040416	0.018864	0.018864	0.003
•				
•				
•				
26	0.8112	0.04524	0.04524	7.68E-05
27	0.85644	0.045317	0.045317	4.32E-05
28	0.901757	0.04536	0.04536	1.92E-05
29	0.947117	0.045379	0.045379	4.8E-06
30	0.992496	0.045384	0.045384	0

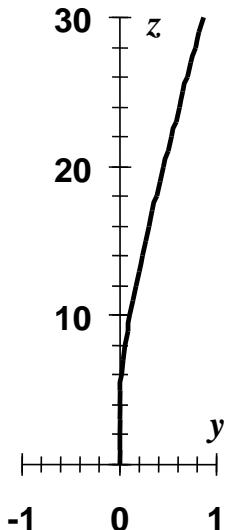


28.16 The second-order equation can be expressed as a pair of first-order equations,

$$\frac{dy}{dz} = w \quad \frac{dw}{dz} = \frac{200ze^{-2z/30}}{(5+z)2EI}(L-z)^2$$

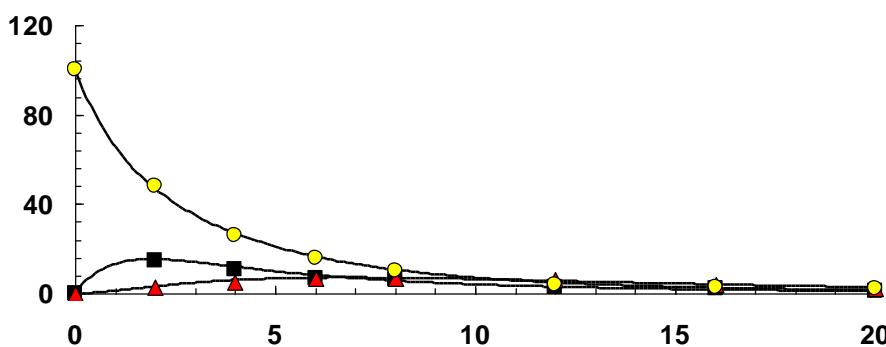
We used Euler's method with $h = 1$ to obtain the solution:

<i>z</i>	<i>f(z)</i>	<i>y</i>	<i>w</i>	<i>dy/dz</i>	<i>dw/dz</i>
0	0	0	0	0	0
1	31.18357	0	0	0	0.002098
2	50.0099	0	0.002098	0.002098	0.003137
3	61.40481	0.002098	0.005235	0.005235	0.003581
4	68.08252	0.007333	0.008816	0.008816	0.003682
5	71.65313	0.016148	0.012498	0.012498	0.003583
•					
•					
•					
26	29.63907	0.700979	0.040713	0.040713	3.79E-05
27	27.89419	0.741693	0.040751	0.040751	2.01E-05
28	26.24164	0.782444	0.040771	0.040771	8.4E-06
29	24.67818	0.823216	0.04078	0.04078	1.97E-06
30	23.20033	0.863995	0.040782	0.040782	0



28.17 This problem was solved using the Excel spreadsheet in a fashion similar to the last example in Sec. 28.1. We set up Euler's method to solve the 3 ODEs using guesses for the diffusion coefficients. Then we formed a column containing the squared residuals between our predictions and the measured values. Adjusting the diffusion coefficients with the Solver tool minimized the sum of the squares. At first, we assumed that the diffusion coefficients were zero. For this case the Solver did not converge on a credible answer. We then made guesses of 1×10^6 for both. This magnitude was based on the fact that the volumes were of this order of magnitude. The resulting simulation did not fit the data very well, but was much better than when we had guessed zero. When we used Solver, it converged on $E_{12} = 1.243 \times 10^6$ and $E_{13} = 2.264 \times 10^6$ which corresponded to a sum of the squares of residuals of 7.734. Some of the Euler calculations are displayed below along with a plot of the fit.

<i>t</i>	<i>c</i> ₁	<i>c</i> ₂	<i>c</i> ₃	<i>dc</i> ₁ / <i>dt</i>	<i>dc</i> ₂ / <i>dt</i>	<i>dc</i> ₃ / <i>dt</i>
0	0	0	100	22.63911	0	-45.2782
0.1	2.2639108		95.47218	19.91464	0.351651	-42.203
0.2	4.2553743	0.035165	91.25187	17.46867	0.655521	-39.3905
0.3	6.0022409	0.100717	87.31283	15.27375	0.916677	-36.816
0.4	7.5296162	0.192385	83.63123	13.30513	1.139684	-34.4575
0.5	8.8601294	0.306353	80.18548	11.54045	1.328649	-32.2948

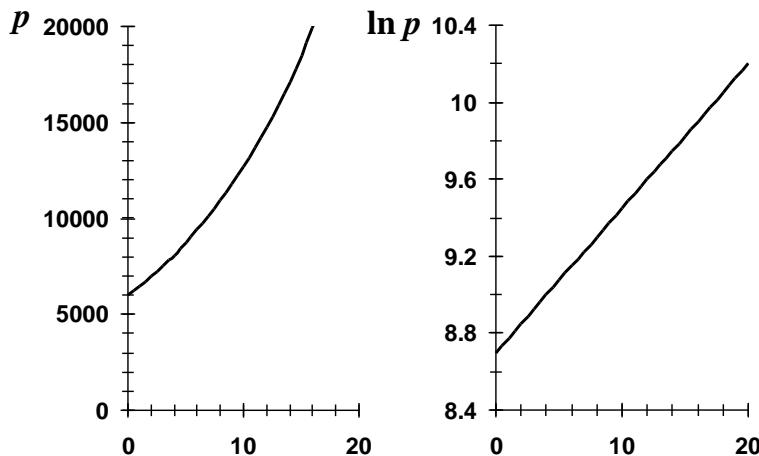


It should be noted that we made up the “measurements” for this problem using the 4th-order RK method with values for diffusive mixing of $E_{12} = 1 \times 10^6$ and $E_{13} = 2 \times 10^6$. We then used a random number generator to add some error to this “data.”

28.18 The Heun method without corrector can be used to compute

<i>t</i>	<i>p</i>	<i>k</i> ₁	<i>p</i> _{end}	<i>k</i> ₂	ϕ
0	6000	450	6225	466.875	458.4375
0.5	6229.219	467.1914	6462.814	484.7111	475.9512
1	6467.194	485.0396	6709.714	503.2286	494.1341
1.5	6714.261	503.5696	6966.046	522.4535	513.0115
2	6970.767	522.8075	7232.171	542.4128	532.6102
•					
•					
•					
18	23137.43	1735.308	24005.09	1800.382	1767.845
18.5	24021.36	1801.602	24922.16	1869.162	1835.382
19	24939.05	1870.429	25874.26	1940.57	1905.499
19.5	25891.8	1941.885	26862.74	2014.705	1978.295
20	26880.94	2016.071	27888.98	2091.673	2053.872

The results can be plotted. In addition, linear regression can be used to fit a straight line to $\ln p$ versus *t* to give $\ln p = 8.6995 + 0.075t$. Thus, as would be expected from a first-order model, the slope is equal to the growth rate of the population.

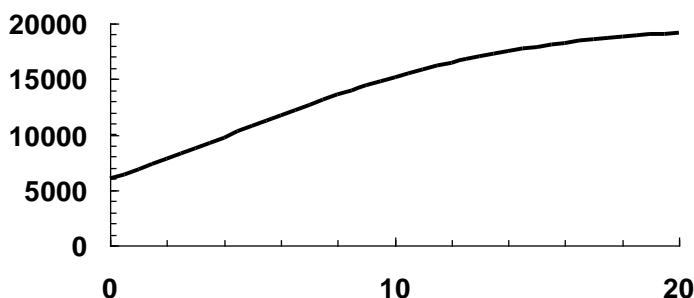


28.19 The Heun method can be used to compute

<i>t</i>	<i>p</i>	<i>k</i> ₁	<i>p</i> _{end}	<i>k</i> ₂	ϕ
0	6000	840	6420	871.836	855.918
0.5	6427.959	872.4052	6864.162	901.6652	887.0352
1	6871.477	902.1234	7322.538	928.312	915.2177
1.5	7329.085	928.6622	7793.417	951.3099	939.986
•					

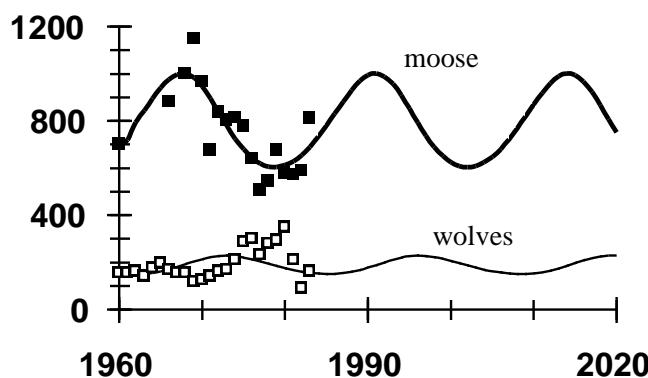
•						
18	18798.01	225.95	18910.99	205.9432	215.9466	
18.5	18905.98	206.8344	19009.4	188.3068	197.5706	
19	19004.77	189.1412	19099.34	172.02	180.5806	
19.5	19095.06	172.7988	19181.46	157.008	164.9034	
20	19177.51	157.7327	19256.38	143.1946	150.4637	

The results can be plotted.

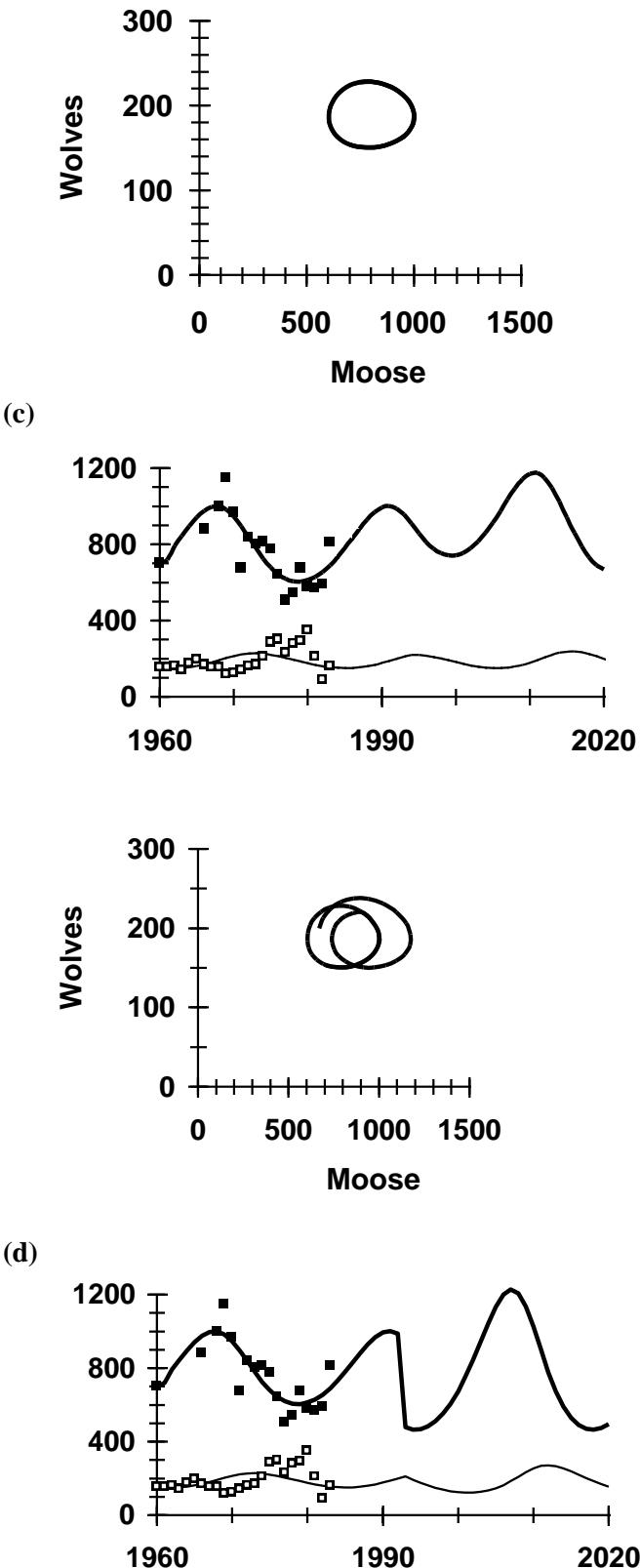


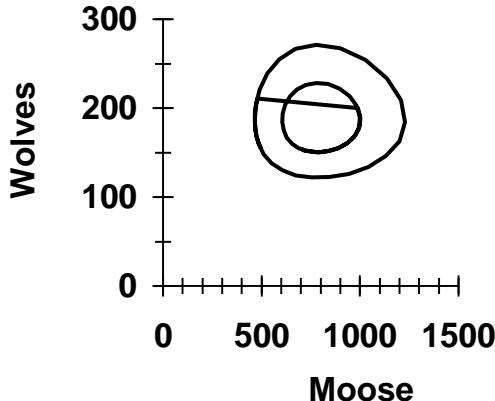
The curve is s-shaped. This shape occurs because initially the population is increasing exponentially since p is much less than p_{\max} . However, as p approaches p_{\max} , the growth rate decreases and the population levels off.

28.20 (a) Nonlinear regression (e.g., using the Excel solver option) can be used to minimize the sum of the squares of the residuals between the data and the simulation. The resulting estimates are: $a = 0.32823$, $b = 0.01231$, $c = 0.22445$, and $d = 0.00029$. The fit is:



(b) The results in state space are,



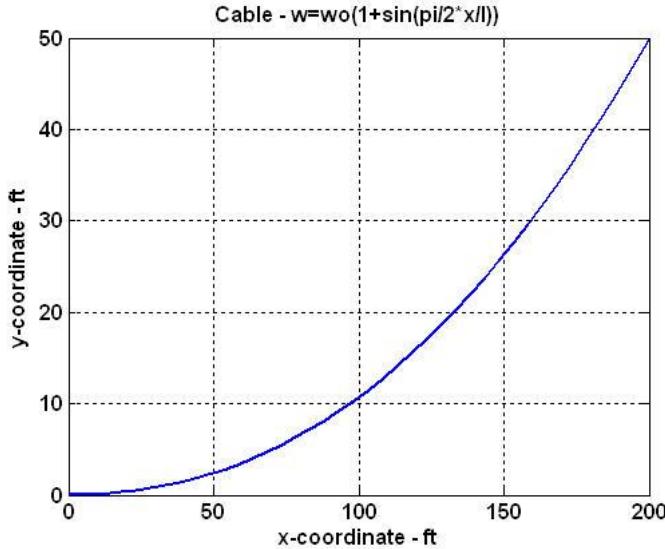


28.21 Main Program:

```
% Hanging static cable - w=w(x)
% Parabolic solution w=w(x)
% CUS Units (lb,ft,s)
% w = wo(1+sin(pi/2*x/l))
% Independent Variable x, xs=start x, xf=end x
% initial conditions [y(1)=cable y-coordinate, y(2)=cable slope];
es=0.5e-7;
xspan=[0,200];
ic=[0 0];
global wToP
wToP=0.0025;
[x,y]=ode45(@slp,xspan,ic);
yf(1)=y(length(x));
wTo(1)=wToP;
ea(1)=1;
wToP=0.002;
[x,y]=ode45(@slp,xspan,ic);
yf(2)=y(length(x));
wTo(2)=wToP;
ea(2)=abs( (yf(2)-yf(1))/yf(2) );
for k=3:10
    wTo(k)=wTo(k-1)+(wTo(k-1)-wTo(k-2))/(yf(k-1)-yf(k-2))*(50-yf(k-1));
    wToP=wTo(k);
    [x,y]=ode45(@slp,xspan,ic);
    yf(k)=y(length(x));
    ea(k)=abs( (yf(k)-yf(k-1))/yf(k) );
    if (ea(k)<=es)
        plot(x,y(:,1)); grid;
        xlabel('x-coordinate - ft'); ylabel('y-coordinate - ft');
        title('Cable - w=wo(1+sin(pi/2*x/200))');
        break
    end
end
```

Function 'slp':

```
function dxy=slp(x,y)
global wToP
dxy=[y(2);(wToP)*(1+sin(pi/2*x/200))];
```



28.22 This is an initial-value problem because the values of the variables are given at the start of the integration interval; i.e., at $x = 0$, $y = z = 0$. In order to obtain a numerical solution, the second-order differential equation can be expressed as a pair of first-order ODEs,

$$\frac{dy}{dx} = z$$

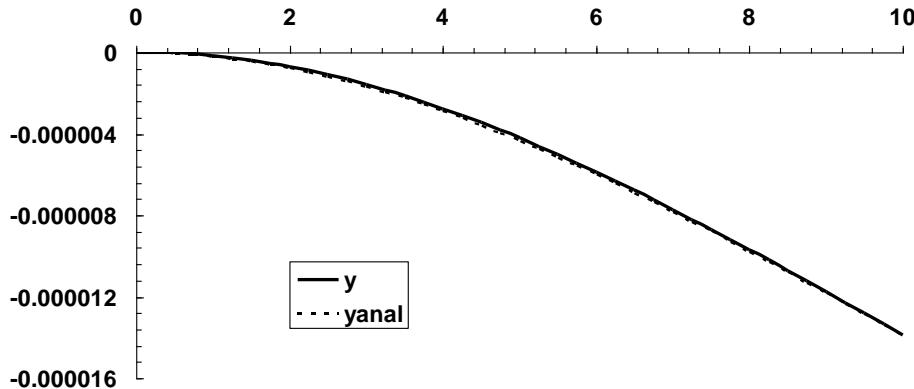
$$\frac{dz}{dx} = -\frac{P}{EI}(L - x)$$

These equations can be simply integrated with any of the one-step techniques from this part of the book. For example, the following shows the results of using the simple Euler's method with $h = 0.25$. Note that we have included the analytical solution in the last column.

x	y	z	dydx	dzdx	yanal
0	0	0	0	-4.1667E-07	0
0.25	0	-1.0417E-07	-1.0417E-07	-4.0625E-07	-1.2912E-08
0.5	-2.6042E-08	-2.0573E-07	-2.0573E-07	-3.9583E-07	-5.1215E-08
0.75	-7.7474E-08	-3.0469E-07	-3.0469E-07	-3.8542E-07	-1.1426E-07
1	-1.5365E-07	-4.0104E-07	-4.0104E-07	-3.7500E-07	-2.0139E-07
•					
•					
•					
9	-1.1758E-05	-2.1094E-06	-2.1094E-06	-4.1667E-08	-1.1813E-05
9.25	-1.2285E-05	-2.1198E-06	-2.1198E-06	-3.1250E-08	-1.2329E-05
9.5	-1.2815E-05	-2.1276E-06	-2.1276E-06	-2.0833E-08	-1.2848E-05
9.75	-1.3347E-05	-2.1328E-06	-2.1328E-06	-1.0417E-08	-1.3368E-05
10	-1.3880E-05	-2.1354E-06	-2.1354E-06	0.0000E+00	-1.3889E-05

Although there is a discrepancy, the results are fairly close. As shown below, the agreement between the approaches can be seen when the numerical and analytical solutions are plotted

on the same graph. Note that if a higher-order approach like the 4th-order RK method had been used, the results would be almost indistinguishable.



28.23 This is a boundary-value problem because the values of only one of the variables are known at two separate points; i.e., at $x = 0$, $y = 0$ and at $x = L$, $y = 0$. We can either use the shooting method or finite differences to obtain results. In the present solution, we will employ the shooting method. To do this, the second-order differential equation is first expressed as a pair of first-order ODEs,

$$\frac{dy}{dx} = z$$

$$\frac{dz}{dx} = \frac{wLx}{2EI} - \frac{wx^2}{2EI}$$

These can be solved using two guesses for the initial condition of z . For our cases we used $z(0) = -1 \times 10^{-6}$ and $z(0) = -2 \times 10^{-6}$. We solved the ODEs with Euler's method using a step size of 0.25. The results are

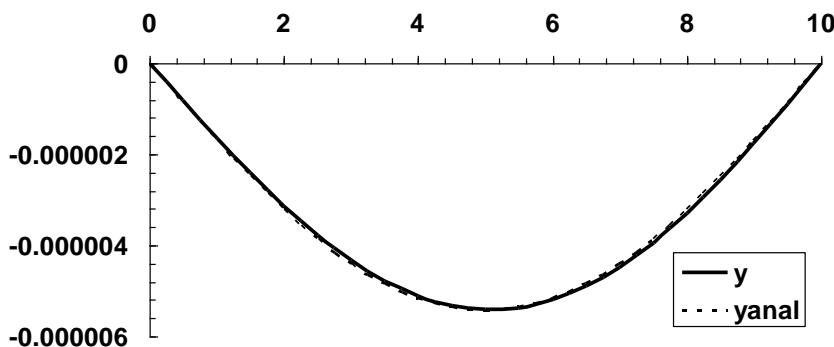
$z(0)$	-1E-6	-2E-6
$y(10)$	6.48275E-06	-3.51725E-06

These values can then be used to derive the correct initial condition that corresponds with $y(10) = 0$. Using linear interpolation, the result is -1.64827×10^{-6} . Using this value, the results of the final shot can be tabulated along with the analytical solution:

x	y	z	dy/dx	dz/dx	y_{anal}
0	0	-1.6483E-06	-1.6483E-06	0	0
0.25	-4.1207E-07	-1.6483E-06	-1.6483E-06	5.0781E-08	-4.3349E-07
0.5	-8.2414E-07	-1.6356E-06	-1.6356E-06	9.8958E-08	-8.6382E-07
0.75	-1.2330E-06	-1.6108E-06	-1.6108E-06	1.4453E-07	-1.2880E-06
1	-1.6357E-06	-1.5747E-06	-1.5747E-06	1.8750E-07	-1.7031E-06
•					
•					
•					

9	-1.7607E-06	1.7013E-06	1.7013E-06	1.8750E-07	-1.7031E-06
9.25	-1.3354E-06	1.7482E-06	1.7482E-06	1.4453E-07	-1.2880E-06
9.5	-8.9836E-07	1.7843E-06	1.7843E-06	9.8958E-08	-8.6382E-07
9.75	-4.5227E-07	1.8091E-06	1.8091E-06	5.0781E-08	-4.3349E-07
10	7.4115E-22	1.8218E-06	1.8218E-06	0.0000E+00	0.0000E+00

Note that although there is a discrepancy, the results are fairly close. In particular, notice that the end displacement is effectively zero. As shown below, the close agreement between the approaches can be seen when the numerical and analytical solutions are plotted on the same graph. In fact, if a higher-order approach like the 4th-order RK method had been used, the results would be almost indistinguishable.

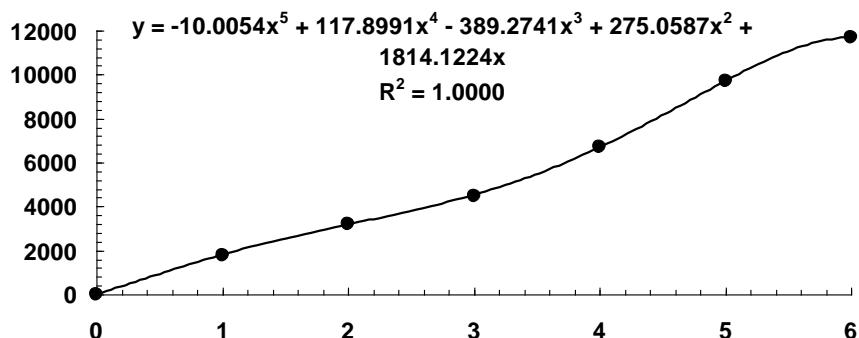


28.24 [Errata: In the first printing, a minus sign should be included in the differential equation:]

$$\frac{dh}{dt} = -\frac{\pi d^2}{4A(h)} \sqrt{2g(h+e)}$$

Later printings should have the correct equation.]

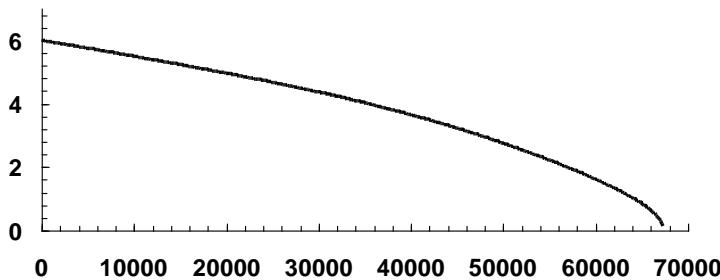
This problem can be approached in a number of ways. The simplest way is to fit the area-depth data with polynomial regression. A fifth-order polynomial with a zero intercept yields a perfect fit:



This polynomial can then be substituted into the differential equation to yield

$$\frac{dh}{dt} = -\frac{\pi d^2}{4(-10.0054h^5 + 117.8991h^4 - 389.2741h^3 + 275.0587h^2 + 1814.1224h)} \sqrt{2g(h+e)}$$

This equation can then be integrated numerically. This is a little tricky because a singularity occurs as the lake's depth approaches zero. Therefore, the software to solve this problem should be designed to terminate just prior to this occurring. For example, the software can be designed to terminate when a negative area is detected. As displayed below, the results indicate that the reservoir will empty in a little over 67,300 s.



28.25 The parameters can be substituted into force balance equations to give

$$\begin{aligned} (0.45 - \omega^2)X_1 - 0.2X_2 &= 0 \\ -0.24X_1 + (0.42 - \omega^2)X_2 - 0.18X_3 &= 0 \\ -0.225X_2 + (0.225 - \omega^2)X_3 &= 0 \end{aligned}$$

A MATLAB session can be conducted to evaluate the eigenvalues and eigenvectors as

```
>> A = [0.450 -0.200 0.000; -0.240 0.420 -0.180; 0.000 -0.225 0.225];
>> [v, d] = eig(A)

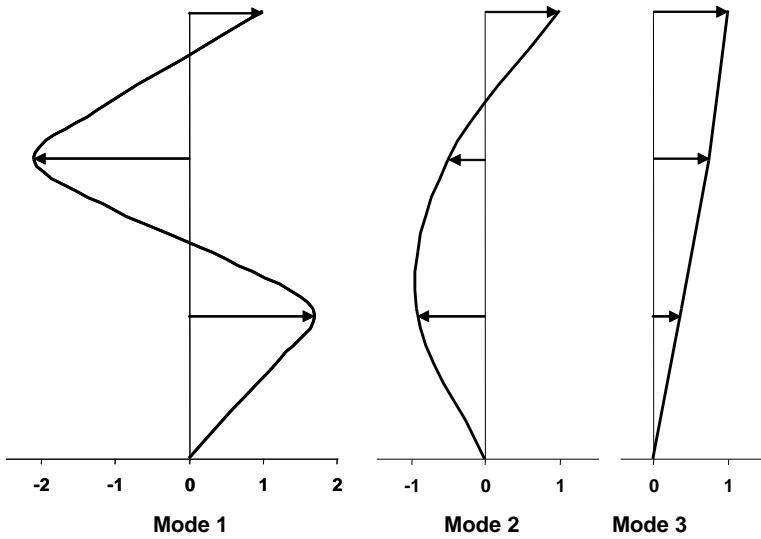
v =
-0.5879 -0.6344 0.2913
0.7307 -0.3506 0.5725
-0.3471 0.6890 0.7664

d =
0.6986 0 0
0 0.3395 0
0 0 0.0569
```

Therefore, the eigenvalues are 0.6986, 0.3395 and 0.0569. The corresponding eigenvectors are (normalizing so that the amplitude for the third floor is one),

$$\begin{Bmatrix} 1.693748 \\ -2.10516 \\ 1 \end{Bmatrix} \quad \begin{Bmatrix} -0.92075 \\ -0.50885 \\ 1 \end{Bmatrix} \quad \begin{Bmatrix} 0.380089 \\ 0.746999 \\ 1 \end{Bmatrix}$$

A graph can be made showing the three modes



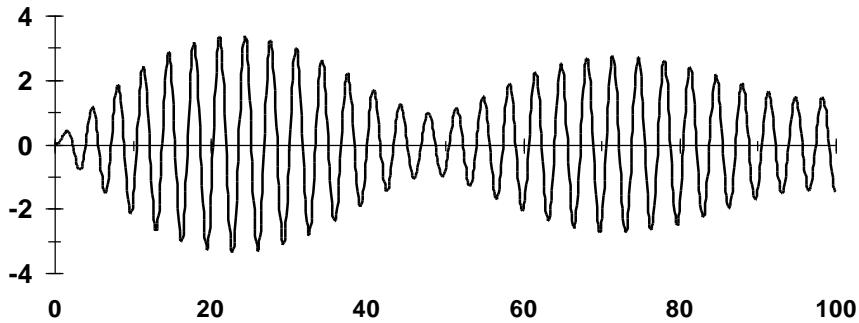
28.26 The second-order equation can be expressed as a pair of first-order equations,

$$\frac{dq}{dt} = i$$

$$\frac{di}{dt} = -0.025i - 4q + \sin 1.8708t$$

The parameters can be substituted and the system solved with the 4th-order RK method in double-precision with $h = 0.1$. A table showing the first few steps and a graph of the entire solution are shown below.

<i>t</i>	<i>i</i>	<i>q</i>	<i>k11</i>	<i>k12</i>	<i>imid</i>	<i>qmid</i>	<i>k21</i>	<i>k22</i>	<i>imid</i>	<i>qmid</i>	<i>k31</i>	<i>k32</i>	<i>iend</i>	<i>qend</i>	<i>k41</i>	<i>k42</i>	<i>phi1</i>	<i>phi2</i>
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0934	0.0000	0.0047	0.0000	0.0933	0.0047	0.0093	0.0005	0.1839	0.0093	0.0929	0.0031	
0.1	0.0093	0.0003	0.1843	0.0093	0.0185	0.0008	0.2734	0.0185	0.0230	0.0012	0.2714	0.0230	0.0364	0.0026	0.3542	0.0364	0.2713	0.0214
0.2	0.0364	0.0025	0.3548	0.0364	0.0542	0.0043	0.4324	0.0542	0.0580	0.0052	0.4287	0.0580	0.0793	0.0083	0.4972	0.0793	0.4290	0.0567
0.3	0.0793	0.0081	0.4978	0.0793	0.1042	0.0121	0.5580	0.1042	0.1072	0.0133	0.5530	0.1072	0.1346	0.0188	0.6017	0.1346	0.5536	0.1061
0.4	0.1347	0.0187	0.6021	0.1347	0.1648	0.0255	0.6399	0.1648	0.1667	0.0270	0.6338	0.1667	0.1981	0.0354	0.6583	0.1981	0.6346	0.1659

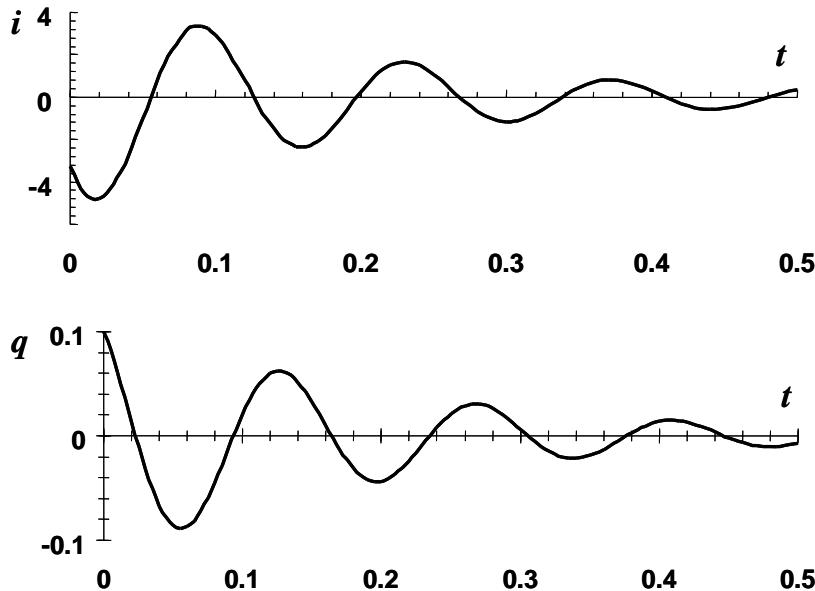


28.27 The second-order equation can be expressed as a pair of first-order equations,

$$\begin{aligned}\frac{dq}{dt} &= i \\ \frac{di}{dt} &= -\frac{R}{L}i - \frac{q}{CL}\end{aligned}$$

The parameters can be substituted and the system solved with the 4th-order RK method with $h = 0.005$. A table showing the first few steps and a graph of the entire solution are shown below.

<i>t</i>	<i>i</i>	<i>q</i>	<i>k11</i>	<i>k12</i>	<i>imid</i>	<i>qmid</i>	<i>k21</i>	<i>k22</i>	<i>imid</i>	<i>qmid</i>	<i>k31</i>	<i>k32</i>	<i>iend</i>	<i>qend</i>	<i>k41</i>	<i>k42</i>	<i>phi1</i>	<i>phi2</i>
0.000	-3.282	0.100	-167.18	-3.282	-3.699	0.092	-146.60	-3.699	-3.648	0.091	-145.02	-3.648	-4.007	0.082	-123.45	-4.007	-145.6	-3.664
0.005	-4.010	0.082	-123.26	-4.010	-4.318	0.072	-100.13	-4.318	-4.260	0.071	-99.17	-4.260	-4.506	0.060	-75.70	-4.506	-99.60	-4.279
0.010	-4.508	0.060	-75.499	-4.508	-4.696	0.049	-51.07	-4.696	-4.635	0.049	-50.74	-4.635	-4.761	0.037	-26.61	-4.761	-50.96	-4.655
0.015	-4.763	0.037	-26.396	-4.763	-4.828	0.025	-1.92	-4.828	-4.767	0.025	-2.21	-4.767	-4.774	0.013	21.39	-4.774	-2.21	-4.788
0.020	-4.774	0.013	21.594	-4.774	-4.720	0.001	44.92	-4.720	-4.661	0.001	44.07	-4.661	-4.553	-0.010	66.00	-4.553	44.26	-4.681



28.28 The equation can be solved analytically as

$$\frac{di}{dt} = -\frac{R}{L}i$$

$$\frac{di}{i} = -\frac{R}{L} dt$$

$$\ln i = -(R/L)t + C$$

$$C = \ln 0.5$$

$$\ln(i/0.5) = -(R/L)t$$

$$i = 0.5e^{-2t}$$

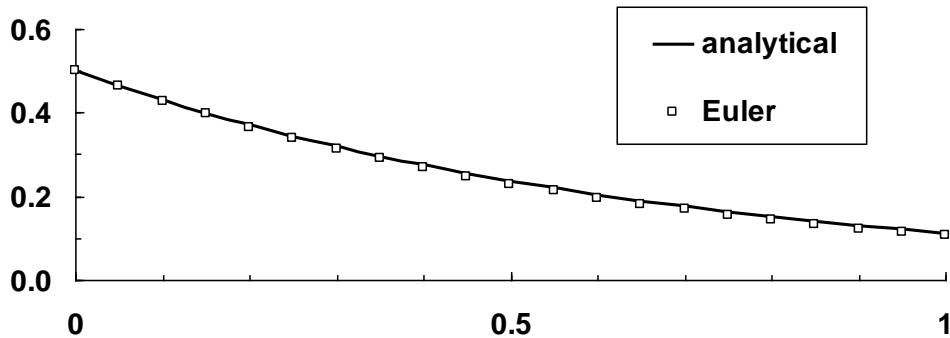
The numerical solution can be obtained by expressing the equation as

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$\frac{di}{dt} = -1.5i$$

and using Euler's method with $h = 0.05$ to solve for the current. Some selected values are shown, along with a plot of both the analytical and numerical result. A better match would be obtained by using a smaller step or a higher-order method.

<i>t</i>	<i>analytical</i>	<i>Euler</i>	<i>di/dt</i>
0	0.500000	0.500000	-0.750000
0.05	0.463872	0.462500	-0.693750
0.1	0.430354	0.427813	-0.641719
0.15	0.399258	0.395727	-0.593590
0.2	0.370409	0.366047	-0.549071
0.25	0.343645	0.338594	-0.507890

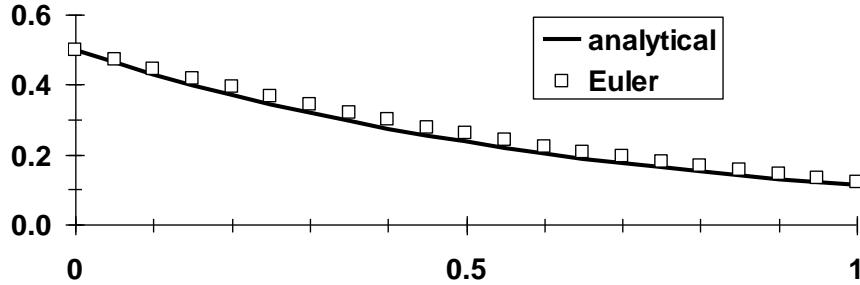


28.29 The numerical solution can be obtained by expressing the equation as

$$\frac{di}{dt} = -1.5(i - i^3)$$

and using Euler's method with $h = 0.05$ to solve for the current. Some selected values are shown, along with a plot of the numerical result. Note that the table and plot also show the analytical solution for the linear case computed in Prob. 28.19.

<i>t</i>	<i>analytical</i>	<i>Euler</i>	<i>di/dt</i>
0	0.500000	0.500000	-0.562500
0.05	0.463872	0.471875	-0.550207
0.1	0.430354	0.444365	-0.534931
0.15	0.399258	0.417618	-0.517175
0.2	0.370409	0.391759	-0.497451
0.25	0.343645	0.366887	-0.476253



28.30 Using an approach similar to Sec. 28.3, the system can be expressed in matrix form as

$$\begin{bmatrix} 1-\lambda & -1 \\ -1 & 2-\lambda \end{bmatrix} \begin{Bmatrix} i_1 \\ i_2 \end{Bmatrix} = \{0\}$$

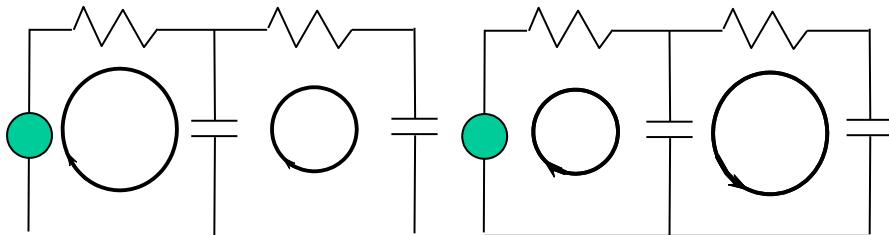
A package like MATLAB can be used to evaluate the eigenvalues and eigenvectors as in

```
>> a=[1 -1;-1 2];
>> [v,d]=eig(a)

v =
    0.8507    -0.5257
    0.5257     0.8507

d =
    0.3820         0
        0     2.6180
```

Thus, we can see that the eigenvalues are $\lambda = 0.382$ and 2.618 or natural frequencies of $\omega = 0.618/\sqrt{LC}$ and $1.618/\sqrt{LC}$. The eigenvectors tell us that these correspond to oscillations that coincide $(0.8507 \ 0.5257)$ and which run counter to each other $(-0.5257 \ 0.8507)$.



$$\omega = \frac{0.618}{\sqrt{LC}}$$

$$\omega = \frac{1.618}{\sqrt{LC}}$$

28.31 The differential equations to be solved are

linear :

$$\frac{d\theta}{dt} = v$$

$$\frac{dv}{dt} = -\frac{9.81}{1} \theta$$

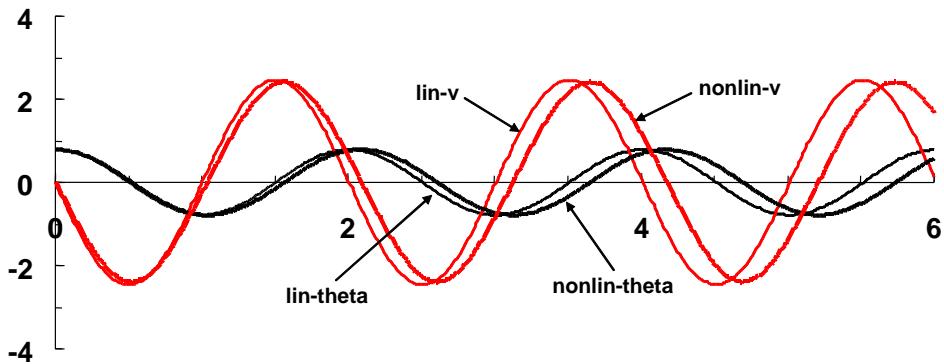
nonlinear :

$$\frac{d\theta}{dt} = v$$

$$\frac{dv}{dt} = -\frac{9.81}{1} \sin \theta$$

A few steps for the 4th-order RK solution of the nonlinear system are contained in the following table and a plot of both solutions is shown below.

<i>t</i>	<i>theta</i>	<i>v</i>	<i>k₁₁</i>	<i>k₁₂</i>	<i>theta</i>	<i>v</i>	<i>k₂₁</i>	<i>k₂₂</i>	<i>theta</i>	<i>v</i>	<i>k₃₁</i>	<i>k₃₂</i>	<i>theta</i>	<i>v</i>	<i>k₄₁</i>	<i>k₄₂</i>	ϕ_1	ϕ_2
0	0.785	0.000	0.000	-6.937	0.785	-0.035	-0.035	-6.937	0.785	-0.035	-0.035	-6.936	0.785	-0.069	-0.069	-6.934	-0.035	-6.936
0.01	0.785	-0.069	-0.069	-6.934	0.785	-0.104	-0.104	-6.932	0.785	-0.104	-0.104	-6.931	0.784	-0.139	-0.139	-6.927	-0.104	-6.931
0.02	0.784	-0.139	-0.139	-6.927	0.783	-0.173	-0.173	-6.922	0.783	-0.173	-0.173	-6.921	0.782	-0.208	-0.208	-6.915	-0.173	-6.921
0.03	0.782	-0.208	-0.208	-6.915	0.781	-0.242	-0.242	-6.908	0.781	-0.242	-0.242	-6.907	0.780	-0.277	-0.277	-6.898	-0.242	-6.907
0.04	0.780	-0.277	-0.277	-6.898	0.778	-0.311	-0.311	-6.888	0.778	-0.311	-0.311	-6.887	0.777	-0.346	-0.346	-6.876	-0.311	-6.887



28.32 The differential equations to be solved are

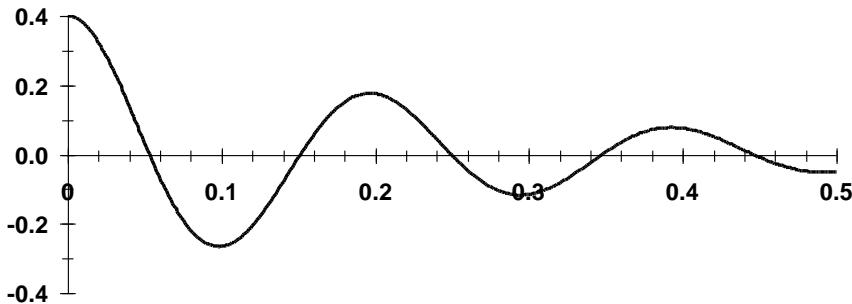
$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\frac{c}{m}v - \frac{k}{m}x$$

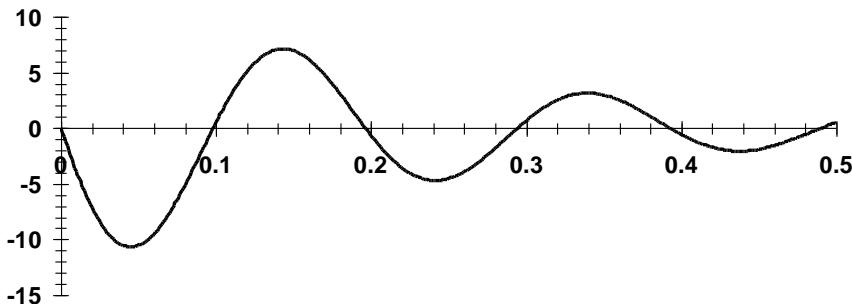
A few steps for the 2nd-order RK solution (Heun without iteration) are shown in the following table and plots of displacement and velocity are shown below.

<i>t</i>	<i>x</i>	<i>v</i>	<i>k₁₁</i>	<i>k₁₂</i>	<i>x</i>	<i>v</i>	<i>k₂₁</i>	<i>k₂₂</i>	ϕ_1	ϕ_2
0	0.4000	0.0000	0.0000	-416.67	0.4000	-0.4167	-0.4167	-413.19	-0.2083	-414.93
0.001	0.3998	-0.4149	-0.4149	-412.99	0.3994	-0.8279	-0.8279	-409.12	-0.6214	-411.05
0.002	0.3992	-0.8260	-0.8260	-408.92	0.3983	-1.2349	-1.2349	-404.65	-1.0304	-406.79
0.003	0.3981	-1.2328	-1.2328	-404.46	0.3969	-1.6372	-1.6372	-399.80	-1.4350	-402.13
0.004	0.3967	-1.6349	-1.6349	-399.61	0.3951	-2.0345	-2.0345	-394.58	-1.8347	-397.09
0.005	0.3949	-2.0320	-2.0320	-394.39	0.3928	-2.4264	-2.4264	-388.99	-2.2292	-391.69

Displacement:



Velocity:

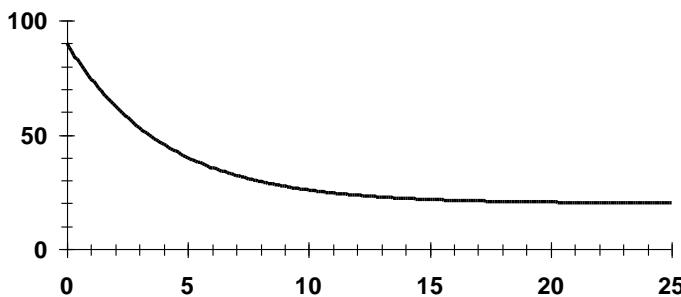


28.33 The differential equation to be solved is

$$\frac{dT}{dt} = 0.25(20 - T)$$

A few steps for the 2nd-order RK solution (Heun without iteration) are shown in the following table and a plot of temperature versus time is shown below.

<i>t</i>	<i>T</i>	<i>k</i> ₁₁	<i>T</i> _{end}	<i>k</i> ₂₁	<i>ϕ</i>
0	90	-17.5	88.25	-17.0625	-17.2813
0.1	88.27188	-17.068	86.56508	-16.6413	-16.8546
0.2	86.58641	-16.6466	84.92175	-16.2304	-16.4385
0.3	84.94256	-16.2356	83.31900	-15.8297	-16.0327
0.4	83.33929	-15.8348	81.75581	-15.4390	-15.6369
0.5	81.77560	-15.4439	80.23121	-15.0578	-15.2509



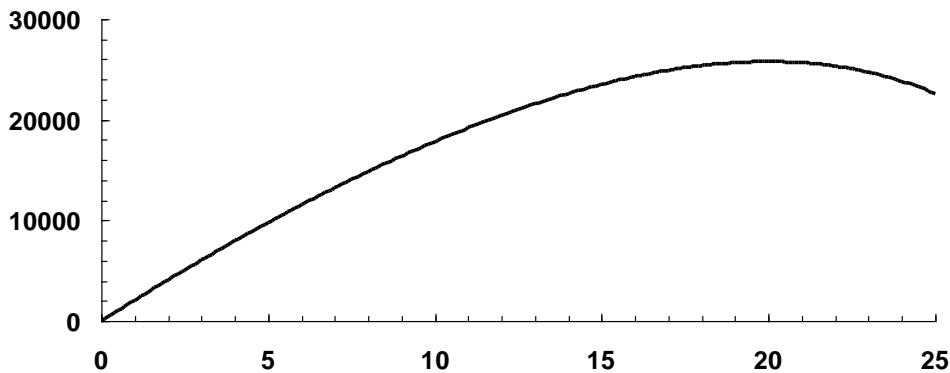
The temperature will drop to 40°C in approximately 5 minutes.

28.34 The differential equation to be solved is

$$\frac{dQ}{dt} = 0.5(12) \frac{100(20 - 2.5)(20 - t)}{100 - 2.5t}$$

A few steps for the 2nd-order RK solution (Heun without iteration) are shown in the following table and a plot of heat flow versus time is shown below.

<i>t</i>	<i>Q</i>	<i>k₁₁</i>	<i>t_{end}</i>	<i>k₂₁</i>	<i>ϕ</i>
0	0	2100	0.1	2094.737	2097.368
0.1	209.7368	2094.737	0.2	2089.447	2092.092
0.2	418.946	2089.447	0.3	2084.131	2086.789
0.3	627.625	2084.131	0.4	2078.788	2081.459
0.4	835.7709	2078.788	0.5	2073.418	2076.103
0.5	1043.381	2073.418	0.6	2068.02	2070.719



28.35 The differential equations to be solved are

nonlinear:

$$\frac{dv}{dt} = 9.8 - \frac{0.225}{68.1} v^2$$

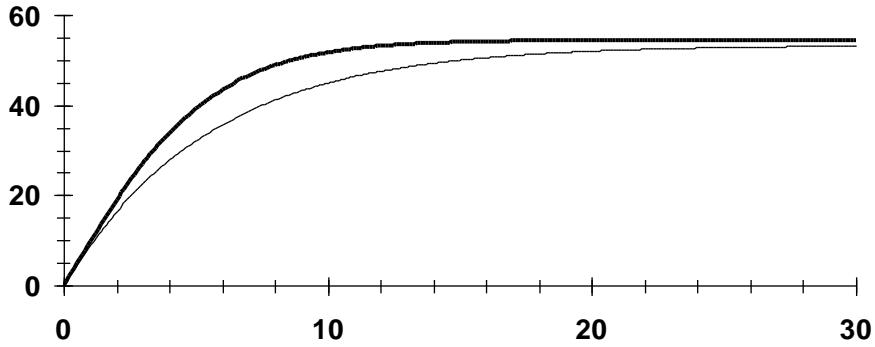
linear:

$$\frac{dv}{dt} = 9.8 - \frac{12.5}{68.1} v$$

A few steps for the solution (Euler) are shown in the following table, which also includes the analytical solution from Example 1.1. A plot of the result is also shown below. Note, the nonlinear solution is the bolder line.

<i>t</i>	<i>linear</i> <i>dv/dt</i>		<i>nonlinear</i> <i>dv/dt</i>		<i>analytical</i> <i>v</i>

0	0	9.8	0	9.8	0
0.1	0.98	9.620117	0.98	9.796827	0.971061
0.2	1.942012	9.443537	1.959683	9.787312	1.92446
0.3	2.886365	9.270197	2.938414	9.771473	2.860518
0.4	3.813385	9.100039	3.915561	9.749345	3.779552
0.5	4.723389	8.933005	4.890496	9.720979	4.681871



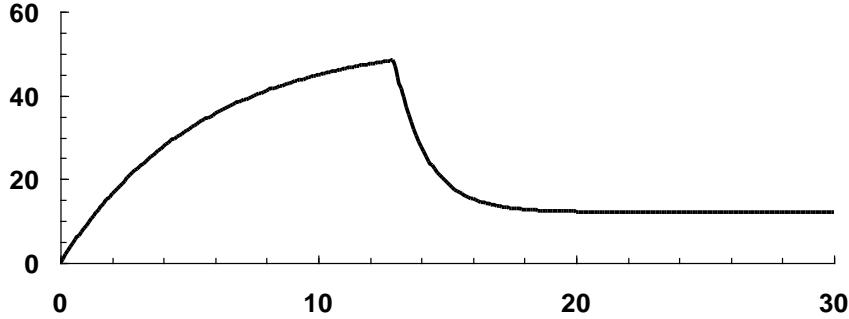
28.36 The differential equations to be solved are

$$t < 13 \text{ s:} \quad \frac{dv}{dt} = 9.8 - \frac{12.5}{68.1} v$$

$$t \geq 13 \text{ s:} \quad \frac{dv}{dt} = 9.8 - \frac{55}{68.1} v$$

The first few steps for the solution (Heun) are shown in the following table, along with the steps when the parachute opens. A plot of the result is also shown below.

<i>t</i>	<i>v</i>	<i>k</i> ₁₁	<i>t</i> _{end}	<i>v</i> _{end}	<i>k</i> ₂₁	<i>ϕ</i>
0	0	9.8	0.1	0.98	9.620117	9.710059
0.1	0.971006	9.621768	0.2	1.933183	9.445157	9.533463
0.2	1.924352	9.446778	0.3	2.86903	9.273379	9.360079
0.3	2.86036	9.274971	0.4	3.787857	9.104725	9.189848
0.4	3.779345	9.106288	0.5	4.689974	8.939138	9.022713
0.5	4.681616	8.940673	0.6	5.575683	8.776563	8.858618
•						
•						
•						
12.8	48.2953	0.935223	12.9	48.38883	0.918057	0.92664
12.9	48.38797	0.918215	13.0	48.47979	-29.354	-14.2179
13	46.96618	-28.1316	13.1	44.15302	-25.8596	-26.9956
13.1	44.26662	-25.9513	13.2	41.67149	-23.8554	-24.9033
13.2	41.77629	-23.94	13.3	39.38228	-22.0065	-22.9733
13.3	39.47896	-22.0846	13.4	37.2705	-20.301	-21.1928



28.37 This problem can be solved with MATLAB:

```
%Damped spring mass system
%mass: m=1 kg
%damping, nonlinear: a abs(dx/dt) (dx/dt), a=2 N/(m/s)^2
%spring, nonlinear: bx^3, b=5 N/m^3
% MATLAB 5 version

%Independent Variable t, tspan=[tstart tstop]
%initial conditions [x(1)=velocity, x(2)=displacement];

t0=0;
tf=8;
tspan=[0 8]; ic=[1 0.5];

% a) linear solution
[t,x]=ode45('kl',tspan,ic);
subplot(221)
plot(t,x); grid; xlabel('time - sec.');
ylabel('displacement - m; velocity - m/s');
title('d2x/dt2+2(dx/dt)+5x=0')
subplot(222)
%Phase-plane portrait
plot(x(:,2),x(:,1)); grid;
xlabel('displacement - m'); ylabel('velocity - m/s');
title('d2x/dt2+2(dx/dt)+5x=0');

% b) nonlinear spring
[t,x]=ode45('nlk',tspan,ic);
subplot(223)
plot(t,x); grid;
xlabel('time - sec.'); ylabel('displacement - m; velocity - m/s');
title('d2x/dt2+2(dx/dt)+5x^3=0')
%Phase-plane portrait
subplot(224)
plot(x(:,2),x(:,1)); grid;
xlabel('displacement - m'); ylabel('velocity - m/s');
title('d2x/dt2+2(dx/dt)+5x=0');
pause

% c) nonlinear damping
[t,x]=ode45('nlc',tspan,ic);
subplot(221)
plot(t,x); grid;
xlabel('time - sec.'); ylabel('displacement - m; velocity - m/s');
title('d2x/dt2+2abs(dx/dt)(dx/dt)+5x=0')
%Phase-plane portrait
```

```

subplot(222)
plot(x(:,2),x(:,1)); grid;
xlabel('displacement - m'); ylabel('velocity - m/s');
title('d2x/dt2+2abs(dx/dt)(dx/dt)+5x=0');

% d) nonlinear damping and spring
[t,x]=ode45('nlck',tspan,ic);
subplot(223)
plot(t,x); grid;
xlabel('time - sec.');
```

ylabel('displacement - m; velocity - m/s');

```
title('d2x/dt2+2abs(dx/dt)(dx/dt)+5x^3=0')
%Phase-plane portrait
subplot(224)
plot(x(:,2),x(:,1)); grid;
xlabel('displacement - m'); ylabel('velocity - m/s');
title('d2x/dt2+2abs(dx/dt)(dx/dt)+5x^3=0');
```

Functions:

```
%Damped spring mass system - m d2x/dt2 + c dx/dt + k x =0
%mass:      m=1 kg
%    linear-   c=2 N/(m/s)
%    linear-   k=5 N/m
%x(1)=velocity, x(2)=displacement

function dx=kl(t,x);
dx=[-2*x(1)-5*x(2); x(1)];

%Damped spring mass system - m d2x/dt2 + c dx/dt + k x =0
%mass:      m=1 kg
%damping:  linear-   c=2 N/(m/s)
%spring:   nonlinear- kx=bx^3,     b=5 N/m^3

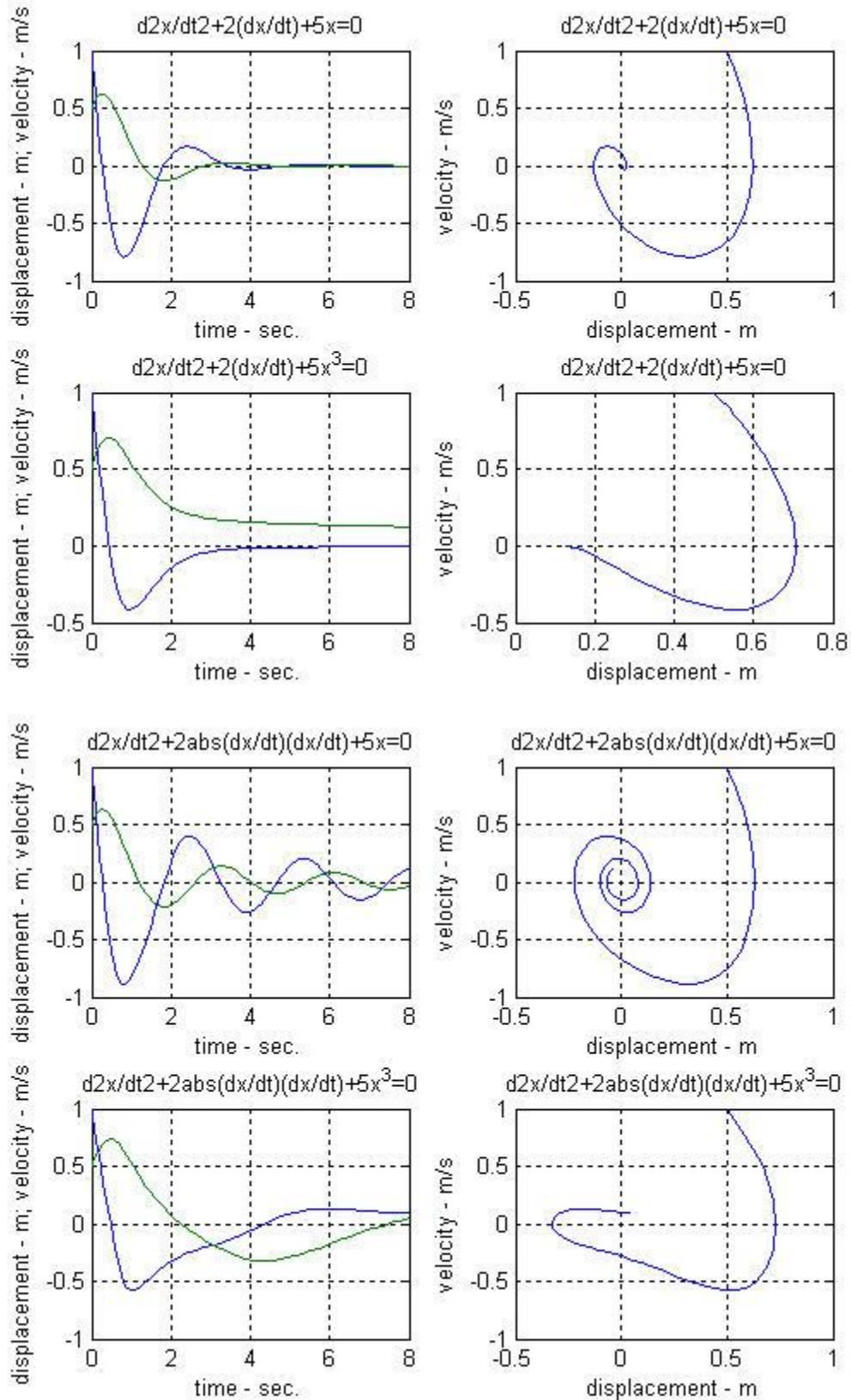
function dx=nlk(t,x);
dx=[-2*x(1)-5*x(2).*x(2).*x(2); x(1)];

%Damped spring mass system - m d2x/dt2 + c dx/dt + k x =0
%mass:      m=1 kg
%damping:  nonlinear- c dx/dt = a sgn(dx/dt) (dx/dt)^2,  a=2 N/(m/s)^2
%spring:   linear-   kx=5x
%x(1)=velocity, x(2)=dispacement

function dx=nlc(t,x);
dx=[-2*abs(x(1))*x(1)-5*x(2); x(1)];

%Damped spring mass system - m d2x/dt2 + c dx/dt + k x =0
%mass:      m=1 kg
%damping:  nonlinear- c dx/dt = a sgn(dx/dt) (dx/dt)^2,  a=2 N/(m/s)^2
%spring:   nonlinear- k x = bx^3,     b=5 N/m^3
%x(1)=velocity, x(2)=dispacement

function dx=nlck(t,x);
dx=[-2*abs(x(1)).*x(1)-5*x(2).*x(2).*x(2); x(1)];
```



28.38 This problem can be solved with MATLAB:

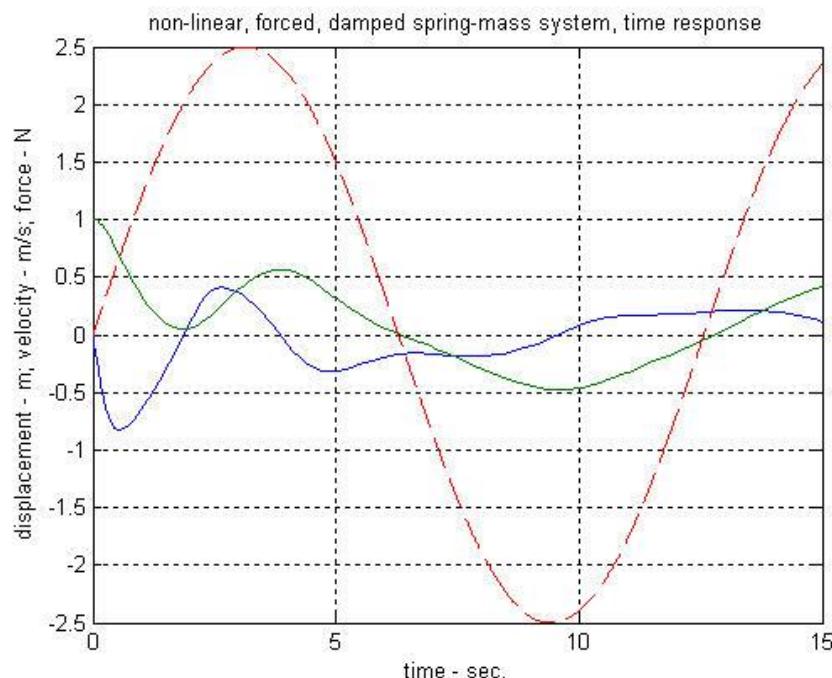
```
%Forced damped spring-mass system w/ material damping
%mass: m=2 kg
%damping, nonlinear air: a abs(dx/dt) dx/dt, a=5 N/(m/s)^2
%spring, linear: kx = 6x
%forcing function: F=Fo(sin(wt)), Fo=2.5 N, w=0.5 rad/s
%Independent Variable t, tspan=[tstart tstop]
%initial conditions [x(1)=velocity, x(2)=displacement];

tspan=[0 15]; ic=[0 1];
[t,x]=ode45('nlF',tspan,ic);
ts=0:.01:15;
Sin=2.5*sin(0.5*ts);
plot(t,x,ts,Sin,'--'); grid; xlabel('time - sec.');
ylabel('displacement - m; velocity - m/s; force - N');
title('non-linear, forced, damped spring-mass system, time response')
```

Function nlF:

```
%Forced damped spring-mass system w/ material damping
%mass: m=2 kg
%damping, nonlinear air: a abs(dx/dt) (dx/dt), a=5 N/(m/s)^2
%spring, linear: kx = 6x
%forcing function: F=Fo(sin(wt)), Fo=2.5 N, w=0.5 rad/s
% x(1)= velocity, x(2)= displacement

function dx=nlF(t,x);
dx=[-2.5*abs(x(1)).*x(1)-3*x(2)+1.25*sin(0.5*t); x(1)];
```



28.39

Errata: In the book's first printing, the differential equation was erroneously printed as

$$\frac{d^2u}{dx^2} + \left(\frac{2}{x}\right) \left(\frac{du}{dx} pu \right) = 0$$

The correct equation, which appears in later printings, is

$$\frac{d^2u}{dx^2} + \left(\frac{2}{x}\right) \left(\frac{du}{dx} - pu \right) = 0$$

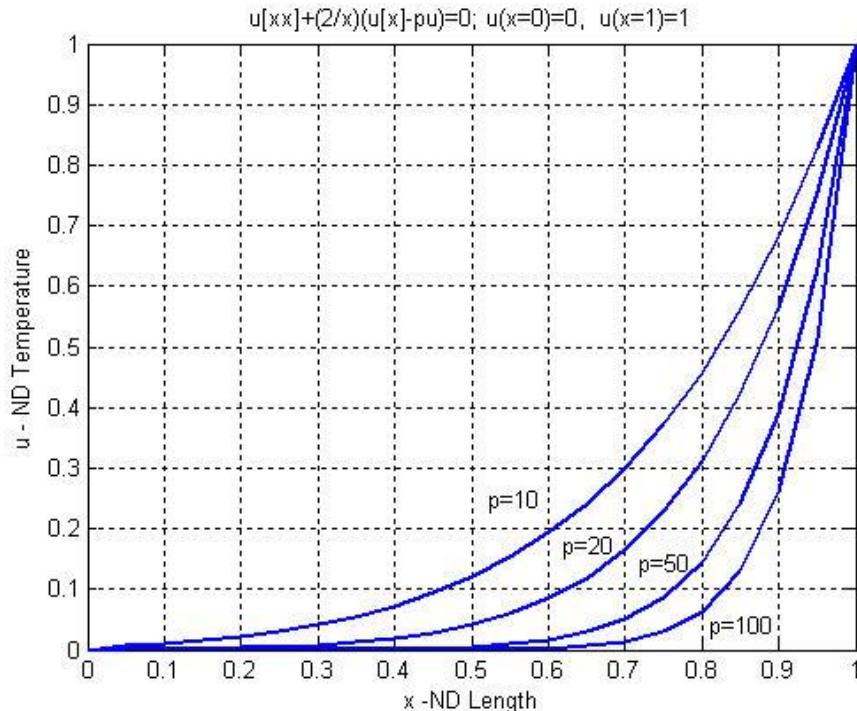
This problem can be solved with MATLAB:

```
% ODE Boundary Value Problem
% Tapered conical cooling fin
% u[xx]+(2/x)(u[x]-pu)=0
% BC. u(x=0)=0 u(x=1)=1
% i=spatial index, from 1 to R
% numbering for points is i=1 to i=R for (R-1) dx spaces
% u(i=1)=0 and u(i=R)=1

R=21;
%Constants
dx=1/(R-1);
dx2=dx*dx;
%Parameters
p(1)=10; p(2)=20; p(3)=50; p(4)=100;
%sizing matrices
u=zeros(1,R); x=zeros(1,R);
a=zeros(1,R); b=zeros(1,R); c=zeros(1,R); d=zeros(1,R);
ba=zeros(1,R); ga=zeros(1,R);
%Independent Variable
x=0:dx:1;
%Boundary Conditions
u(1)=0; u(R)=1;

for k=1:4;
    %Coefficients
    b(2)=-2-2*p(k)*dx2/dx;
    c(2)=2;
    for i=3:R-2,
        a(i)=1-dx/(dx*(i-1));
        b(i)=-2-2*p(k)*dx2/(dx*(i-1));
        c(i)=1+1/(i-1);
    end
    a(R-1)=1-dx/(dx*(R-2));
    b(R-1)=-2-2*p(k)*dx2/(dx*(R-2));
    d(R-1)=-(1+1/(R-2));
    %Solution by Thomas Algorithm
    ba(2)=b(2);
    ga(2)=d(2)/b(2);
    for i=3:R-1,
        ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
        ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
    end
end
```

```
%back substitution step
u(R-1)=ga(R-1);
for i=R-2:-1:2,
    u(i)=ga(i)-c(i)*u(i+1)/ba(i);
end
%Plot
plot(x,u)
title('u[xx]+(2/x)(u[x]-pu)=0; u(x=0)=0, u(x=1)=1')
xlabel('x -ND Length')
ylabel('u - ND Temperature')
hold on
end
grid
hold off
gtext('p=10');gtext('p=20');gtext('p=50');gtext('p=100');
```



28.40 The second-order equation can be expressed as a pair of first-order equations,

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -\frac{c}{m}v - \frac{k_1}{m}x - \frac{k_3}{m}x^3 + \frac{P}{m}\cos(\omega t)\end{aligned}$$

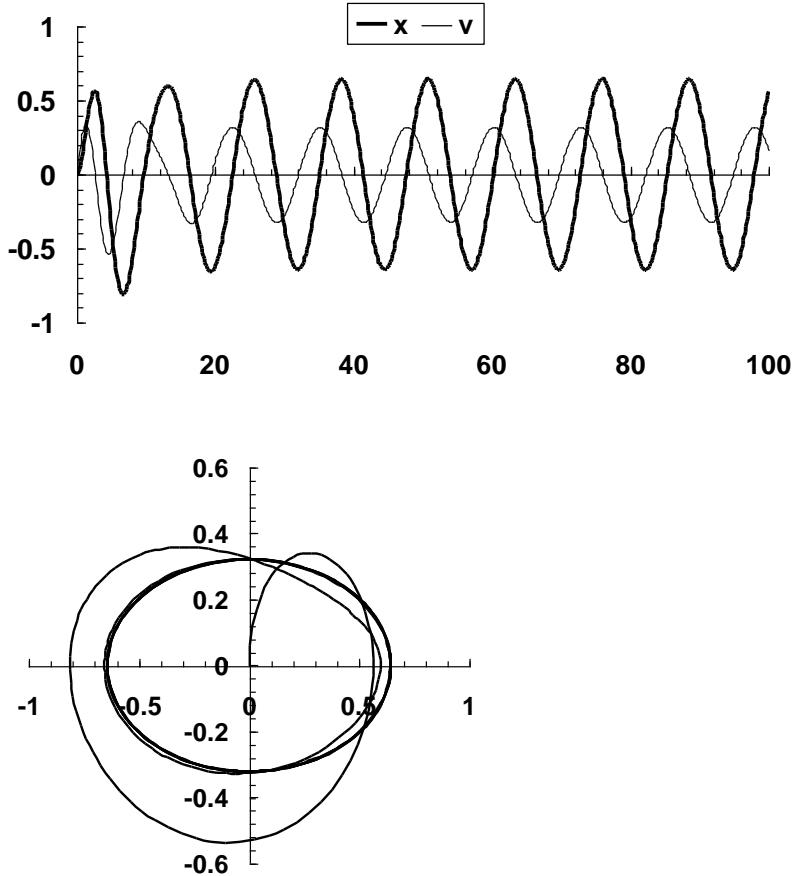
Substituting the parameter values:

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -0.4v - k_1x - k_3x^3 + 0.5\cos(0.5t)\end{aligned}$$

(a) Linear ($k_1 = 1$; $k_3 = 0$):

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -0.4v - x + 0.5 \cos(0.5t)\end{aligned}$$

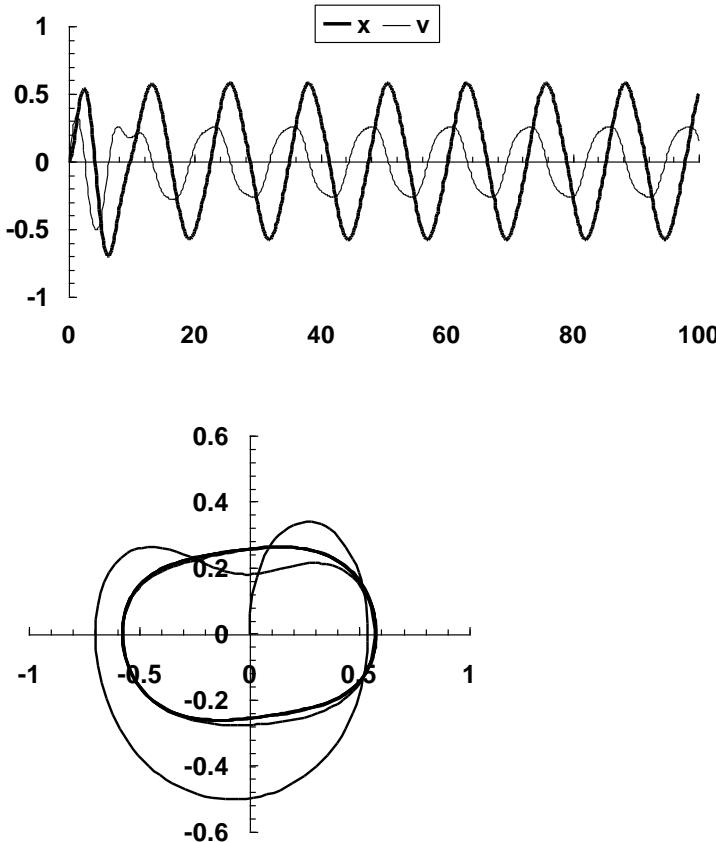
Here are the results of solving these equations with the initial condition, $x = v = 0$, using the Heun method without corrector and a step-size $h = 0.125$.



(b) Nonlinear ($k_1 = 1$; $k_3 = 0.5$):

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -0.4v - x - 0.5x^3 + 0.5 \cos(0.5t)\end{aligned}$$

Here are the results of solving these equations with the initial condition, $x = v = 0$, using the Heun method without corrector and a step-size $h = 0.125$.



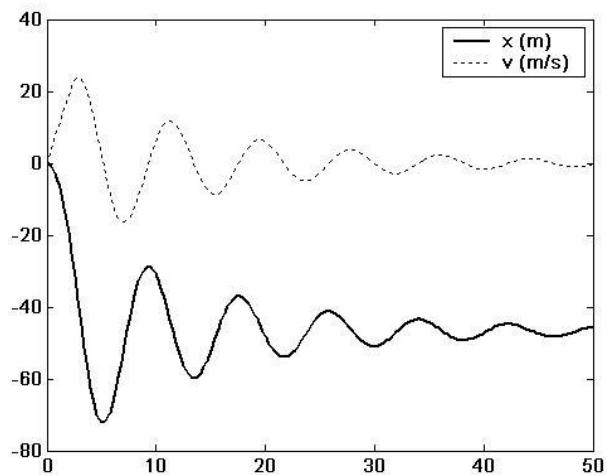
28.41 The following MATLAB M-file can be set up to compute the right-hand-sides of the ODEs,

```
function dydt = bungee(t,y,L,cd,m,k,gamma)
g = 9.81;
cord = 0;
if y(1) > L %determine if the cord exerts a force
    cord = k/m*(y(1)-L)+gamma/m*y(2);
end
dydt = [y(2); g - sign(y(2))*cd/m*y(2)^2 - cord];
```

We can use `ode45` to obtain the solutions and display them on a plot,

```
>> [t,y]=ode45(@bungee,[0 50],[0 0],[],30,0.25,68.1,40,8);
>> plot(t,-y(:,1),'-'',t,y(:,2),':')
>> legend('x (m)', 'v (m/s)')
```

As in the following figure, we have reversed the sign of distance for the plot so that negative distance is in the downward direction.



PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

CHAPTER 29

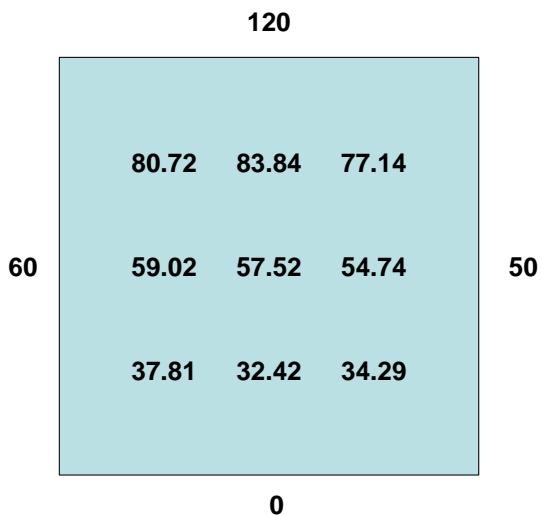
29.1 Here are the results of using Liebmann's method to obtain the solution. Notice that after 6 iterations all the relative error estimates have fallen below 1% and the computation is terminated.

```

iteration = 1
    18          5.4          16.62
    23.4        8.64         22.578
    61.02       56.898       74.8428
ea:
    100         100         100
    100         100         100
    100         100         100
iteration = 2
    23.04       13.41       22.4724
    41.13       38.4768     51.222
    71.2044     79.9776     75.39132
ea:
    21.875      59.73154362  26.04261227
    43.10722101 77.54491018  55.92128382
    14.30304869 28.85758012  0.727563863
•
•
•
iteration = 6
    37.80666066 32.41632148 34.29437766
    59.01952801 57.51727394 54.73884006
    80.71921628 83.84433834 77.14262488
ea:
    0.688181746 0.130204454 0.019913358
    0.010933694 0.020874012 0.049145839
    0.033767672 0.037176168 0.02465349

```

Therefore, the results are:



29.2 The fluxes for Prob. 29.1 can be calculated as in Example 29.2. For example, for $i = j = 1$,

$$q_x = -0.49 \frac{32.41632148 - 60}{2(10)} = 0.675800124$$

$$q_y = -0.49 \frac{59.01952801 - 0}{2(10)} = -1.445978436$$

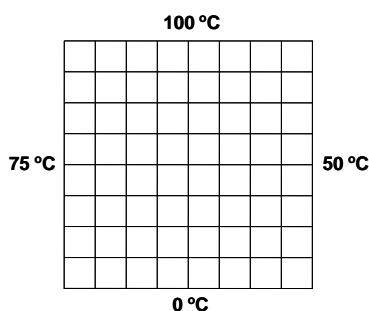
$$q_n = \sqrt{0.675800124^2 + (-1.445978436)^2} = 1.596107592$$

$$\theta = \tan^{-1} \left(\frac{-1.445978436}{0.675800124} \right) \times \frac{180^\circ}{\pi} = -64.95024572^\circ$$

All the results are summarized below:

qx:			
0.675800124	0.086050934	-0.430800124	
0.060826788	0.104876855	0.184173212	
-0.584186289	0.087626489	0.829186289	
qy:			
-1.445978436	-1.409173212	-1.341101582	
-1.051357613	-1.259986413	-1.049782057	
-1.494021564	-1.530826788	-1.598898418	
qn:			
1.596107592	1.41179811	1.408595825	
1.053115724	1.26434367	1.065815246	
1.604173947	1.533332664	1.801118001	
theta:			
-64.95024572	-86.50558167	-107.8085014	
-86.68881693	-85.24186843	-80.04932455	
-111.356292	-86.72389108	-62.588814	

29.3 The plate is redrawn below



After 15 iterations of the Liebmann method, the result is

	100	100	100	100	100	100	100	
75	85.32617	88.19118	88.54443	87.79909	86.06219	82.39736	73.69545	50
75	78.10995	78.88691	78.1834	76.58771	74.05069	69.82967	62.38146	50
75	73.23512	71.06672	68.71675	66.32057	63.72554	60.48986	55.99875	50
75	68.75568	63.42793	59.30269	56.25934	54.04625	52.40787	51.1222	50
75	63.33804	54.57569	48.80562	45.37425	43.79945	43.97646	46.08048	50
75	54.995	42.71618	35.95756	32.62971	31.80514	33.62176	39.22063	50

75	38.86852	25.31308	19.66293	17.3681	17.16645	19.48972	27.17735	50
	0	0	0	0	0	0	0	

with percent approximate errors of

	0	0	0	0	0	0	0	
0	0.012%	0.011%	0.007%	0.005%	0.004%	0.004%	0.003%	0
0	0.011%	0.012%	0.008%	0.005%	0.006%	0.006%	0.005%	0
0	0.024%	0.010%	0.001%	0.001%	0.004%	0.007%	0.006%	0
0	0.054%	0.016%	0.007%	0.011%	0.002%	0.007%	0.008%	0
0	0.101%	0.040%	0.008%	0.007%	0.003%	0.008%	0.011%	0
0	0.234%	0.119%	0.063%	0.033%	0.012%	0.010%	0.015%	0
0	0.712%	0.292%	0.219%	0.126%	0.030%	0.001%	0.014%	0
	0	0	0	0	0	0	0	

29.4 The solution is identical to Prob. 29.3, except that now the bottom edge must be modeled. This means that the nodes along the bottom edge are simulated with equations of the form

$$4T_{i,j} - T_{i-1,j} - T_{i+1,j} - 2T_{i,j+1} = 0$$

The resulting simulation (after 15 iterations) yields

	100	100	100	100	100	100	100	
75	86.4529	90.2627	91.2337	90.6948	88.7323	84.4436	74.8055	50
75	80.5554	83.3649	83.9691	82.8006	79.7785	74.2277	64.7742	50
75	77.4205	78.6771	78.4736	76.7481	73.3375	67.8999	60.0537	50
75	75.5117	75.4794	74.5080	72.3743	68.9073	63.9608	57.5247	50
75	74.2631	73.2996	71.7406	69.3405	65.9436	61.4870	56.0597	50
75	73.4348	71.8433	69.8853	67.3320	64.0357	59.9572	55.1934	50
75	72.8998	70.9218	68.7359	66.1171	62.9167	59.0892	54.7153	50
75	72.5345	70.4401	68.1797	65.5685	62.4467	58.7477	54.5354	50

with percent approximate errors of

	0	0	0	0	0	0	0	
0	0.009%	0.018%	0.024%	0.024%	0.018%	0.011%	0.004%	0
0	0.042%	0.066%	0.074%	0.069%	0.053%	0.034%	0.015%	0
0	0.079%	0.140%	0.155%	0.140%	0.110%	0.071%	0.032%	0
0	0.113%	0.224%	0.260%	0.239%	0.187%	0.124%	0.057%	0
0	0.133%	0.327%	0.388%	0.359%	0.284%	0.190%	0.089%	0
0	0.173%	0.471%	0.544%	0.502%	0.395%	0.261%	0.124%	0
0	0.289%	0.628%	0.709%	0.651%	0.508%	0.330%	0.153%	0
0	0.220%	0.665%	0.779%	0.756%	0.620%	0.407%	0.180%	0

29.5 The solution is identical to Examples 29.1 and 29.3, except that now heat balances must be developed for the three interior nodes on the bottom edge. For example, using the control-volume approach, node 1,0 can be modeled as

$$-0.49(5)\frac{T_{10} - T_{00}}{10} + 0.49(5)\frac{T_{20} - T_{10}}{10} + 0.49(10)\frac{T_{11} - T_{10}}{10} - 1(10) = 0$$

$$4T_{10} - T_{00} - T_{20} - 2T_{11} = -40.8163$$

Using Liebmann's method and iterating to a high level of precision, the results are

	100	100	100	
75	81.6571	80.1712	72.5078	50
75	71.4579	66.5204	59.8602	50
75	62.6549	54.5930	50.4129	50
75	49.5694	38.7843	37.1984	50

The fluxes for the computed nodes can be determined as

q_x :

	-0.1267	0.2242	0.7392	
	0.2077	0.2841	0.4048	
	0.5000	0.2999	0.1125	
	0.8873	0.3031	-0.2748	

q_y :

	-0.6993	-0.8202	-0.9834	
	-0.4656	-0.6267	-0.5413	
	-0.5363	-0.6795	-0.5552	
	-0.6412	-0.7746	-0.6475	

q_n :

	0.7107	0.8503	1.2303	
	0.5098	0.6881	0.6759	
	0.7332	0.7428	0.5665	
	1.0947	0.8318	0.7034	

θ (degrees):

	-100.269	-74.7153	-53.0695	
	-65.9517	-65.6094	-53.2145	
	-47.0062	-66.1845	-78.5428	
	-35.8536	-68.6311	-112.995	

29.6 The solution is identical to Example 29.4, except that now heat balances must be developed for the interior nodes at the lower left and the upper right edges. The balances for nodes 1,1 and 3,3 can be written as

$$-4T_{11} + 0.8453T_{21} + 0.8453T_{12} = -1.154701(T_{01} + T_{10})$$

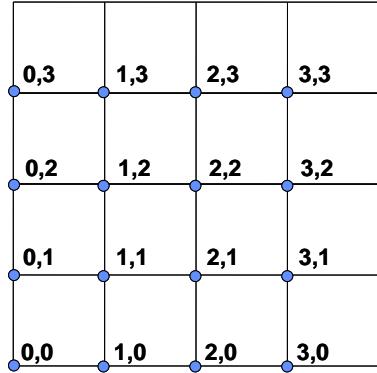
$$-4T_{33} + 0.8453T_{32} + 0.8453T_{23} = -1.154701(T_{34} + T_{43})$$

Using the appropriate boundary conditions, simple Laplacians can be used for the remaining interior nodes. The resulting simulation yields

75	100	100	100	
50	75	86.02317	94.09269	100

50	63.97683	75	86.02317	100
50	55.90731	63.97683	75	100
	50	50	50	75

29.7 The nodes to be simulated are



Simple Laplacians are used for all interior nodes. Balances for the edges must take insulation into account. For example, node 1,0 is modeled as

$$4T_{1,0} - T_{0,0} - T_{2,0} - 2T_{1,1} = 0$$

The corner node, 0,0 would be modeled as

$$4T_{0,0} - 2T_{1,0} - 2T_{0,1} = 0$$

The resulting set of equations can be solved for

0	25	50	75	100
23.89706	32.16912	45.58824	60.29412	75
31.25	34.19118	39.88971	45.58824	50
32.72059	33.45588	34.19118	32.16912	25
32.72059	32.72059	31.25	23.89706	0

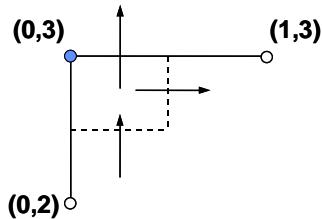
The fluxes can be computed as

J_x	-1.225	-1.225	-1.225	-1.225	-1.225
	-0.40533	-0.53143	-0.68906	-0.72059	-0.72059
	-0.14412	-0.21167	-0.27923	-0.2477	-0.21618
	-0.03603	-0.03603	0.031526	0.225184	0.351287
	0	0.036029	0.216176	0.765625	1.170956
J_y	1.170956	0.351287	-0.21618	-0.72059	-1.225
	0.765625	0.225184	-0.2477	-0.72059	-1.225
	0.216176	0.031526	-0.27923	-0.68906	-1.225
	0.036029	-0.03603	-0.21167	-0.53143	-1.225
	0	-0.03603	-0.14412	-0.40533	-1.225
J_n	1.694628	1.274373	1.243928	1.421222	1.732412
	0.866299	0.577174	0.732232	1.019066	1.421222
	0.259812	0.214008	0.394888	0.732232	1.243928

0.050953	0.050953	0.214008	0.577174	1.274373
0	0.050953	0.259812	0.866299	1.694628
θ (degrees)				
136.2922	163.999	-169.992	-149.534	-135
117.8973	157.0362	-160.228	-135	-120.466
123.6901	171.5289	-135	-109.772	-100.008
135	-135	-81.5289	-67.0362	-73.999
0	-45	-33.6901	-27.8973	-46.2922

29.8 Node 0,3:

There are two approaches for modeling this node. One would be to consider it a Dirichlet node and not model it at all (i.e., set its temperature at 50°C). The second alternative is to use a heat balance to model it as shown here



$$0 = 0.75(15)(0.5) \frac{T_{1,3} - T_{0,3}}{40} - 0.75(20)(0.5) \frac{T_{0,3} - T_{0,2}}{30} + 0.015(20)(0.5)(10 - T_{0,3})$$

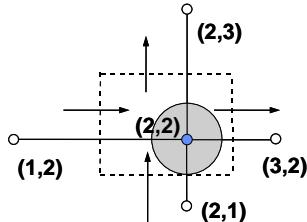
$$-1.04046T_{1,3} + 4T_{0,3} - 1.84971T_{0,2} = 11.09827$$

Node 2,3:

$$0 = -0.75(15)(0.5) \frac{T_{2,3} - T_{1,3}}{40} + 0.75(15)(0.5) \frac{T_{3,3} - T_{2,3}}{20} - 0.75(30)(0.5) \frac{T_{2,3} - T_{2,2}}{30} + 0.015(30)(0.5)(10 - T_{2,3})$$

$$4T_{2,3} - 0.55046T_{1,3} - 1.10092T_{3,3} - 1.46789T_{2,2} = 8.807339$$

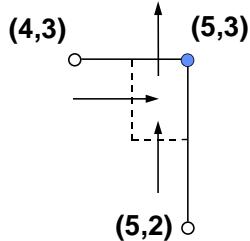
Node 2,2:



$$0 = -0.75(22.5)(0.5) \frac{T_{2,2} - T_{1,2}}{40} + 0.75(22.5)(0.5) \frac{T_{3,2} - T_{2,2}}{20} - 0.75(30)(0.5) \frac{T_{2,2} - T_{2,1}}{30} + 0.75(30)(0.5) \frac{T_{2,3} - T_{2,2}}{30} + 10\pi(7.5)^2$$

$$4T_{2,2} - 0.48T_{1,2} - 0.96T_{3,2} - 1.70667T_{2,1} - 0.85333T_{2,3} = 4021.239$$

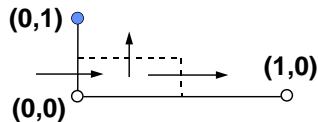
Node 5,3:



$$0 = -0.75(15)(0.5) \frac{T_{5,3} - T_{4,3}}{20} - 0.75(10)(0.5) \frac{T_{5,3} - T_{5,2}}{30} + 0.015(10)(0.5)(10 - T_{5,3})$$

$$4T_{5,3} - 2.33766T_{4,3} - 1.03896T_{5,2} = 6.23377$$

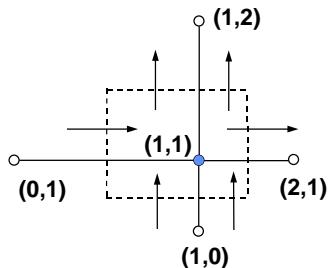
29.9 Node 0,0:



$$0 = 0.015(7.5)(1.5)(20 - T_{0,0}) + 0.7(7.5)(1.5) \frac{T_{1,0} - T_{0,0}}{40} + 0.7(20)(1.5) \frac{T_{0,1} - T_{0,0}}{15}$$

$$4T_{0,0} - 0.44602T_{1,0} - 3.17168T_{0,1} = 7.646018$$

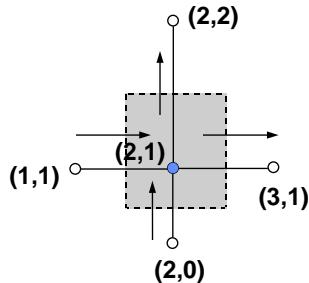
Node 1,1:



$$\begin{aligned} 0 = & -0.7(22.5)(1.5) \frac{T_{1,1} - T_{0,1}}{40} + 0.5(22.5)(1.5) \frac{T_{2,1} - T_{1,1}}{20} - 0.7(20)(1.5) \frac{T_{1,1} - T_{1,0}}{15} \\ & - 0.5(10)(1.5) \frac{T_{1,1} - T_{1,0}}{15} + 0.7(20)(1.5) \frac{T_{1,2} - T_{1,1}}{30} + 0.5(10)(1.5) \frac{T_{1,2} - T_{1,1}}{30} \end{aligned}$$

$$4T_{1,1} - 0.78775T_{2,1} - 1.77389T_{1,0} - 0.88694T_{1,2} - 0.55142T_{0,1} = 0$$

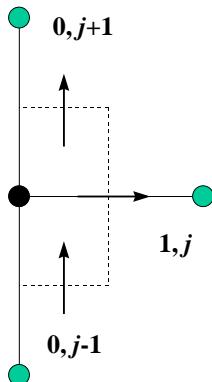
Node 2,1:



$$0 = -0.5(22.5)(1.5) \frac{T_{2,1} - T_{1,1}}{20} + 0.5(22.5)(1.5) \frac{T_{3,1} - T_{2,1}}{20} - 0.5(20)(1.5) \frac{T_{2,1} - T_{2,0}}{15} \\ + 0.5(20)(1.5) \frac{T_{2,2} - T_{2,1}}{30} + 10(22.5)(20)$$

$$4T_{2,1} - 1.05882T_{1,1} - 1.05882T_{3,1} - 1.2549T_{2,0} - 0.62745T_{2,2} = 5,647.059$$

29.10 The control volume is drawn as in



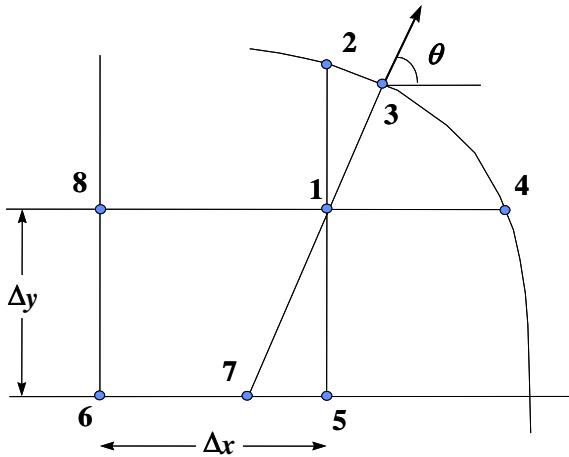
A flux balance around the node can be written as (note $\Delta x = \Delta y = h$)

$$-kh\Delta z \frac{T_{1,j} - T_{0,j}}{h} + k(h/2)\Delta z \frac{T_{0,j} - T_{0,j-1}}{h} - k(h/2)\Delta z \frac{T_{0,j+1} - T_{0,j}}{h} = 0$$

Collecting and canceling terms gives

$$4T_{0,j} - T_{0,j-1} - T_{0,j+1} - 2T_{1,j} = 0$$

29.11 A setup similar to Fig. 29.11, but with $\theta > 45^\circ$ can be drawn as in



The normal derivative at node 3 can be approximated by the gradient between nodes 1 and 7,

$$\left. \frac{\partial T}{\partial \eta} \right|_3 = \frac{T_1 - T_7}{L_{17}}$$

When θ is greater than 45° as shown, the distance from node 5 to 7 is $\Delta y \cot \theta$, and linear interpolation can be used to estimate

$$T_7 = T_5 + (T_6 - T_5) \frac{\Delta y \cot \theta}{\Delta x}$$

The length L_{17} is equal to $\Delta y / \sin \theta$. This length, along with the approximation for T_7 can be substituted into the gradient equation to give

$$T_1 = \left(\frac{\Delta y}{\sin \theta} \right) \left. \frac{\partial T}{\partial \eta} \right|_3 + T_6 \frac{\Delta y \cot \theta}{\Delta x} + T_5 \left(1 - \frac{\Delta y \cot \theta}{\Delta x} \right)$$

29.12 The following VBA program implements Liebmann's method with relaxation.

```

Option Explicit

Sub Liebmann()
    Dim nx As Integer, ny As Integer, l As Integer
    Dim i As Integer, j As Integer
    Dim T(20, 20) As Double, ea(20, 20) As Double, Told(20, 20) As Double
    Dim qy(20, 20) As Double, qx(20, 20) As Double, qn(20, 20) As Double
    Dim th(20, 20) As Double
    Dim Trit As Double, Tlef As Double, Ttop As Double, Tbot As Double
    Dim lam As Double, emax As Double, es As Double
    Dim pi As Double
    Dim k As Double, x As Double, y As Double, dx As Double, dy As Double
    nx = 4
    ny = 4
    pi = 4 * Atan(1)
    x = 40
    y = 40

```

```

k = 0.49
lam = 1.5
es = 1
dx = x / nx
dy = y / ny
Tbot = 0
Tlef = 75
Trit = 50
Ttop = 100
For i = 1 To nx - 1
    T(i, 0) = Tbot
Next i
For i = 1 To nx - 1
    T(i, ny) = Ttop
Next i
For j = 1 To ny - 1
    T(0, j) = Tlef
Next j
For j = 1 To ny - 1
    T(nx, j) = Trit
Next j
l = 0
Sheets("sheet1").Select
Range("a5:z5000").ClearContents
Range("a5").Select
Do
    l = l + 1
    emax = 0
    For j = 1 To ny - 1
        For i = 1 To nx - 1
            Told(i, j) = T(i, j)
            T(i, j) = (T(i + 1, j) + T(i - 1, j) + T(i, j + 1) + T(i, j - 1)) / 4
            T(i, j) = lam * T(i, j) + (1 - lam) * Told(i, j)
            ea(i, j) = Abs((T(i, j) - Told(i, j)) / T(i, j)) * 100
            If (ea(i, j) > emax) Then emax = ea(i, j)
        Next i
    Next j
    ActiveCell.Value = "iteration = "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = l
    ActiveCell.Offset(1, -1).Select
    For j = 1 To ny - 1
        For i = 1 To nx - 1
            ActiveCell.Value = T(i, j)
            ActiveCell.Offset(0, 1).Select
        Next i
        ActiveCell.Offset(1, -(nx - 1)).Select
    Next j
    ActiveCell.Value = "ea = "
    ActiveCell.Offset(1, 0).Select
    For j = 1 To ny - 1
        For i = 1 To nx - 1
            ActiveCell.Value = ea(i, j)
            ActiveCell.Offset(0, 1).Select
        Next i
        ActiveCell.Offset(1, -(nx - 1)).Select
    Next j
    If emax <= es Then Exit Do
Loop
For j = 1 To ny - 1
    For i = 1 To nx - 1
        qy(i, j) = -k * (T(i, j + 1) - T(i, j - 1)) / 2 / dy

```

```

    qx(i, j) = -k * (T(i + 1, j) - T(i - 1, j)) / 2 / dx
    qn(i, j) = Sqr(qy(i, j) ^ 2 + qx(i, j) ^ 2)
    th(i, j) = Application.WorksheetFunction.Atan2(qx(i, j), qy(i, j)) * 180/pi
    Next i
Next j
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = "qx = "
ActiveCell.Offset(1, 0).Select
For j = 1 To ny - 1
    For i = 1 To nx - 1
        ActiveCell.Value = qx(i, j)
        ActiveCell.Offset(0, 1).Select
    Next i
    ActiveCell.Offset(1, -(nx - 1)).Select
Next j
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = "qy = "
ActiveCell.Offset(1, 0).Select
For j = 1 To ny - 1
    For i = 1 To nx - 1
        ActiveCell.Value = qy(i, j)
        ActiveCell.Offset(0, 1).Select
    Next i
    ActiveCell.Offset(1, -(nx - 1)).Select
Next j
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = "qn = "
ActiveCell.Offset(1, 0).Select
For j = 1 To ny - 1
    For i = 1 To nx - 1
        ActiveCell.Value = qn(i, j)
        ActiveCell.Offset(0, 1).Select
    Next i
    ActiveCell.Offset(1, -(nx - 1)).Select
Next j
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = "theta = "
ActiveCell.Offset(1, 0).Select
For j = 1 To ny - 1
    For i = 1 To nx - 1
        ActiveCell.Value = th(i, j)
        ActiveCell.Offset(0, 1).Select
    Next i
    ActiveCell.Offset(1, -(nx - 1)).Select
Next j
End Sub

```

When the program is run, the result of the last iteration is shown below. Note that for simplicity in programming, the sense of the y dimension is reversed from Fig. 29.3. That is, node 1,1 is at the upper left rather than the lower left corner of the domain.

iteration =	9	
43.0006	33.29754	33.88506
63.21151	56.11237	52.33998
78.58717	76.06401	69.71051
ea =		
0.711643	0.342925	0.247647
0.045667	0.464174	0.027927
0.194747	0.17208	0.47127

```

qx =
    1.02171    0.223331   -0.40921
    0.462747   0.266352   0.149753
   -0.02607   0.217478   0.638568

qy =
   -1.54868   -1.37475   -1.28233
   -0.87187   -1.04778   -0.87772
   -0.90132   -1.07525   -1.16767

qn =
    1.855346   1.392775   1.346039
    0.987063   1.081103   0.890407
    0.901695   1.09702    1.330873

theta =
   -56.586    -80.7728   -107.699
   -62.0428   -75.7371   -80.3177
   -91.6567   -78.5657   -61.3269

```

29.13 When the program is run, the result of the last iteration is:

```

iteration =          6
    37.80666   32.41632   34.29438
    59.01953   57.51727   54.73884
    80.71922   83.84434   77.14262

ea =
    0.688182   0.130204   0.019913
    0.010934   0.020874   0.049146
    0.033768   0.037176   0.024653

qx =
    0.6758     0.086051   -0.4308
    0.060827   0.104877   0.184173
   -0.58419    0.087626   0.829186

qy =
   -1.44598   -1.40917   -1.3411
   -1.05136   -1.25999   -1.04978
   -1.49402   -1.53083   -1.5989

qn =
    1.596108   1.411798   1.408596
    1.053116   1.264344   1.065815
    1.604174   1.533333   1.801118

theta =
   -64.9502   -86.5056   -107.809
   -86.6888   -85.2419   -80.0493
   -111.356   -86.7239   -62.5888

```

29.14 When the program is run, the result of the last iteration is:

```

iteration =          15
    38.86852   25.31308   19.66293   17.3681   17.16645   19.48972   27.17735
    54.995     42.71618   35.95756   32.62971   31.80514   33.62176   39.22063
    63.33804   54.57569   48.80562   45.37425   43.79945   43.97646   46.08048
    68.75568   63.42793   59.30269   56.25934   54.04625   52.40787   51.1222
    73.23512   71.06672   68.71675   66.32057   63.72554   60.48986   55.99875

```

78.10995	78.88691	78.1834	76.58771	74.05069	69.82967	62.38146
85.32617	88.19118	88.54443	87.79909	86.06219	82.39736	73.69545
ea =						
0.711857	0.292484	0.219373	0.125895	0.030211	0.001306	0.013573
0.234333	0.119163	0.06299	0.032735	0.01195	0.009868	0.014938
0.100506	0.040025	0.007652	0.007407	0.003258	0.007935	0.010582
0.054168	0.016381	0.006687	0.010552	0.002001	0.006646	0.007611
0.024103	0.009696	0.000684	0.000661	0.003571	0.006687	0.006189
0.010882	0.012035	0.007912	0.00547	0.005693	0.006169	0.005055
0.011993	0.011445	0.007054	0.004884	0.004262	0.003984	0.003025
qx =						
2.434659	0.941074	0.389304	0.122327	-0.10396	-0.49053	-1.495
1.581907	0.932835	0.494237	0.203469	-0.04861	-0.36336	-0.80253
1.000791	0.712088	0.450871	0.245303	0.068492	-0.11177	-0.29515
0.567032	0.463196	0.351261	0.257566	0.188722	0.143278	0.117986
0.192731	0.2214	0.232561	0.24457	0.285705	0.378613	0.514003
-0.19046	-0.0036	0.112661	0.202503	0.331144	0.571792	0.971654
-0.64637	-0.15769	0.019212	0.12163	0.264685	0.60597	1.587471
qy =						
-2.69476	-2.09309	-1.76192	-1.59886	-1.55845	-1.64747	-1.92181
-1.19901	-1.43387	-1.42799	-1.3723	-1.30502	-1.19985	-0.92625
-0.67427	-1.01488	-1.14391	-1.15785	-1.08981	-0.92052	-0.58318
-0.48496	-0.80806	-0.97565	-1.02637	-0.97638	-0.80916	-0.486
-0.45836	-0.75749	-0.92515	-0.99609	-0.98022	-0.85367	-0.5517
-0.59246	-0.8391	-0.97156	-1.05245	-1.0945	-1.07347	-0.86714
-1.07261	-1.03454	-1.06901	-1.1472	-1.27152	-1.47835	-1.84331
qn =						
3.631704	2.29492	1.804417	1.603529	1.561915	1.718944	2.434829
1.984955	1.710602	1.511103	1.387304	1.305922	1.253663	1.225563
1.206742	1.239775	1.22956	1.183552	1.091965	0.92728	0.653614
0.746129	0.931403	1.036951	1.058194	0.99445	0.821744	0.500112
0.497231	0.789183	0.953937	1.025675	1.021006	0.933861	0.75404
0.622322	0.839106	0.978066	1.071752	1.143494	1.216256	1.302321
1.252313	1.046491	1.069186	1.153632	1.298773	1.597719	2.432663
theta =						
-47.9028	-65.7909	-77.5404	-85.6249	-93.8164	-106.581	-127.88
-37.1603	-56.9531	-70.9089	-81.5663	-92.1332	-106.848	-130.907
-33.9697	-54.9446	-68.4882	-78.0382	-86.4039	-96.923	-116.845
-40.5389	-60.1778	-70.1997	-75.9126	-79.0603	-79.9587	-76.3542
-67.1942	-73.7074	-75.8896	-76.2051	-73.7501	-66.0821	-47.026
-107.821	-90.2458	-83.3856	-79.1087	-73.1666	-61.9576	-41.7469
-121.074	-98.6669	-88.9704	-83.9479	-78.241	-67.7114	-49.2647

29.15

$$\Sigma q = 0$$

$$q_{left} - q_{right} + q_{lower-A} + q_{lower-B} - q_{upper-A} - q_{upper-B} + q_{source} = 0$$

$$k_A h \Delta z \frac{T_{21} - T_{22}}{h} - k_B h \Delta z \frac{T_{22} - T_{23}}{h/2} + k_A \frac{h}{2} \Delta z \frac{T_{12} - T_{22}}{h} + k_B \frac{h}{4} \Delta z \frac{T_{12} - T_{22}}{h}$$

$$- k_A \frac{h}{2} \Delta z \frac{T_{22} - T_{32}}{h} - k_B \frac{h}{4} \Delta z \frac{T_{22} - T_{32}}{h} + S(h) \left(\frac{3}{4} h \cdot h \right) \Delta z = 0$$

$$0.25(10)(0.25) \frac{T_{21} - T_{22}}{10} - 0.45(10)(0.25) \frac{T_{22} - T_{23}}{5} + 0.25(5)(0.25) \frac{T_{12} - T_{22}}{10}$$

$$+ 0.45(2.5)(0.25) \frac{T_{12} - T_{22}}{10} - 0.25(5)(0.25) \frac{T_{22} - T_{32}}{10}$$

$$- 0.45(2.5)(0.25) \frac{T_{22} - T_{32}}{10} + 6(10)(7.5)(0.25) = 0$$

$$0.0625(T_{21} - T_{22}) - 0.225(T_{22} - T_{23}) + 0.03125(T_{12} - T_{22}) + 0.028125(T_{12} - T_{22})$$

$$- 0.03125(T_{22} - T_{32}) - 0.028125(T_{22} - T_{32}) + 112.5 = 0$$

$$0.0625T_{21} - (0.0625 + 0.225 + 0.03125 + 0.028125 + 0.03125 + 0.028125)T_{22} + 0.225T_{23}$$

$$+ (0.03125 + 0.028125)T_{12} + (0.03125 + 0.028125)T_{32} = -112.5$$

$$- 0.40625T_{22} + 0.0625T_{21} + 0.225T_{23} + 0.059375T_{12} + 0.059375T_{32} = -112.5$$

This equation can be multiplied by $-4/0.40625$ in order that the coefficient of the T_{22} term is 4. The result is

$$4T_{22} - 0.61538T_{21} - 2.21538T_{23} - 0.58462T_{12} - 0.58462T_{32} = 1107.692$$

29.16

Horizontal flux:

$$q_{xA} = -0.25 \frac{51.6 - 74.2}{10} = 0.565 \quad \text{from } A \text{ into } B$$

$$q_{xB} = -0.45 \frac{45.3 - 51.6}{5} = 0.567 \quad \text{from } A \text{ into } B$$

The horizontal flux at the boundary should be equal.

Vertical flux:

$$q_{yA} = -0.25 \frac{38.6 - 87.4}{2(10)} = 0.610 \quad \text{upward}$$

$$q_{yB} = -0.45 \frac{38.6 - 87.4}{2(10)} = 1.098 \quad \text{upward}$$

The 2 vertical fluxes are unequal.

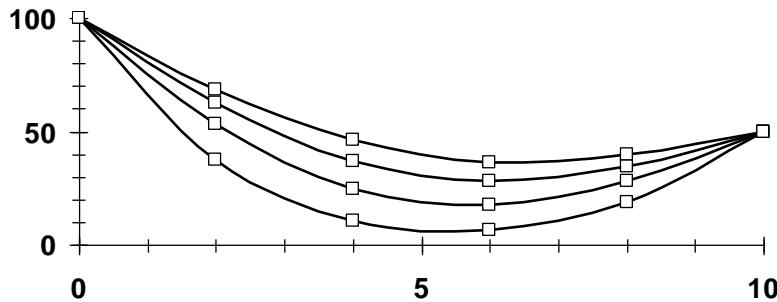
CHAPTER 30

30.1 The key to approaching this problem is to recast the PDE as a system of ODEs. Thus, by substituting the finite-difference approximation for the spatial derivative, we arrive at the following general equation for each node

$$\frac{dT_i}{dt} = k \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

By writing this equation for each node, the solution reduces to solving 4 simultaneous ODEs with Heun's method. The results for the first two steps along with some later selected values are tabulated below. In addition, a plot similar to Fig. 30.4, is also shown

t	$x = 0$	$x = 2$	$x = 4$	$x = 6$	$x = 8$	$x = 10$
0	100	0	0	0	0	50
0.1	100	2.043923	0.021788	0.010894	1.021962	50
0.2	100	4.005178	0.084022	0.042672	2.002593	50
•						
•						
•						
3	100	37.54054	10.27449	6.442319	18.95732	50
6	100	53.24294	24.66052	17.4603	27.92251	50
9	100	62.39032	36.64937	27.84901	34.34692	50
12	100	68.71331	46.03498	36.54213	39.5355	50



30.2 Because we now have derivative boundary conditions, the boundary nodes must be simulated. For node 0,

$$T_0^{l+1} = T_0^l + \lambda(T_1^l - 2T_0^l + T_{-1}^l) \quad (i)$$

This introduces an exterior node into the solution at $i = -1$. The derivative boundary condition can be used to eliminate this node,

$$\left. \frac{dT}{dx} \right|_0 = \frac{T_1 - T_{-1}}{2\Delta x}$$

which can be solved for

$$T_{-1} = T_1 - 2\Delta x \frac{dT_0}{dx}$$

which can be substituted into Eq. (i) to give

$$T_0^{l+1} = T_0^l + \lambda \left(2T_1^l - 2T_0^l - 2\Delta x \frac{dT_0^l}{dx} \right)$$

For our case, $dT_0/dx = 1$ and $\Delta x = 2$, and therefore $T_{-1} = T_1 - 4$. This can be substituted into Eq. (i) to give,

$$T_0^{l+1} = T_0^l + \lambda(2T_1^l - 2T_0^l - 4)$$

A similar analysis can be used to embed the zero derivative in the equation for the n^{th} node,

$$T_n^{l+1} = T_n^l + \lambda(T_{n+1}^l - 2T_n^l + T_{n-1}^l) \quad (ii)$$

This introduces an exterior node into the solution at $n + 1$. The derivative boundary condition can be used to eliminate this node,

$$\left. \frac{dT}{dx} \right|_n = \frac{T_{n+1} - T_{n-1}}{2\Delta x}$$

which can be solved for

$$T_{n+1} = T_{n-1} + 2\Delta x \frac{dT_n}{dx}$$

which can be substituted into Eq. (ii) to give

$$T_n^{l+1} = T_n^l + \lambda \left(2T_{n-1}^l - 2T_n^l + 2\Delta x \frac{dT_n}{dx} \right)$$

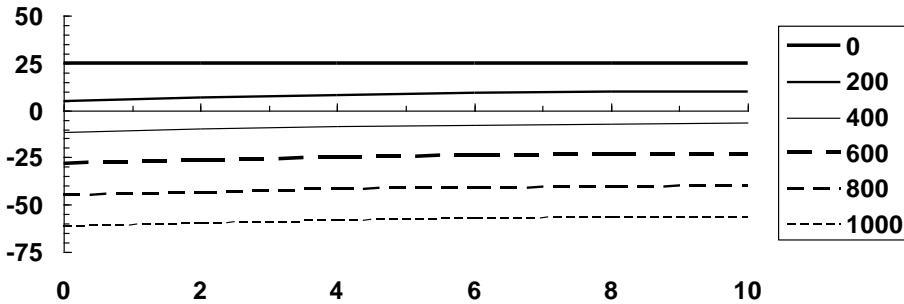
For our case, $n = 5$ and $dT_n/dx = 0$, and therefore

$$T_5^{l+1} = T_5^l + \lambda(2T_4^l - 2T_5^l)$$

Together with the equations for the interior nodes, the entire system can be solved with a step of 0.1 s. The results for some of the early steps along with some later selected values are tabulated below. In addition, a plot of the later results is also shown

t	x = 0	x = 2	x = 4	x = 6	x = 8	x = 10
0	25.0000	25.0000	25.0000	25.0000	25.0000	25.0000
0.1	24.9165	25.0000	25.0000	25.0000	25.0000	25.0000
0.2	24.8365	24.9983	25.0000	25.0000	25.0000	25.0000
0.3	24.7597	24.9949	25.0000	25.0000	25.0000	25.0000

0.4	24.6861	24.9901	24.9999	25.0000	25.0000	25.0000
0.5	24.6153	24.9840	24.9997	25.0000	25.0000	25.0000
•						
•						
•						
200	5.000081	6.800074	8.200059	9.200048	9.800042	10.00004
400	-11.6988	-9.89883	-8.49883	-7.49882	-6.89881	-6.69881
600	-28.4008	-26.6008	-25.2008	-24.2008	-23.6007	-23.4007
800	-45.1056	-43.3056	-41.9056	-40.9056	-40.3056	-40.1056
1000	-61.8104	-60.0104	-58.6104	-57.6104	-57.0104	-56.8104



Notice what's happening. The rod never reaches a steady state, because of the heat loss at the left end (unit gradient) and the insulated condition (zero gradient) at the right.

30.3 The solution for $\Delta t = 0.1$ is (as computed in Example 30.1),

<i>t</i>	<i>x = 0</i>	<i>x = 2</i>	<i>x = 4</i>	<i>x = 6</i>	<i>x = 8</i>	<i>x = 10</i>
0	100	0	0	0	0	50
0.1	100	2.0875	0	0	1.04375	50
0.2	100	4.087847	0.043577	0.021788	2.043923	50

For $\Delta t = 0.05$, it is

<i>t</i>	<i>x = 0</i>	<i>x = 2</i>	<i>x = 4</i>	<i>x = 6</i>	<i>x = 8</i>	<i>x = 10</i>
0	100	0.000000	0.000000	0.000000	0.000000	50
0.05	100	1.043750	0.000000	0.000000	0.521875	50
0.1	100	2.065712	0.010894	0.005447	1.032856	50
0.15	100	3.066454	0.032284	0.016228	1.533227	50
0.2	100	4.046528	0.063786	0.032229	2.023265	50

To assess the differences between the results, we performed the simulation a third time using a more accurate approach (the Heun method) with a much smaller step size ($\Delta t = 0.001$). It was assumed that this more refined approach would yield a prediction close to true solution. These values could then be used to assess the relative errors of the two Euler solutions. The results are summarized as

	<i>x = 0</i>	<i>x = 2</i>	<i>x = 4</i>	<i>x = 6</i>	<i>x = 8</i>	<i>x = 10</i>
Heun ($h = 0.001$)	100	4.006588	0.083044	0.042377	2.003302	50

Euler ($h = 0.1$)	100	4.087847	0.043577	0.021788	2.043923	50
Error relative to Heun		2.0%	47.5%	48.6%	2.0%	
Euler ($h = 0.05$)	100	4.046528	0.063786	0.032229	2.023265	50
Error relative to Heun		1.0%	23.2%	23.9%	1.0%	

Notice, that as would be expected for Euler's method, halving the step size approximately halves the global relative error.

30.4 The approach described in Example 30.2 must be modified to account for the zero derivative at the right hand node ($i = 5$). To do this, Eq. (30.8) is first written for that node as

$$-\lambda T_4^{l+1} + (1 + 2\lambda)T_5^{l+1} - \lambda T_6^{l+1} = T_5^l \quad (i)$$

The value outside the system ($i = 6$) can be eliminated by writing the finite difference relationship for the derivative at node 5 as

$$\frac{dT}{dx}\Big|_5 = \frac{T_6 - T_4}{2\Delta x}$$

which can be solved for

$$T_6 = T_4 + 2\Delta x \frac{dT}{dx}\Big|_5$$

For our case, $dT/dx = 0$, so $T_6 = T_4$ and Eq. (i) becomes

$$-2\lambda T_4^{l+1} + (1 + 2\lambda)T_5^{l+1} = T_5^l$$

Thus, the simultaneous equations to be solved at the first step are

$$\begin{bmatrix} 1.04175 & -0.020875 \\ -0.020875 & 1.04175 & -0.020875 \\ & -0.020875 & 1.04175 & -0.020875 \\ & & -0.020875 & 1.04175 & -0.020875 \\ & & & -0.04175 & 1.04175 \end{bmatrix} \begin{Bmatrix} T_1^1 \\ T_2^1 \\ T_3^1 \\ T_4^1 \\ T_5^1 \end{Bmatrix} = \begin{Bmatrix} 2.0875 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} 2.004645 \\ 0.040186 \\ 0.000806 \\ 1.62 \times 10^{-5} \\ 6.47 \times 10^{-7} \end{Bmatrix}$$

For the second step, the right-hand side is modified to reflect these computed values of T at $t = 0.1$,

$$\begin{bmatrix} 1.04175 & -0.020875 & & & \\ -0.020875 & 1.04175 & -0.020875 & & \\ & -0.020875 & 1.04175 & -0.020875 & \\ & & -0.020875 & 1.04175 & -0.020875 \\ & & & -0.04175 & 1.04175 \end{bmatrix} \begin{Bmatrix} T_1^1 \\ T_2^1 \\ T_3^1 \\ T_4^1 \\ T_5^1 \end{Bmatrix} = \begin{Bmatrix} 4.092145 \\ 0.040186 \\ 0.000806 \\ 1.62 \times 10^{-5} \\ 6.47 \times 10^{-7} \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} 3.930497 \\ 0.117399 \\ 0.003127 \\ 7.83 \times 10^{-5} \\ 3.76 \times 10^{-6} \end{Bmatrix}$$

30.5 The solution is identical to Example 30.3, but with 9 interior nodes. Thus, the simultaneous equations to be solved at the first step are

$$\begin{bmatrix} 2.167 & -0.0835 & & & & & & & \\ -0.0835 & 2.167 & -0.0835 & & & & & & \\ & -0.0835 & 2.167 & -0.0835 & & & & & \\ & & -0.0835 & 2.167 & -0.0835 & & & & \\ & & & -0.0835 & 2.167 & -0.0835 & & & \\ & & & & -0.0835 & 2.167 & -0.0835 & & \\ & & & & & -0.0835 & 2.167 & -0.0835 & \\ & & & & & & -0.0835 & 2.167 & -0.0835 \\ & & & & & & & -0.0835 & 2.167 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{Bmatrix} = \begin{Bmatrix} 16.7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8.35 \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} 7.717983 \\ 0.297836 \\ 0.011493 \\ 0.000444 \\ 0.000026 \\ 0.000222 \\ 0.005747 \\ 0.148918 \\ 3.858992 \end{Bmatrix}$$

For the second step, the right-hand side is modified to reflect these computed values of T at $t = 0.1$,

$$\begin{bmatrix} 2.167 & -0.0835 & & & & & & & \\ -0.0835 & 2.167 & -0.0835 & & & & & & \\ & -0.0835 & 2.167 & -0.0835 & & & & & \\ & & -0.0835 & 2.167 & -0.0835 & & & & \\ & & & -0.0835 & 2.167 & -0.0835 & & & \\ & & & & -0.0835 & 2.167 & -0.0835 & & \\ & & & & & -0.0835 & 2.167 & -0.0835 & \\ & & & & & & -0.0835 & 2.167 & -0.0835 \\ & & & & & & & -0.0835 & 2.167 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{Bmatrix} = \begin{Bmatrix} 30.8719 \\ 1.1913 \\ 0.04597 \\ 0.001775 \\ 0.000103 \\ 0.00089 \\ 0.002299 \\ 0.59567 \\ 15.436 \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} 14.28889 \\ 1.102814 \\ 0.063836 \\ 0.003288 \\ 0.000238 \\ 0.001650 \\ 0.031918 \\ 0.551407 \\ 7.144443 \end{Bmatrix}$$

30.6 Using the approach followed in Example 30.5, Eq. (30.20) is applied to nodes (1,1), (1,2), and (1,3) to yield the following tridiagonal equations

$$\begin{bmatrix} 2.167 & -0.0835 & & \\ -0.0835 & 2.167 & -0.0835 & \\ & -0.0835 & 2.167 & \end{bmatrix} \begin{Bmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \end{Bmatrix} = \begin{Bmatrix} 5.01 \\ 5.01 \\ 15.03 \end{Bmatrix}$$

which can be solved for

$$T_{1,1} = 2.415074 \quad T_{1,2} = 2.67624 \quad T_{1,3} = 7.038978$$

In a similar fashion, tridiagonal equations can be developed and solved for

$$T_{2,1} = 0.1044726 \quad T_{2,2} = 0.2962096 \quad T_{2,3} = 4.906547$$

and

$$T_{3,1} = 2.01846 \quad T_{3,2} = 2.278894 \quad T_{3,3} = 6.827404$$

For the second step to $t = 10$, Eq. (30.22) is applied to nodes (1,1), (2,1), and (3,1) to yield

$$\begin{bmatrix} 2.167 & -0.0835 & & \\ -0.0835 & 2.167 & -0.0835 & \\ & -0.0835 & 2.167 & \end{bmatrix} \begin{Bmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \end{Bmatrix} = \begin{Bmatrix} 9.660297 \\ 0.216232 \\ 8.065132 \end{Bmatrix}$$

which can be solved for

$$T_{1,1} = 4.47395 \quad T_{1,2} = 0.416205 \quad T_{1,3} = 3.737833$$

Tridiagonal equations for the other rows can be developed and solved for

$$T_{2,1} = 5.050756 \quad T_{2,2} = 0.815685 \quad T_{2,3} = 4.292810$$

and

$$T_{3,1} = 13.462935 \quad T_{3,2} = 9.820288 \quad T_{3,3} = 12.869439$$

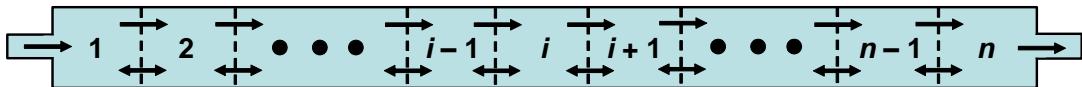
Thus, the result at the end of the first step can be summarized as

	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
$j = 4$	90	120	120	120	85
$j = 3$	60	13.46293	9.820288	12.86944	50
$j = 2$	60	5.050756	0.815685	4.29281	50
$j = 1$	60	4.47395	0.416205	3.737833	50
$j = 0$	30	0	0	0	25

The computation can be repeated and the results for $t = 2000$ s are shown below:

	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
$j = 4$	90	120	120	120	85
$j = 3$	60	80.71429	83.83929	77.14286	50
$j = 2$	60	59.01786	57.5	54.73214	50
$j = 1$	60	37.85714	32.41071	34.28571	50
$j = 0$	30	0	0	0	25

30.7 Although this problem can be modeled with the finite-difference approach (see Sec. 32.1), the control-volume method provides a more straightforward way to handle the boundary conditions. Thus, the tank is idealized as a series of control volumes:



The boundary fluxes and the reaction term can be used to develop the discrete form of the advection-diffusion equation for the interior volumes as

$$\Delta x \frac{dc_i^l}{dt} = -D \frac{c_i^l - c_{i-1}^l}{\Delta x} + D \frac{c_{i+1}^l - c_i^l}{\Delta x} + U \frac{c_i^l + c_{i-1}^l}{2} - U \frac{c_{i+1}^l + c_i^l}{2} - k \Delta x c_i^l$$

or dividing both sides by Δx ,

$$\frac{dc_i^l}{dt} = D \frac{c_{i+1}^l - 2c_i^l + c_{i-1}^l}{\Delta x^2} - U \frac{c_{i+1}^l - c_{i-1}^l}{2\Delta x} - k c_i^l$$

which is precisely the form that would have resulted by substituting centered finite difference approximations into the advection-diffusion equation.

For the first boundary node, no diffusion is allowed up the entrance pipe and advection is handled with a backward difference,

$$\Delta x \frac{dc_1^l}{dt} = D \frac{c_2^l - c_1^l}{\Delta x} + U c_0^l - U \frac{c_2^l + c_1^l}{2} - k \Delta x c_1^l$$

or dividing both sides by Δx ,

$$\frac{dc_1^l}{dt} = D \frac{c_2^l - c_1^l}{\Delta x^2} + U \frac{2c_0^l - c_2^l - c_1^l}{2\Delta x} - k c_1^l$$

For the last boundary node, no diffusion is allowed through the exit pipe and advection out of the tank is again handled with a backward difference,

$$\Delta x \frac{dc_n^l}{dt} = D \frac{c_{n-1}^l - c_n^l}{\Delta x} + U \frac{c_n^l + c_{n-1}^l}{2} - U c_n^l - k \Delta x c_n^l$$

or dividing both sides by Δx ,

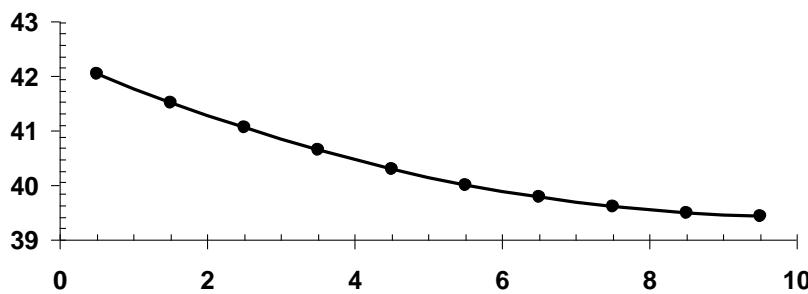
$$\frac{dc_n^l}{dt} = D \frac{c_{n-1}^l - c_n^l}{\Delta x^2} + U \frac{c_{n-1}^l - c_n^l}{2\Delta x} - k c_n^l$$

By writing these equations for each equally-spaced volume, the PDE is transformed into a system of ODEs. Explicit methods like Euler's method or other higher-order RK methods can then be used to solve the system.

The results with an initial condition that the reactor has zero concentration with an inflow concentration of 100 (using Euler with a step size of 0.005) for $t = 100$ are

x	0.5	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
c	42.0320	41.5128	41.0509	40.6463	40.2989	40.0087	39.7760	39.6008	39.4836	39.4248

A plot of the results is shown below:



30.8 Here is a VBA program that implements the explicit method. It is set up to duplicate Example 30.1.

```

Option Explicit

Sub Explicit()
    Dim i As Integer, j As Integer, np As Integer, ns As Integer
    Dim Te(20) As Double, dTe(20) As Double, tpr(20) As Double, Tepr(20, 20) As
        Double
    Dim k As Double, dx As Double, L As Double, tc As Double, tf As Double
    Dim tp As Double, t As Double, tend As Double, h As Double, tol As Double
    Dim x As Double
    tol = 0.000001
    L = 10
    ns = 5
    dx = L / ns
    k = 0.835
    Te(0) = 100
    Te(5) = 50
    tc = 0.1
    tf = 12
    tp = 3
    np = 0
    tpr(np) = t
    For i = 0 To ns
        Tepr(i, np) = Te(i)
    Next i
    Do
        tend = t + tp
        If tend > tf Then tend = tf
        h = tc
        Do
            If t + h > tend Then h = tend - t
            Call Derivs(Te, dTe, ns, dx, k)
            For j = 1 To ns - 1
                Te(j) = Te(j) + dTe(j) * h
            Next j
            t = t + h
            If t >= tend Then Exit Do
        Loop
        np = np + 1
        tpr(np) = t
        For j = 0 To ns
            Tepr(j, np) = Te(j)
        Next j
        If t + tol >= tf Then Exit Do
    Loop
    Sheets("sheet1").Select
    Range("a4:bb5000").ClearContents
    Range("a4").Select
    ActiveCell.Value = "time"
    x = 0
    For j = 0 To ns
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = "x = " & x
        x = x + dx
    Next j
    Range("a5").Select

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

For i = 0 To np
    ActiveCell.Value = tpr(i)
    For j = 0 To ns
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = Tepr(j, i)
    Next j
    ActiveCell.Offset(1, -ns - 1).Select
Next i
Range("a5").Select
End Sub

Sub Derivs(Te, dTe, ns, dx, k)
Dim j As Integer
For j = 1 To ns - 1
    dTe(j) = k * (Te(j - 1) - 2 * Te(j) + Te(j + 1)) / dx ^ 2
Next j
End Sub

```

When the program is run, the result is

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4	time	x = 0	x = 2	x = 4	x = 6	x = 8	x = 10		
5	0	100	0	0	0	0	50		
6	3	100	37.8557	10.24094	6.378532	19.09671	50		
7	6	100	53.44055	24.75211	17.49043	27.99545	50		
8	9	100	62.52413	36.78229	27.93798	34.40585	50		
9	12	100	68.81906	46.1739	36.65582	39.59994	50		

30.9 This VBA program is set up to either use Dirichlet or gradient boundary conditions depending on the values of the parameters *istrt* and *iend*. It is set up to solve the first few steps of Prob. 30.2.

```

Option Explicit

Sub EulerPDE()
Dim i As Integer, j As Integer, np As Integer, ns As Integer
Dim istrt As Integer, iend As Integer
Dim Te(2000) As Double, dTe(2000) As Double
Dim tpr(20000) As Double, Tepr(2000, 20000) As Double
Dim k As Double, dx As Double, L As Double, tc As Double, tf As Double
Dim tp As Double, t As Double, tend As Double, h As Double
Dim dTedx(2000) As Double, tol As Double, x As Double
tol = 0.000001
L = 10
ns = 5
dx = L / ns
k = 0.835
dTedx(0) = 1
istrt = 0
dTedx(ns) = 0
iend = ns
Te(0) = 25
Te(1) = 25
Te(2) = 25
Te(3) = 25
Te(4) = 25
Te(5) = 25

```

```

tc = 0.1
tf = 0.5
tp = 0.1
np = 0
tpr(np) = t
For i = 0 To ns
    Tepr(i, np) = Te(i)
Next i
Do
    tend = t + tp
    If tend > tf Then tend = tf
    h = tc
    Do
        If t + h > tend Then h = tend - t
        Call Derivs(Te, dTe, istrat, iend, ns, dx, k, dTedx)
        For j = istrat To iend
            Te(j) = Te(j) + dTe(j) * h
        Next j
        t = t + h
        If t >= tend Then Exit Do
    Loop
    np = np + 1
    tpr(np) = t
    For j = 0 To ns
        Tepr(j, np) = Te(j)
    Next j
    If t + tol >= tf Then Exit Do
Loop
Sheets("sheet1").Select
Range("a4:bb5000").ClearContents
Range("a4").Select
ActiveCell.Value = "time"
x = 0
For j = 0 To ns
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "x = " & x
    x = x + dx
Next j
Range("a5").Select
For i = 0 To np
    ActiveCell.Value = tpr(i)
    For j = 0 To ns
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = Tepr(j, i)
    Next j
    ActiveCell.Offset(1, -ns - 1).Select
Next i
Range("a5").Select
End Sub

Sub Derivs(Te, dTe, istrat, iend, ns, dx, k, dTedx)
Dim j As Integer
If istrat = 0 Then
    dTe(0) = k * (2 * Te(1) - 2 * Te(0) - 2 * dx * dTedx(0)) / dx ^ 2
End If
For j = 1 To ns - 1
    dTe(j) = k * (Te(j - 1) - 2 * Te(j) + Te(j + 1)) / dx ^ 2
Next j
If iend = ns Then
    dTe(ns) = k * (2 * Te(ns - 1) - 2 * Te(ns) + 2 * dx * dTedx(ns)) / dx ^ 2
End If
End Sub

```

When the program is run, the result is

	A	B	C	D	E	F	G	H	I
1									RUN
2									
3									
4	time	x = 0	x = 2	x = 4	x = 6	x = 8	x = 10		
5	0	25	25	25	25	25	25		
6	0.1	24.9165	25	25	25	25	25		
7	0.2	24.83649	24.99826	25	25	25	25		
8	0.3	24.75974	24.99492	24.99996	25	25	25		
9	0.4	24.68606	24.99011	24.99986	25	25	25		
10	0.5	24.61525	24.98397	24.99966	25	25	25		

30.10 Here is a VBA program that implements the implicit method. It is set up to duplicate Example 30.2.

```

Option Explicit

Sub Implicit()
Dim np As Integer, ns As Integer, i As Integer, j As Integer, n As Integer
Dim Te(10) As Double, dTe(10) As Double, tpr(100) As Double
Dim Tepr(10, 100) As Double, Tei As Double
Dim k As Double, dx As Double, L As Double, tc As Double, tf As Double
Dim tp As Double, t As Double, tend As Double, h As Double, lambda As Double
Dim e(10) As Double, f(10) As Double, g(10) As Double
Dim r(10) As Double, x(10) As Double, xrod As Double
Dim tol As Double
tol = 0.000001
L = 10#
ns = 5
dx = L / ns
k = 0.835
Te(0) = 100#
Te(ns) = 50#
Tei = 0
For i = 1 To ns - 1
    Te(i) = Tei
Next i
t = 0
np = 0
tpr(np) = t
For i = 0 To ns
    Tepr(i, np) = Te(i)
Next i
tc = 0.1
tp = 0.1
tf = 0.5
Do
    tend = t + tp
    If tend > tf Then tend = tf
    h = tc
    Do
        If t + h > tend Then h = tend - t
        lambda = k * h / dx ^ 2
        f(1) = 1 + 2 * lambda
        g(1) = -lambda
        r(1) = Te(1) + lambda * Te(0)
        For j = 2 To ns - 2
            Tepr(j, np) = Te(j)
            Te(j) = Te(j) + lambda * (Te(j + 1) - Te(j - 1)) / (2 * dx)
        Next j
        Tepr(ns, np) = Te(ns)
    Loop Until Abs(h) < tol
    t = tend
    np = np + 1
    tpr(np) = t
    If np > 100 Then Exit Do
Loop
End Sub

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

e(j) = -lambda
f(j) = 1 + 2 * lambda
g(j) = -lambda
r(j) = Te(j)
Next j
e(ns - 1) = -lambda
f(ns - 1) = 1 + 2 * lambda
r(ns - 1) = Te(ns - 1) + lambda * Te(ns)
Call Tridiag(e, f, g, r, Te, ns - 1)
t = t + h
If t >= tend Then Exit Do
Loop
np = np + 1
tpr(np) = t
For j = 0 To ns
    Tepr(j, np) = Te(j)
Next j
If t + tol >= tf Then Exit Do
Loop
Sheets("sheet1").Select
Range("a4:bb5000").ClearContents
Range("a4").Select
ActiveCell.Value = "time"
xrod = 0
For j = 0 To ns
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "x = " & xrod
    xrod = xrod + dx
Next j
Range("a5").Select
For i = 0 To np
    ActiveCell.Value = tpr(i)
    For j = 0 To ns
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = Tepr(j, i)
    Next j
    ActiveCell.Offset(1, -ns - 1).Select
Next i
Range("a5").Select
End Sub

Sub Tridiag(e, f, g, r, x, n)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

When the program is run, the result is

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4	time	x = 0	x = 2	x = 4	x = 6	x = 8	x = 10		RUN
5	0	100	0	0	0	0	50		
6	0.1	100	2.004653	0.040589	0.020899	1.002339	50		
7	0.2	100	3.930536	0.118963	0.061827	1.965327	50		
8	0.3	100	5.781512	0.232491	0.121937	2.890926	50		
9	0.4	100	7.561235	0.378704	0.200404	3.781003	50		
10	0.5	100	9.273172	0.555286	0.296424	4.637332	50		

30.11 Here is a VBA program that implements the Crank-Nicolson method. It is set up to duplicate Example 30.3.

```

Option Explicit

Sub CrankNic()
    Dim np As Integer, ns As Integer, i As Integer, j As Integer, n As Integer
    Dim Te(10) As Double, dTe(10) As Double, tpr(100) As Double
    Dim Tepr(10, 100) As Double, Tei As Double
    Dim k As Double, dx As Double, L As Double, tc As Double, tf As Double
    Dim tp As Double, t As Double, tend As Double, h As Double, lambda As Double
    Dim e(10) As Double, f(10) As Double, g(10) As Double
    Dim r(10) As Double, x(10) As Double, xrod As Double
    Dim tol As Double
    tol = 0.000001
    L = 10#
    ns = 5
    dx = L / ns
    k = 0.835
    Te(0) = 100#
    Te(ns) = 50#
    Tei = 0
    t = 0
    np = 0
    tpr(np) = t
    For i = 0 To ns
        Tepr(i, np) = Tei
    Next i
    tc = 0.1
    tf = 0.5
    tp = 0.1
    Do
        tend = t + tp
        If tend > tf Then tend = tf
        h = tc
        Do
            If t + h > tend Then h = tend - t
            lambda = k * h / dx ^ 2
            f(1) = 2 * (1 + lambda)
            g(1) = -lambda
            r(1) = lambda * Te(0) + 2 * (1 - lambda) * Te(1) + lambda * Te(2)
            r(1) = r(1) + lambda * Te(0)
            For j = 2 To ns - 2
                e(j) = -lambda
                f(j) = 2 * (1 + lambda)
                g(j) = -lambda
                r(j) = lambda * Te(j - 1) + 2 * (1 - lambda) * Te(j) + lambda * Te(j + 1)
            Next j
            e(ns - 1) = -lambda
        End If
        t = tend
        tpr(i, ns) = t
    Loop
End Sub

```

```

f(ns - 1) = 2 * (1 + lambda)
r(ns - 1) = lambda * Te(ns - 2) + 2 * (1 - lambda) * Te(ns - 1)
r(ns - 1) = r(ns - 1) + lambda * Te(ns) + lambda * Te(ns)
Call Tridiag(e, f, g, r, Te, ns - 1)
t = t + h
If t >= tend Then Exit Do
Loop
np = np + 1
tpr(np) = t
For j = 0 To ns
    Texpr(j, np) = Te(j)
Next j
If t + tol >= tf Then Exit Do
Loop
Sheets("sheet1").Select
Range("a4:bb5000").ClearContents
Range("a4").Select
ActiveCell.Value = "time"
xrod = 0
For j = 0 To ns
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = "x = " & xrod
    xrod = xrod + dx
Next j
Range("a5").Select
For i = 0 To np
    ActiveCell.Value = tpr(i)
    For j = 0 To ns
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = Texpr(j, i)
    Next j
    ActiveCell.Offset(1, -ns - 1).Select
Next i
Range("a5").Select
End Sub

Sub Tridiag(e, f, g, r, x, n)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

When the program is run, the result is

	A	B	C	D	E	F	G	H	I
1									
2									
3								RUN	
4	time	x = 0	x = 2	x = 4	x = 6	x = 8	x = 10		
5	0	0	0	0	0	0	0		
6	0.1	100	2.045029	0.021018	0.010669	1.022516	50		
7	0.2	100	4.007269	0.082578	0.042232	2.003647	50		
8	0.3	100	5.890904	0.181791	0.093808	2.945504	50		
9	0.4	100	7.699891	0.315951	0.164539	3.850092	50		
10	0.5	100	9.437972	0.482524	0.253588	4.71932	50		

30.12 Here is VBA code to solve this problem. The Excel output is also attached showing values for the first two steps along with selected snapshots of the solution as it evolves in time.

```

Option Explicit

Sub ADI()
Dim np As Integer, i As Integer, j As Integer
Dim nx As Integer, ny As Integer
Dim Lx As Double, dx As Double
Dim Ly As Double, dy As Double
Dim Te(10, 10) As Double, dTe(10, 10) As Double
Dim tpr(100) As Double, Tepr(10, 10, 100) As Double, Tei As Double
Dim k As Double
Dim dt As Double, ti As Double, tf As Double, tp As Double
Dim t As Double, tend As Double, h As Double
Dim lamx As Double, lamy As Double
Dim e(10) As Double, f(10) As Double, g(10) As Double
Dim r(10) As Double, Ted(10) As Double
'set computation parameters
Lx = 40
nx = 4
dx = Lx / nx
Ly = 40
ny = 4
dy = Ly / ny
k = 0.835
dt = 10
tf = 500
ti = 0
tp = 10
Tei = 0
'set top boundary
For i = 1 To nx - 1
    Te(i, ny) = 100
Next i
'set bottom boundary
For i = 1 To nx - 1
    Te(i, 0) = 0
Next i
'set left boundary
For j = 1 To ny - 1
    Te(0, j) = 75
Next j
'set right boundary
For j = 1 To ny - 1
    Te(nx, j) = 50
Next j
'set corners for plot
Te(0, 0) = (dy * Te(1, 0) + dx * Te(0, 1)) / (dy + dx)
Te(nx, 0) = (dy * Te(nx - 1, 0) + dx * Te(nx, 1)) / (dy + dx)
Te(0, ny) = (dy * Te(1, ny) + dx * Te(0, ny - 1)) / (dy + dx)
Te(nx, ny) = (dy * Te(nx - 1, ny) + dx * Te(nx, ny - 1)) / (dy + dx)
'set interior
For i = 1 To nx - 1
    For j = 1 To ny - 1

```

```

Te(i, j) = Tei
Next j
Next i
'save initial values for output
np = 0
t = ti
tpr(np) = t
For i = 0 To nx
    For j = 0 To ny
        Tepr(i, j, np) = Te(i, j)
    Next j
Next i
Do
    tend = t + tp
    If tend > tf Then tend = tf
    h = dt
    Do
        If t + h > tend Then h = tend - t
        'Sweep y
        lamx = k * h / dx ^ 2
        lamy = k * h / dy ^ 2
        For i = 1 To nx - 1
            f(1) = 2 * (1 + lamy)
            g(1) = -lamy
            r(1) = lamx * Te(i - 1, 1) + 2 * (1 - lamx) * Te(i, 1) + lamx * Te(i + 1, 1) -
                    + lamy * Te(i, 0)
            For j = 2 To ny - 2
                e(j) = -lamy
                f(j) = 2 * (1 + lamy)
                g(j) = -lamy
                r(j) = lamx * Te(i - 1, j) + 2 * (1 - lamx) * Te(i, j) + lamx * Te(i + 1, j)
            Next j
            e(ny - 1) = -lamy
            f(ny - 1) = 2 * (1 + lamy)
            r(ny - 1) = lamx * Te(i - 1, ny - 1) + 2 * (1 - lamx) * Te(i, ny - 1) -
                        + lamx * Te(i + 1, ny - 1) + lamy * Te(i, nx)
            Call Tridiag(e, f, g, r, Ted, nx - 1)
            For j = 1 To ny - 1
                Te(i, j) = Ted(j)
            Next j
        Next i
        t = t + h / 2
        'Sweep x
        For j = 1 To ny - 1
            f(1) = 2 * (1 + lamx)
            g(1) = -lamx
            r(1) = lamy * Te(1, j - 1) + 2 * (1 - lamy) * Te(1, j) + lamy * Te(1, j + 1) -
                    + lamx * Te(0, j)
            For i = 2 To nx - 2
                e(i) = -lamx
                f(i) = 2 * (1 + lamx)
                g(i) = -lamx
                r(i) = lamy * Te(i, j - 1) + 2 * (1 - lamy) * Te(i, j) + lamy * Te(i, j + 1)
            Next i
            e(nx - 1) = -lamx
            f(nx - 1) = 2 * (1 + lamx)
            r(nx - 1) = lamy * Te(nx - 1, j - 1) + 2 * (1 - lamy) * Te(nx - 1, j) -
                        + lamy * Te(nx - 1, j + 1) + lamx * Te(ny, j)
            Call Tridiag(e, f, g, r, Ted, nx - 1)
            For i = 1 To nx - 1
                Te(i, j) = Ted(i)
            Next i
            Next j
            t = t + h / 2
            If t >= tend Then Exit Do
        Loop
        'save values for output
        np = np + 1
        tpr(np) = t
    End If
End Sub

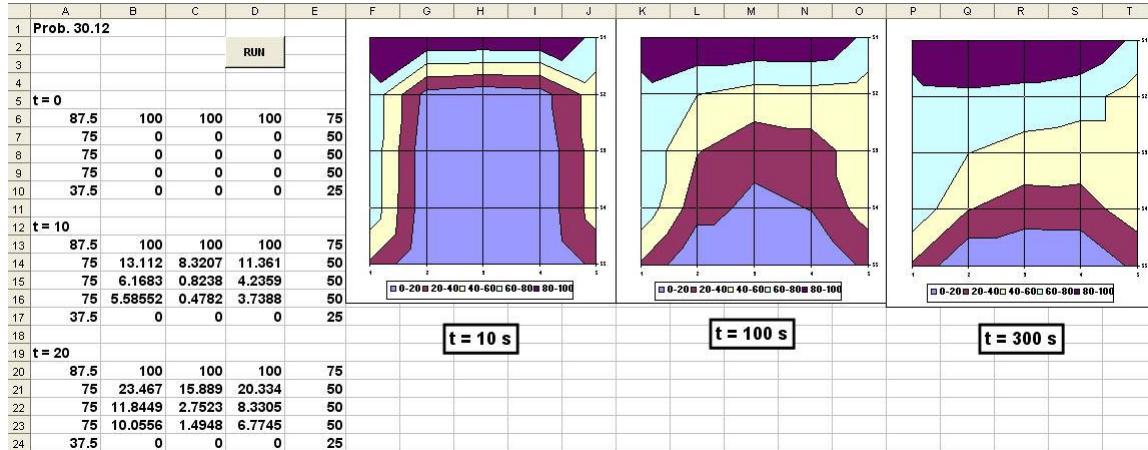
```

```

For i = 0 To nx
    For j = 0 To ny
        Tepr(i, j, np) = Te(i, j)
    Next j
Next i
If t >= tf Then Exit Do
Loop
'output results back to sheet
Range("a5").Select
Range("a5:e2005").ClearContents
For k = 0 To np
    ActiveCell.Value = "t = " & tpr(k)
    ActiveCell.Offset(1, 0).Select
    For j = ny To 0 Step -1
        For i = 0 To nx
            ActiveCell.Value = Tepr(i, j, k)
            ActiveCell.Offset(0, 1).Select
        Next i
        ActiveCell.Offset(1, -nx - 1).Select
    Next j
    ActiveCell.Offset(1, 0).Select
Next k
Range("a5").Select
End Sub

Sub Tridiag(e, f, g, r, x, n)
Dim k As Integer
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```



30.13 MATLAB solution:

```

%PDE Parabolic Problem - Heat conduction in a rod
% u[xx]=u[t]
% BC u(0,t)=0 u(1,t)=1
% IC u(x,0)=0 x<1
% i=spatial index, from 1 to imax
% imax = no. of x points

```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

```

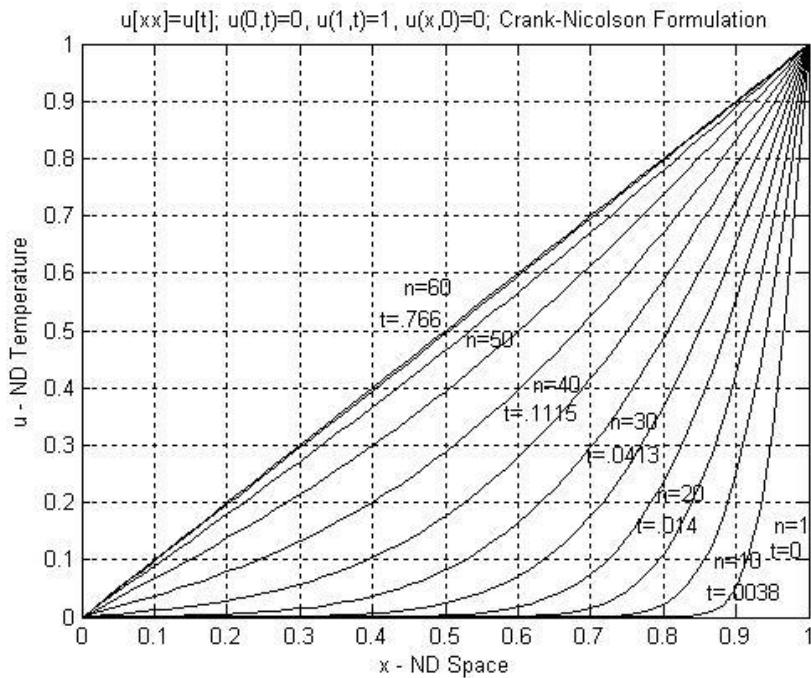
% n=time index from 1 to nmax
% nmax = no. of time steps,
% Crank-Nicolson Formulation
imax=61;
nmax=60;      % last time step = nmax+1
% Constants
dx=1/(imax-1);
dx2=dx*dx;
dt=dx2;       % Setting dt to dx2 for good stability and results
% Independent space variable
x=0:dx:1;
% Sizing matrices
u=zeros(imax,nmax+1); t=zeros(1,nmax+1);
a=zeros(1,imax); b=zeros(1,imax);
c=zeros(1,imax); d=zeros(1,imax);
ba=zeros(1,imax); ga=zeros(1,imax);
up=zeros(1,imax);
% Boundary Conditions
u(1,1)=0;
u(imax,1)=1;
% Time step loop
% n=1 represents 0 time, n+1 = next time step
t(1)=0;
for n=1:nmax
    t(n+1)=t(n)+dt;
    %Boundary conditions & Constants
    u(1,n+1)=0;
    u(imax,n+1)=1;
    dx2dt=dx2/dt;
    % coefficients
    b(2)=-2-2*dx2dt;
    c(2)=1;
    d(2)=(2-2*dx2dt)*u(2,n)-u(3,n);
    for i=3:imax-2
        a(i)=1;
        b(i)=-2-2*dx2dt;
        c(i)=1;
        d(i)=-u(i-1,n)+(2-2*dx2dt)*u(i,n)-u(i+1,n);
    end
    a(imax-1)=1;
    b(imax-1)=-2-2*dx2dt;
    d(imax-1)=-u(imax-2,n)+(2-2*dx2dt)*u(imax-1,n)-2;
    % Solution by Thomas Algorithm
    ba(2)=b(2);
    ga(2)=d(2)/b(2);
    for i=3:imax-1
        ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
        ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
    end
    % Back substitution step
    u(imax-1,n+1)=ga(imax-1);
    for i=imax-2:-1:2
        u(i,n+1)=ga(i)-c(i)*u(i+1,n+1)/ba(i);
    end
    dt=1.1*dt;
end
% end of time step loop
% Plot
% Storing plot value of u as up, at every 5 time steps, np=5
% j=time index
% i=space index
np=5;

```

```

for j=np:np:nmax
    for i=1:imax
        up(i)=u(i,j);
    end
    plot(x,up)
    hold on
end
grid
title('u[xx]=u[t]; u(0,t)=0, u(1,t)=1, u(x,0)=0; Crank-Nicolson Formulation')
xlabel('x - ND Space')
ylabel('u - ND Temperature')
hold off
gtext('n=60');gtext('n=50');gtext('n=40');gtext('n=30');
gtext('n=20');gtext('n=10');gtext('n=1');gtext('t=.766');
gtext('t=.1115');gtext('t=.0413');gtext('t=.014');
gtext('t=.0038');gtext('t=0')

```



30.14

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} = \frac{\partial u}{\partial t}$$

Substituting of second order correct Crank-Nicolson analogues

$$\frac{\partial^2 u}{\partial r^2} = \frac{1}{2} \left[\frac{u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{\Delta r^2} + \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta r^2} \right]$$

$$\frac{\partial u}{\partial r} = \frac{1}{2} \left[\frac{u_{i+1,n+1} - u_{i-1,n+1}}{2\Delta r} + \frac{u_{i+1,n} - u_{i-1,n}}{2\Delta r} \right]$$

$$r = (i - 1)\Delta r$$

$$\frac{\partial u}{\partial t} = \frac{u_{i,n+1} - u_{i,n}}{\Delta t}$$

into the governing equation give the following finite difference equations:

$$\left[1 - \frac{1}{2(i-1)} \right] u_{i-1,n+1} + \left[-2 - 2 \frac{\Delta r^2}{\Delta t} \right] u_{i,n+1} + \left[1 + \frac{1}{2(i-1)} \right] u_{i+1,n+1} = \left[-1 + \frac{1}{2(i-1)} \right] u_{i-1,n} \\ + \left[2 - 2 \frac{\Delta r^2}{\Delta t} \right] u_{i,n} + \left[-1 - \frac{1}{2(i-1)} \right] u_{i+1,n}$$

For the end points:

$x = 1$ ($i = R$), substitute the value of $u_R = 1$ into the above FD equation
 $x = 0$ ($i = 1$), set the FD analog to the first derivative = 0

$$\left[\frac{\partial u}{\partial r} \right]_{i=1} = \frac{1}{2} \left[\frac{u_{2,n+1} - u_{0,n+1}}{2\Delta r} + \frac{u_{2,n} - u_{0,n}}{2\Delta r} \right] = 0$$

Also substitute in $i = 1$ into the finite difference equation and algebraically eliminate $u_{0,n+1} + u_{0,n}$ from the two equations and get the FD equation at $i = 1$:

$$\left[-2 - 2 \frac{\Delta r^2}{\Delta t} \right] u_{1,n+1} + [2] u_{2,n+1} = -\left[2 - 2 \frac{\Delta r^2}{\Delta t} \right] u_{1,n} + [-2] u_{2,n}$$

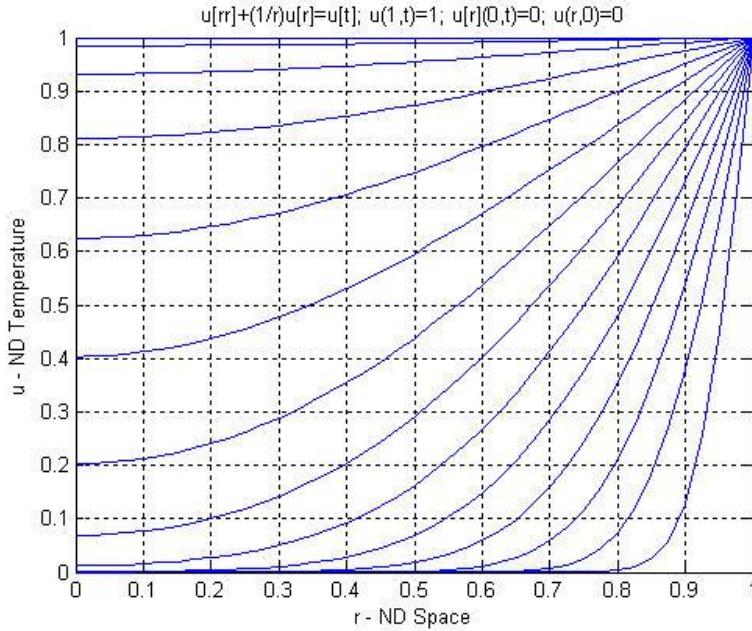
Here is a MATLAB implementation:

```
%PDE Parabolic Problem - Heat conduction in the radial direction in a circular rod
% u[rr]+(1/r)u[r]=u[t]           0< r < 1
% BC    u(1,t)=1     u[r](0,t)=0
% IC    u(r,0)=0           0 < r < 1
% i=spatial index, from 1 to imax
% imax = no. of r points (imax=21 for 20 dr spaces)
% n=time index from 1 to nmax
% nmax = no. of time steps,
% Crank-Nicolson Formulation
imax=41;
nmax=60;                                     % last time step = nmax+1
% Constants
dr=1/(imax-1);
dr2=dr*dr;
dt=dr2;   % Setting dt to dr2 for good stability and results
% Independent space variable
r=0:dr:1;
% Sizing matrices
u=zeros(imax,nmax+1); t=zeros(1,nmax+1);
a=zeros(1,imax); b=zeros(1,imax);
```

```

c=zeros(1,imax); d=zeros(1,imax);
ba=zeros(1,imax); ga=zeros(1,imax);
up=zeros(1,imax);
% Boundary Conditions
u(imax,1)=1;
% Time step loop
% n=1 represents 0 time, new time = n+1
t(1)=0;
for n=1:nmax
    t(n+1)=t(n)+dt;
    % Boundary conditions & Constants
    u(imax,n+1)=1;
    dr2dt=dr2/dt;
    % coefficients
    b(1)=-2-2*dr2dt;
    c(1)=2;
    d(1)=(2-2*dr2dt)*u(1,n)-2*u(2,n);
    for i=2:imax-2
        a(i)=1-1/(2*(i-1));
        b(i)=-2-2*dr2dt;
        c(i)=1+1/(2*(i-1));
        d(i)=(-1+1/(2*(i-1)))*u(i-1,n)+(2-2*dr2dt)*u(i,n)+(-1-1/(2*(i-1)))*u(i+1,n);
    end
    a(imax-1)=1-1/(2*(imax-2));
    b(imax-1)=-2-2*dr2dt;
    d(imax-1)=(-1+1/(2*(imax-2)))*u(imax-2,n)+(2-2*dr2dt)*u(imax-1,n)-2*(1+1/(2*(imax-2)));
    % Solution by Thomas Algorithm
    ba(1)=b(1);
    ga(1)=d(1)/ba(1);
    for i=2:imax-1
        ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
        ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
    end
    % Back substitution step
    u(imax-1,n+1)=ga(imax-1);
    for i=imax-2:-1:1
        u(i,n+1)=ga(i)-c(i)*u(i+1,n+1)/ba(i);
    end
    dt=1.1*dt;
end
% end of time step loop
% Plot
% Storing plot value of u as up, at every 5 time steps
% j=time index
% i=space index
istart=4;
for j=istart:istart:nmax+1
    for i=1:imax
        up(i)=u(i,j);
    end
    plot(r,up)
    hold on
end
grid
title('u[rr]+(1/r)u[r]=u[t]; u(1,t)=1; u[r](0,t)=0; u(r,0)=0')
xlabel('r - ND Space')
ylabel('u - ND Temperature')
hold off

```



30.15

$$\frac{\partial^2 u}{dx^2} + b \frac{\partial u}{\partial x} = \frac{\partial u}{\partial t}$$

Substituting of second order correct Crank-Nicolson analogues

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} &= \frac{1}{2} \left[\frac{u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{\Delta x^2} + \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{\Delta x^2} \right] \\ \frac{\partial u}{\partial x} &= \frac{1}{2} \left[\frac{u_{i+1,n+1} - u_{i-1,n+1}}{2\Delta x} + \frac{u_{i+1,n} - u_{i-1,n}}{2\Delta x} \right] \\ \frac{\partial u}{\partial t} &= \frac{u_{i,n+1} - u_{i,n}}{\Delta t}\end{aligned}$$

into the governing equation give the following finite difference equations

$$\begin{aligned}\left[1 - \frac{1}{2} b \Delta x \right] u_{i-1,n+1} + \left[-2 - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n+1} + \left[1 + \frac{1}{2} b \Delta x \right] u_{i+1,n+1} &= \left[-1 + \frac{1}{2} b \Delta x \right] u_{i-1,n} \\ &\quad + \left[2 - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n} + \left[-1 - \frac{1}{2} b \Delta x \right] u_{i+1,n}\end{aligned}$$

```
%PDE Parabolic Problem with a dispersion term
% u[xx]+bu[x]=u[t]
% BC u(0,t)=0 u(1,t)=1
% IC u(x,0)=0 x<1
% i=spatial index, from 1 to imax
```

```

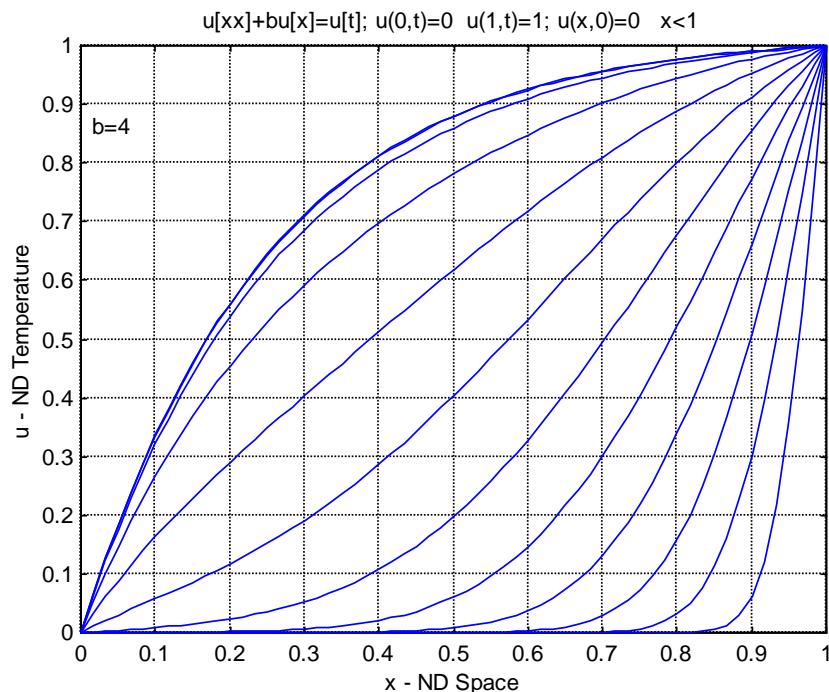
%   imax = no. of spatial points (imax=21 for 20 dx spaces)
%   n = time index, from 1 to nmax
%   nmax = no. of time steps
%   Crank-Nicholson formulation for the spatial derivatives
imax=61;
nmax=60;           % last time step = nmax+1
% constants
dx=1/(imax-1);
dx2=dx*dx;
dt=dx2;
% Parameters
B=-4;
% Independent spatial variable
x=0:dx:1;
% Sizing matrices
u=zeros(imax,nmax); t=zeros(1,nmax);
a=zeros(1,imax); b=zeros(1,imax);
c=zeros(1,imax); d=zeros(1,imax);
ba=zeros(1,imax); ga=zeros(1,imax);
up=zeros(1,imax);
% Boundary Conditions
u(1,1)=0;
u(imax,1)=1;
% Time step loop
% n=1 represents 0 time, new time = n+1
t(1)=0;
for n=1:nmax
    t(n+1)=t(n)+dt;
    % Boundary conditions & constants
    u(1,n+1)=0;
    u(imax,n+1)=1;
    dx2dt=dx2/dt;
    % Coefficients
    b(2)=-2-2*dx2dt;
    c(2)=1+0.5*B*dx;
    d(2)=(-1-0.5*B*dx)*u(3,n)+(2-2*dx2dt)*u(2,n);
    for i=3:imax-2
        a(i)=1-0.5*B*dx;
        b(i)=-2-2*dx2dt;
        c(i)=1+0.5*B*dx;
        d(i)=(-1-0.5*B*dx)*u(i+1,n)+(2-2*dx2dt)*u(i,n)+(-1+0.5*B*dx)*u(i-1,n);
    end
    a(imax-1)=1-0.5*B*dx;
    b(imax-1)=-2-2*dx2dt;
    d(imax-1)=2*(-1-0.5*B*dx)+(2-2*dx2dt)*u(imax-1,n)+(-1+0.5*B*dx)*u(imax-2,n);
    % Solution by Thomas Algorithm
    ba(2)=b(2);
    ga(2)=d(2)/b(2);
    for i=3:imax-1
        ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
        ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
    end
    % Back substitution step
    u(imax-1,n+1)=ga(imax-1);
    for i=imax-2:-1:2
        u(i,n+1)=ga(i)-c(i)*u(i+1,n+1)/ba(i);
    end
    dt=1.1*dt;
end
% End of time step loop
%Plot
%Storing plot value of u as up, at ever 5 time steps

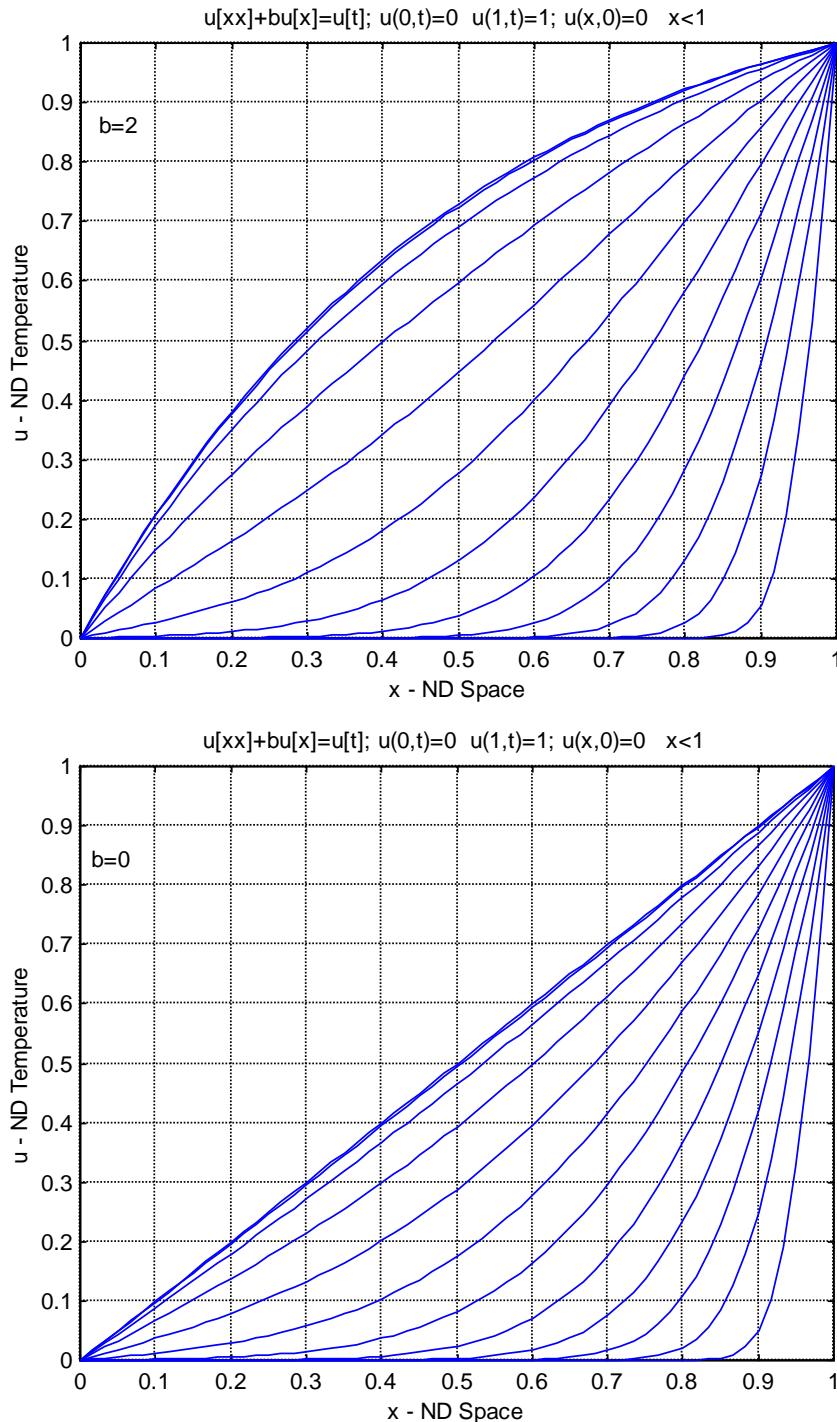
```

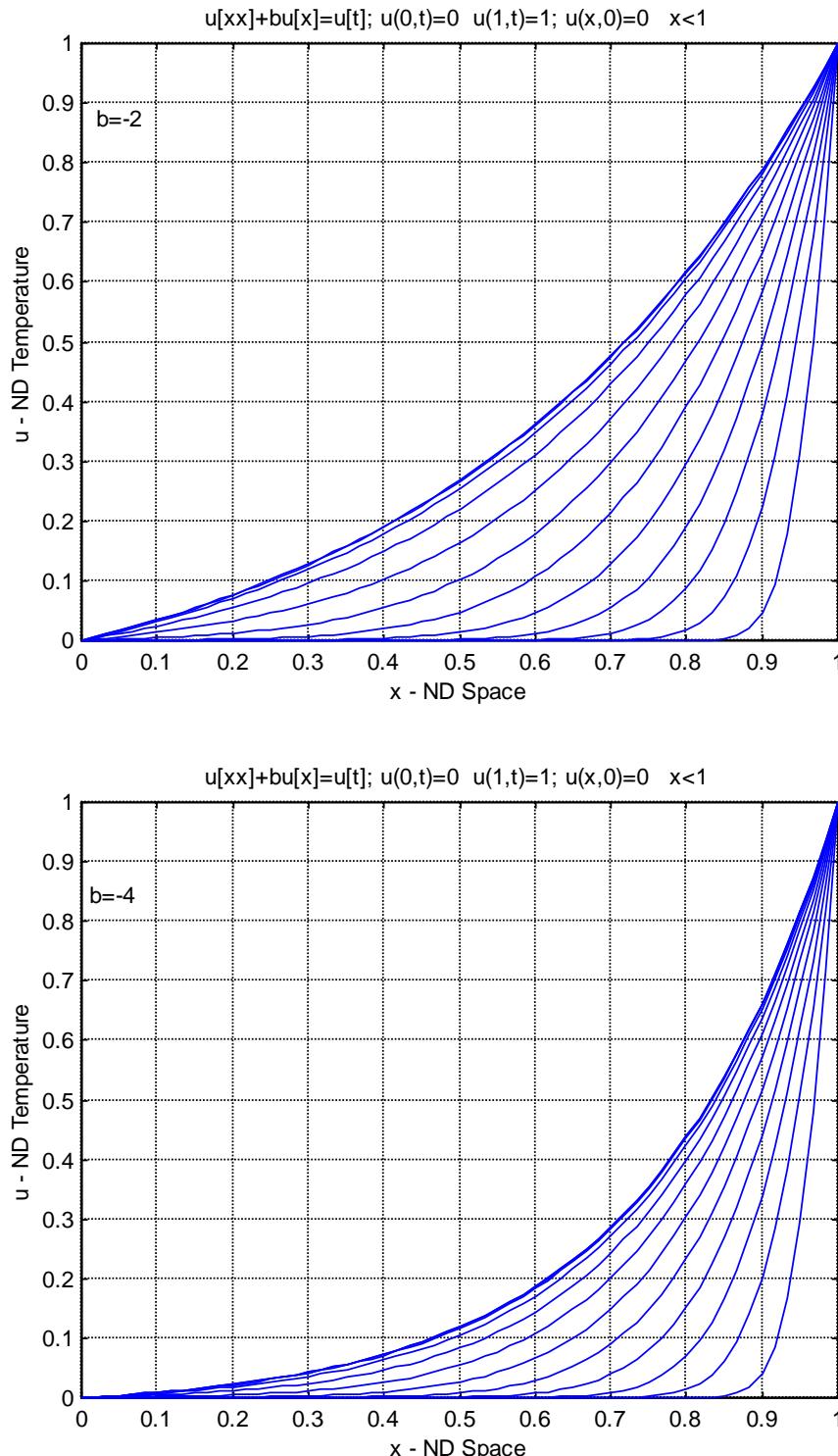
```

% j=time index
% i=speace index
for j=5:5:nmax
    for i=1:imax
        up(i)=u(i,j);
    end
    plot(x,up)
    hold on
end
grid
title('u[xx]+bu[x]=u[t]; u(0,t)=0 u(1,t)=1; u(x,0)=0 x<1')
xlabel('x - ND Space')
ylabel('u - ND Temperature')
hold off
gtext('b=-4')

```







30.16 We will solve this problem with the simple explicit method. Therefore, the interior nodes are handled in a standard fashion as

$$T_i^{l+1} = T_i^l + \lambda(T_{i+1}^l - 2T_i^l + T_{i-1}^l)$$

For the n th-node, the insulated condition can be developed by writing the balance as

$$T_n^{l+1} = T_n^l + \lambda(2T_{n-1}^l - 2T_n^l)$$

Finally, the convective boundary condition at the first node ($i = 0$) can be represented by first writing the general balance as

$$T_0^{l+1} = T_0^l + \lambda(T_1^l - 2T_0^l + T_{-1}^l) \quad (i)$$

This introduces an exterior node into the solution at $i = -1$. The boundary condition can be used to eliminate this node. To do this, a finite difference representation of the condition can be written as

$$-k' \frac{T_1 - T_{-1}}{2\Delta x} = h(T_a - T_0)$$

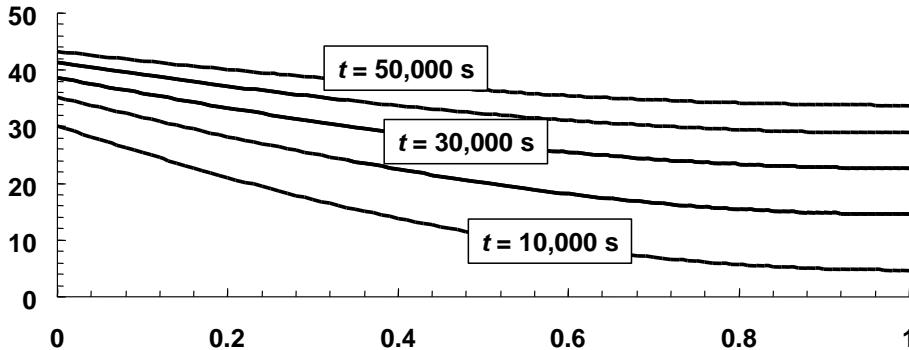
which can be solved for

$$T_{-1} = T_1 + \frac{2\Delta x h}{k'} (T_a - T_0)$$

which can be substituted into Eq. (i) to give

$$T_0^{l+1} = T_0^l + \lambda \left(2T_1^l - 2T_0^l + \frac{2h\Delta x}{k'} (T_a - T_0^l) \right)$$

The entire system can be solved with a step of 1 s. A plot of the results is shown below. After sufficient time, the rod will approach a uniform temperature of 50°C.



CHAPTER 31

31.1 The equation to be solved is

$$\frac{d^2T}{dx^2} = -15$$

Assume a solution of the form $T = ax^2 + bx + c$ which can be differentiated twice to give $T'' = 2a$. Substituting this result into the differential equation gives $a = -7.5$. The boundary conditions can be used to evaluate the remaining coefficients. For the first condition at $x = 0$,

$$75 = -7.5(0)^2 + b(0) + c$$

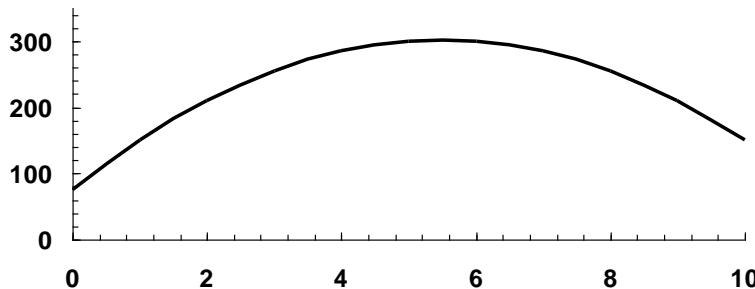
or $c = 75$. Similarly, for the second condition

$$150 = -7.5(10)^2 + b(10) + 75$$

which can be solved for $b = 82.5$. Therefore, the final solution is

$$T = -7.5x^2 + 82.5x + 75$$

The results are plotted as



31.2 The heat source term in the first row of Eq. (31.26) can be evaluated by substituting Eq. (31.3) and integrating to give

$$\int_0^{2.5} 15 \frac{2.5-x}{2.5} dx = 18.75$$

Similarly, Eq. (31.4) can be substituted into the heat source term of the second row of Eq. (31.26), which can also be integrated to yield

$$\int_0^{2.5} 15 \frac{x-0}{2.5} dx = 18.75$$

These results along with the other parameter values can be substituted into Eq. (31.26) to give

$$0.4T_1 - 0.4T_2 = -\frac{dT}{dx}(x_1) + 18.75$$

and

$$-0.4T_1 + 0.4T_2 = \frac{dT}{dx}(x_2) + 18.75$$

31.3 In a manner similar to Fig. 31.7, the equations can be assembled for the total system,

$$\begin{bmatrix} 0.4 & -0.4 & & & \\ -0.4 & 0.8 & -0.4 & & \\ & -0.4 & 0.8 & -0.4 & \\ & & -0.4 & 0.8 & -0.4 \\ & & & -0.4 & 0.4 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{Bmatrix} = \begin{Bmatrix} -dT(x_1)/dx + 18.75 \\ 37.5 \\ 37.5 \\ 37.5 \\ dT(x_5)/dx + 18.75 \end{Bmatrix}$$

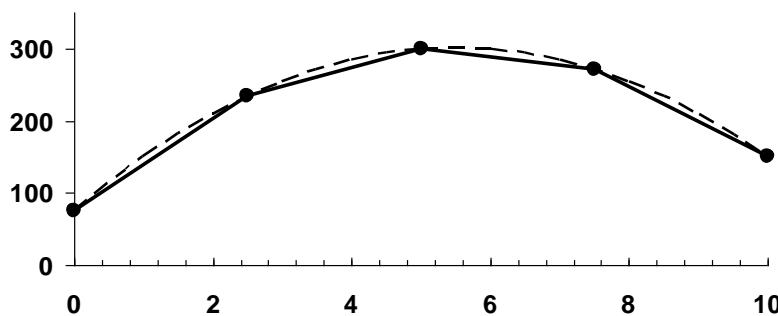
The known end temperatures can be substituted to give

$$\begin{bmatrix} 1 & -0.4 & & & \\ 0.8 & -0.4 & & & \\ -0.4 & 0.8 & -0.4 & & \\ -0.4 & 0.8 & & & \\ -0.4 & 0.4 & & & 1 \end{bmatrix} \begin{Bmatrix} dT(x_1)/dx \\ T_2 \\ T_3 \\ T_4 \\ -dT(x_5)/dx \end{Bmatrix} = \begin{Bmatrix} -11.25 \\ 67.5 \\ 37.5 \\ 97.5 \\ -41.25 \end{Bmatrix}$$

These equations can be solved for

$$\begin{Bmatrix} dT(x_1)/dx \\ T_2 \\ T_3 \\ T_4 \\ -dT(x_5)/dx \end{Bmatrix} = \begin{Bmatrix} 82.5 \\ 234.375 \\ 300 \\ 271.875 \\ -67.5 \end{Bmatrix}$$

The solution, along with the analytical solution (dashed line) is shown below:



31.4

$$0 = D \frac{d^2 c}{dx^2} - U \frac{dc}{dx} - k c$$

$$R = D \frac{d^2 \tilde{c}}{dx^2} - U \frac{d\tilde{c}}{dx} - k \tilde{c}$$

$$\int_{x_1}^{x_2} \left[D \frac{d^2 \tilde{c}}{dx^2} - U \frac{d\tilde{c}}{dx} - k \tilde{c} \right] N_i(x) dx$$

$$D \int_{x_1}^{x_2} \frac{d^2 \tilde{c}}{dx^2} N_i(x) dx \quad (1)$$

$$-U \int_{x_1}^{x_2} \frac{d\tilde{c}}{dx} N_i(x) dx \quad (2)$$

$$-k \int_{x_1}^{x_2} \tilde{c} N_i(x) dx \quad (3)$$

Term (1):

$$D \int_{x_1}^{x_2} \frac{d^2 \tilde{c}}{dx^2} N_i(x) dx = D \left\{ \begin{array}{l} -\frac{dc}{dx}(x_1) - \frac{c_1 - c_2}{x_2 - x_1} \\ \frac{dc}{dx}(x_2) - \frac{c_2 - c_1}{x_2 - x_1} \end{array} \right\}$$

Term (2):

$$\int_{x_1}^{x_2} \frac{d\tilde{c}}{dx} N_i(x) dx = \int_{x_1}^{x_2} \frac{c_2 - c_1}{x_2 - x_1} N_i(x) dx$$

$$\int_{x_1}^{x_2} N_i(x) dx = \frac{x_2 - x_1}{2}$$

$$\therefore \int_{x_1}^{x_2} \frac{d\tilde{c}}{dx} N_i(x) dx = \frac{c_2 - c_1}{2}$$

$$-U \int_{x_1}^{x_2} \frac{d\tilde{c}}{dx} N_i(x) dx = -U \left\{ \begin{array}{l} \frac{c_2 - c_1}{2} \\ \frac{c_2 - c_1}{2} \end{array} \right\}$$

Term (3):

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

$$-k \int_{x_1}^{x_2} \tilde{c} N_i(x) dx = -\frac{k(x_2 - x_1)}{2} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix}$$

Total element equation [(1) + (2) + (3)]

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

where

$$a_{11} = \frac{D}{x_2 - x_1} - \frac{U}{2} + \frac{k}{2}(x_2 - x_1) \quad a_{12} = \frac{-D}{x_2 - x_1} + \frac{U}{2} \quad a_{21} = \frac{-D}{x_2 - x_1} - \frac{U}{2}$$

$$a_{22} = \frac{D}{x_2 - x_1} + \frac{U}{2} + \frac{k}{2}(x_2 - x_1)$$

$$b_1 = -D \frac{dc}{dx}(x_1) \quad b_2 = D \frac{dc}{dx}(x_2)$$

31.5 First we can develop an analytical function for comparison. Substituting parameters gives

$$0.1(200 \times 10^9) \frac{d^2u}{dx^2} = 1000$$

Assume a solution of the form

$$u = ax^2 + bx + c$$

This can be differentiated twice to yield $d^2u/dx^2 = 2a$. This can be substituted into the ODE, which can then be solved for $a = 2.5 \times 10^{-8}$. The boundary conditions can then be used to evaluate the remaining coefficients. At the left side, $u(0) = 0$ and

$$0 = 2.5 \times 10^{-8}(0)^2 + b(0) + c$$

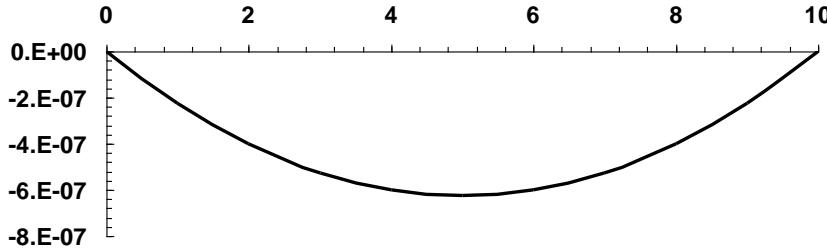
and therefore, $c = 0$. At the right side of the bar, $u(10) = 0$ and

$$0 = 2.5 \times 10^{-8}(10)^2 + b(10)$$

and therefore, $b = -2.5 \times 10^{-7}$, and the solution is

$$u = 2.5 \times 10^{-8} x^2 - 2.5 \times 10^{-7} x$$

which can be displayed as



The element equation can be written as

$$\frac{A_c E}{x_2 - x_1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = A_c E \begin{Bmatrix} -\frac{du}{dx}(x_1) \\ \frac{du}{dx}(x_2) \end{Bmatrix} + \begin{Bmatrix} \int_{x_1}^{x_2} P(x) N_1(x) dx \\ \int_{x_1}^{x_2} P(x) N_2(x) dx \end{Bmatrix}$$

The distributed load can be evaluated as

$$\int_0^2 -1000 \frac{2-x}{2} dx = -1000 \quad \int_0^2 -1000 \frac{x-0}{2} dx = -1000$$

Thus, the final element equation is

$$\begin{bmatrix} 1 \times 10^{10} & -1 \times 10^{10} \\ -1 \times 10^{10} & 1 \times 10^{10} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = A_c E \begin{Bmatrix} -\frac{du}{dx}(x_1) \\ \frac{du}{dx}(x_2) \end{Bmatrix} + \begin{Bmatrix} -1000 \\ -1000 \end{Bmatrix}$$

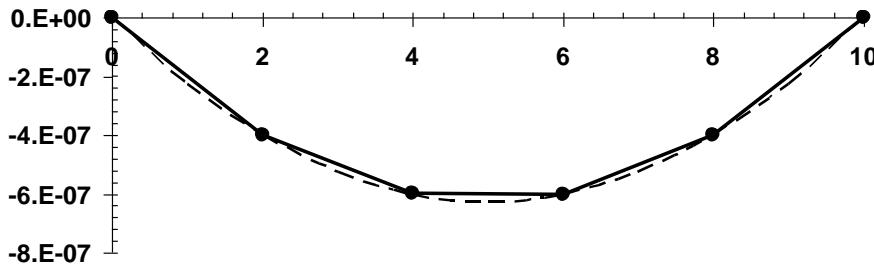
Assembly yields

$$\begin{bmatrix} 2 \times 10^{10} & -1 \times 10^{10} & & & \\ 2 \times 10^{10} & -1 \times 10^{10} & & & \\ -1 \times 10^{10} & 2 \times 10^{10} & -1 \times 10^{10} & & \\ & -1 \times 10^{10} & 2 \times 10^{10} & -1 \times 10^{10} & \\ & & -1 \times 10^{10} & 2 \times 10^{10} & -1 \times 10^{10} \\ & & & -1 \times 10^{10} & -2 \times 10^{10} \end{bmatrix} \begin{Bmatrix} \frac{du}{dx}(x_1) \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \frac{du}{dx}(x_2) \end{Bmatrix} = \begin{Bmatrix} -1000 \\ -2000 \\ -2000 \\ -2000 \\ -2000 \\ -1000 \end{Bmatrix}$$

which can be solved for

$$\begin{Bmatrix} \frac{du}{dx}(x_1) \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \frac{du}{dx}(x_2) \end{Bmatrix} = \begin{Bmatrix} -2.5 \times 10^{-7} \\ -4 \times 10^{-7} \\ -6 \times 10^{-7} \\ -6 \times 10^{-7} \\ -4 \times 10^{-7} \\ 2.5 \times 10^{-7} \end{Bmatrix}$$

These results, along with the analytical solution (dashed line) are displayed below:



31.6 Here is a VBA program to solve for the steady state temperature distribution for a heated rod with a constant heat source. It is set up to solve the system described in Examples 31.1 and 31.2.

```

Option Explicit

Sub FErod()
    Dim ns As Integer, ii As Integer, i As Integer, j As Integer
    Dim k As Integer, m As Integer
    Dim x(100) As Double, st(2, 2) As Double, c As Double
    Dim s(2, 2) As Double, a(100, 100) As Double, b(100) As Double, d(100) As Double
    Dim Te(100) As Double, ff As Double
    Dim e(100) As Double, f(100) As Double, g(100) As Double, r(100) As Double
    Dim dum1 As Double, dum2 As Double
    Dim dTeLeft As Double, dTeRight As Double
    'set parameters
    ns = 4
    x(1) = 0
    x(2) = 2.5
    x(3) = 5
    x(4) = 7.5
    x(5) = 10
    Te(1) = 40
    Te(5) = 200
    ff = 10
    'construct system matrix
    st(1, 1) = 1: st(1, 2) = -1: st(2, 1) = -1: st(2, 2) = 1
    For ii = 1 To ns
        c = 1 / (x(ii + 1) - x(ii))
        For i = 1 To 2
            For j = 1 To 2
                s(i, j) = c * st(i, j)
            Next j
        Next i
    Next ii
    'apply boundary conditions
    a(1, 1) = 1: a(1, 2) = 0: a(2, 1) = 0: a(2, 2) = 1
    b(1) = ff: b(2) = 0
    'solve system
    For i = 1 To ns
        r(i) = b(i)
        For j = 1 To ns
            If i = j Then
                r(i) = r(i) / s(i, i)
            Else
                r(i) = r(i) - s(i, j) * r(j)
            End If
        Next j
    Next i
    'output results
    For i = 1 To ns
        Te(i) = r(i)
    Next i
End Sub

```

```

    Next j
    Next i
    For i = 1 To 2
        k = ii - 1 + i
        For j = 1 To 2
            m = ii - 1 + j
            a(k, m) = a(k, m) + s(i, j)
        Next j
        b(k) = b(k) + ff * ((x(ii + 1) - x(ii)) - (x(ii + 1) - x(ii)) / 2)
    Next i
    Next ii
    'determine impact of uniform source and boundary conditions
    Call Mmult(a, Te, d, ns + 1, ns + 1, 1)
    For i = 1 To ns + 1
        b(i) = b(i) - d(i)
    Next i
    a(1, 1) = 1
    a(2, 1) = 0
    a(ns + 1, ns + 1) = -1
    a(ns, ns + 1) = 0
    'Transform square matrix into tridiagonal form
    f(1) = a(1, 1)
    g(1) = a(1, 2)
    r(1) = b(1)
    For i = 2 To ns
        e(i) = a(i, i - 1)
        f(i) = a(i, i)
        g(i) = a(i, i + 1)
        r(i) = b(i)
    Next i
    e(ns + 1) = a(ns + 1, ns)
    f(ns + 1) = a(ns + 1, ns + 1)
    r(ns + 1) = b(ns + 1)
    'Tridiagonal solver
    dum1 = Te(1)
    dum2 = Te(ns + 1)
    Call Tridiag(e, f, g, r, ns + 1, Te)
    dTeLeft = Te(1)
    dTeRight = Te(ns + 1)
    Te(1) = dum1
    Te(ns + 1) = dum2
    'output results
    Sheets("sheet1").Select
    Range("a7:b5000").ClearContents
    Range("a3").Select
    ActiveCell.Value = "dTe(x = " & x(0) & ")/dx = "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = dTeLeft
    ActiveCell.Offset(1, -1).Select
    ActiveCell.Value = "dTe(x = " & x(ns + 1) & ")/dx = "
    ActiveCell.Offset(0, 1).Select
    ActiveCell.Value = dTeRight
    ActiveCell.Offset(3, -1).Select
    For i = 1 To ns + 1
        ActiveCell.Value = x(i)
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value = Te(i)
        ActiveCell.Offset(1, -1).Select
    Next i
    Range("b3").Select
End Sub

```

```

Sub Mmult(a, b, c, m, n, l)
Dim i As Integer, j As Integer, k As Integer
Dim sum As Double
For i = 1 To n
    sum = 0
    For k = 1 To m
        sum = sum + a(i, k) * b(k)
    Next k
    c(i) = sum
Next i
End Sub

Sub Tridiag(e, f, g, r, n, x)
Dim k As Integer
'decompose
For k = 2 To n
    e(k) = e(k) / f(k - 1)
    f(k) = f(k) - e(k) * g(k - 1)
Next k
'substitute
For k = 2 To n
    r(k) = r(k) - e(k) * r(k - 1)
Next k
x(n) = r(n) / f(n)
For k = n - 1 To 1 Step -1
    x(k) = (r(k) - g(k) * x(k + 1)) / f(k)
Next k
End Sub

```

The output is

	A	B	C	D	E
1	Prob 31.6				
2					
3	dTe(x = 0)/dx =	66			
4	dTe(x = 10)/dx =	-34		RUN	
5					
6	x	Te			
7	0	40			
8	2.5	173.75			
9	5	245			
10	7.5	253.75			
11	10	200			

31.7 After setting up the original spreadsheet, the following modifications would be made to insulate the right edge and add the sink:

Cell I1: Set to 100

Cell I2: = (I1+2*H2+I3) / 4; This formula would then be copied to cells I3:I8.

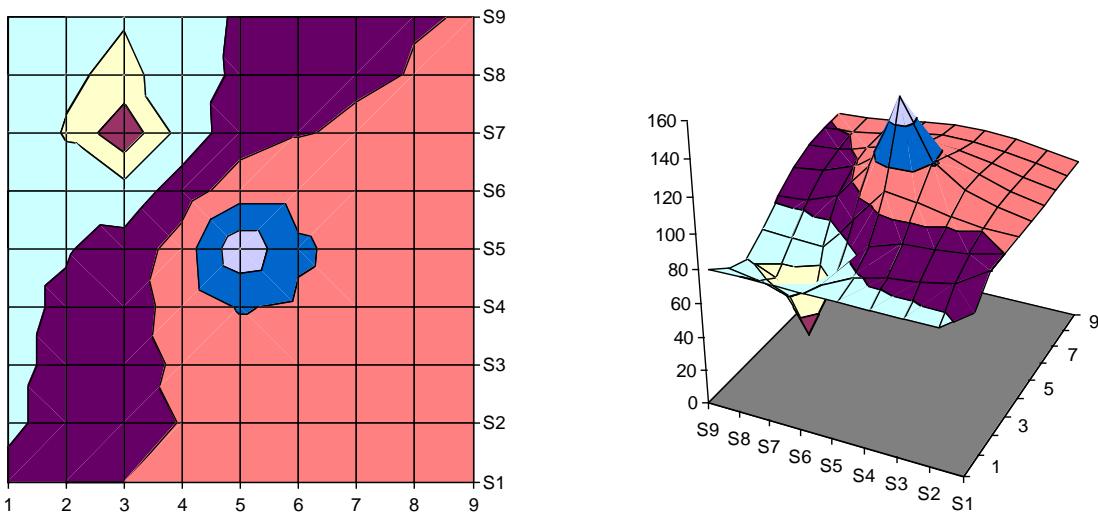
Cell I9: = (I8+H9) / 2

Cell C7: = (C6+D7+C8+B7-150) / 4

The resulting spreadsheet is displayed below:

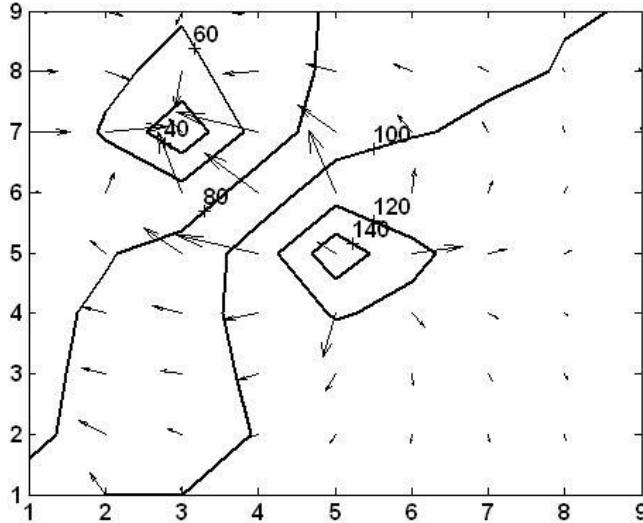
	A	B	C	D	E	F	G	H	I
1		100	100	100	100	100	100	100	100
2	75	89.0	95.9	100.5	103.5	104.2	103.8	103.4	103.2
3	75	85.2	94.1	102.5	109.2	109.4	107.8	106.5	106.0
4	75	82.9	92.6	106.4	121.4	116.4	111.5	108.7	107.9
5	75	78.7	87.0	109.0	153.7	123.3	113.0	109.1	108.1
6	75	69.9	67.7	89.0	110.9	110.3	108.0	106.6	106.2
7	75	58.3	24.9	68.5	90.6	99.0	102.1	103.1	103.4
8	75	63.5	54.9	69.5	83.9	93.1	98.1	100.5	101.2
9	75	65.9	61.6	70.8	82.5	91.4	96.8	99.5	100.4

Corresponding contour plots can be generated as



31.8 The results of the preceding problem can be saved as a tab-delimited text file (in our case, we called the file prob3108.txt). The following commands can then be used to load this file into MATLAB, as well as to generate the contour plot along with heat flow vectors.

```
>> load prob3108.txt
>> [px,py]=gradient(prob3108);
>> cs=contour(prob3108);clabel(cs);hold on
>> quiver(-px,-py);hold off
```



31.9 The scheme for implementing this calculation on Excel is shown below:

	A	B	C	D	E	F	G	H	I	J	K
1	87.5	100	100	100	100	100	100	100	100	100	62.5
2	75										25
3	75										25
4	75										25
5	75										25
6	62.5	50	50	50	50	50	50	50	50	50	37.5

The simple Laplace equation applies directly to the blank white cells (recall Fig. 31.14). However, for the shaded cells impacted by the heat sink, a heat balance equation must be written. For example, for cell E3, the control volume approach can be developed as

$$0 = -k' \frac{E3 - D3}{\Delta x} \Delta y \Delta z + k' \frac{F3 - E3}{\Delta x} \Delta y \Delta z - k' \frac{E3 - E4}{\Delta y} \Delta x \Delta z + k' \frac{E2 - E3}{\Delta y} \Delta x \Delta z - 100 \Delta x \Delta y$$

Collecting and canceling terms yields

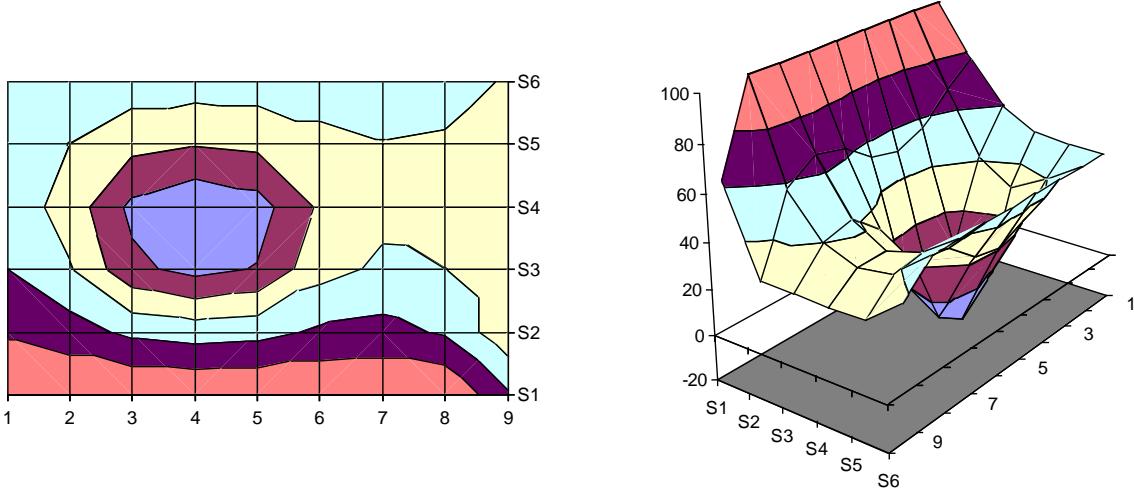
$$0 = -4E3 + D3 + F3 + E4 + E2 - 100 \frac{\Delta x \Delta y}{\Delta z k'}$$

Substituting the length dimensions and the coefficient of thermal conductivity gives,

$$E3 = \frac{D3 + F3 + E4 + E2 - 66.6667}{4}$$

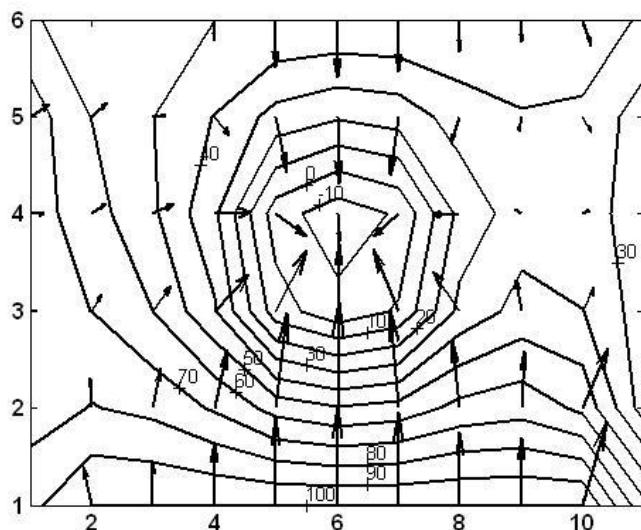
This formula can be copied to the other shaded cells. The result is depicted below, along with the corresponding contour plots.

	A	B	C	D	E	F	G	H	I	J	K
1	87.5	100	100	100	100	100	100	100	100	100	62.5
2	75	80.56	77.32	68.69	56.08	50.83	53.80	63.20	66.15	57.83	25
3	75	69.92	60.04	41.35	4.80	-6.58	1.20	32.83	43.56	40.19	25
4	75	64.08	51.57	31.89	-5.01	-16.46	-8.60	23.37	35.09	34.35	25
5	75	59.84	50.26	39.64	26.42	21.03	24.14	34.15	39.09	37.11	25
6	62.5	50	50	50	50	50	50	50	50	50	37.5



31.10 The results of the preceding problem can be saved as a tab-delimited text file (in our case, we called the file prob3110.txt). The following commands can then be used to load this file into MATLAB, as well as to generate the contour plot along with heat flow vectors.

```
>> load prob3110.txt
>> [px,py]=gradient(prob3110);
>> cs=contour(prob3110);clabel(cs);hold on
>> quiver(-px,-py);hold off
```



31.11

```

Program Plate
Use IMSL
Implicit None
Integer::ncval, nx, nxtabl, ny, nytabl
Parameter (ncval=11, nx=33, nxtabl=5, ny=33, nytabl=5)
Integer::i, ibcty(4), iorder, j, nout
Real::ax,ay,brhs,bx,by,coefu,prhs,u(nx,ny),utabl,x,xdata(nx),y,ydata(ny)
External brhs, prhs
ax = 0
bx = 40
ay = 0
by = 40
ibcty(1) = 1
ibcty(2) = 2
ibcty(3) = 1
ibcty(4) = 1
coefu = 0
iorder = 4
Call FPS2H(prhs, brhs, coefu, nx, ny, ax, bx, ay, by, ibcty, iorder, u, nx)
Do i=1, nx
    xdata(i) = ax + (bx - ax) * Float(i - 1) / Float(nx - 1)
End Do
Do j=1, ny
    ydata(j) = ay + (by - ay) * Float(j - 1) / Float(ny - 1)
End Do
Call UMACH(2, nout)
Write (nout,'(8X,A,11X,A,11X,A)') 'X', 'Y', 'U'
Do j=1, nytabl
    Do i=1, nxtabl
        x = ax + (bx - ax) * Float(i - 1) / Float(nxtabl - 1)
        y = ay + (by - ay) * Float(j - 1) / Float(nytabl - 1)
        utabl = QD2VL(x,y,nx,xdata,ny,ydata,u,nx,.FALSE.)
        Write (nout,'(4F12.4)') x, y, utabl
    End Do
End Do
End Program

Function prhs(x, y)
Implicit None
Real::prhs, x, y
prhs = 0
End Function

Real Function brhs(iside, x, y)
Implicit None
Integer::iside
Real::x, y
If (iside == 1) Then
    brhs = 50
ElseIf (iside == 2) Then
    brhs = 0
ElseIf (iside == 3) Then
    brhs = 75
Else
    brhs = 100
End If
End Function

```

Output:

0.0000	0.0000	75.0000
10.0000	0.0000	71.6339
20.0000	0.0000	66.6152
30.0000	0.0000	59.1933
40.0000	0.0000	50.0000
0.0000	10.0000	75.0000
10.0000	10.0000	72.5423
20.0000	10.0000	67.9412
30.0000	10.0000	60.1914
40.0000	10.0000	50.0000
0.0000	20.0000	75.0000
10.0000	20.0000	75.8115
20.0000	20.0000	72.6947
30.0000	20.0000	64.0001
40.0000	20.0000	50.0000
0.0000	30.0000	75.0000
10.0000	30.0000	83.5385
20.0000	30.0000	83.0789
30.0000	30.0000	74.3008
40.0000	30.0000	50.0000
0.0000	40.0000	87.5000
10.0000	40.0000	100.0000
20.0000	40.0000	100.0000
30.0000	40.0000	100.0000
40.0000	40.0000	75.0000

Press any key to continue

31.12

Element 1:

Node 1:

$$\sum q_k + f(x) = 0$$

$$\left(-kA \frac{dT}{dx} \Big|_1 + kA \left(\frac{1}{x_2 - x_1} \right) (T_2 - T_1) \right) + \int_{x_1}^{x_2} N_1 f(x) dx$$

$$\left(-100 \frac{dT}{dx} \Big|_1 + \frac{100}{10} (T_2 - T_1) \right) + \int_0^{10} \left(\frac{10-x}{10} \right) 30 dx$$

$$10(T_1 - T_2) = -100 \frac{dT}{dx} \Big|_1 + 150$$

Node 2:

$$\left(kA \frac{dT}{dx} \Big|_2 - kA \left(\frac{1}{x_2 - x_1} \right) (T_2 - T_1) \right) + \int_{x_1}^{x_2} N_2 f(x) dx$$

$$\left(100 \frac{dT}{dx} \Big|_1 + \frac{100}{10} (T_2 - T_1) \right) + \int_0^{10} \left(\frac{x-0}{10} \right) 30 dx$$

$$-10(T_1 - T_2) = 100 \frac{dT}{dx} \Big|_2 + 150$$

The other node equations can be derived similarly,

Element 2:

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

Node 2:

$$10(T_2 - T_3) = -100 \frac{dT}{dx} \Big|_2 + 150$$

Node 3:

$$-10(T_2 - T_3) = 100 \frac{dT}{dx} \Big|_3 + 150$$

The other element equations are similar.

Equation assembly:

$$\begin{bmatrix} 10 & -10 & & & & \\ -10 & 20 & -10 & & & \\ & -10 & 20 & -10 & & \\ & & -10 & 20 & -10 & \\ & & & -10 & 20 & -10 \\ & & & & -10 & 10 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{Bmatrix} = \begin{Bmatrix} -100 \frac{dT}{dx} \Big|_1 + 150 \\ 300 \\ 300 \\ 300 \\ 300 \\ 100 \frac{dT}{dx} \Big|_6 + 150 \end{Bmatrix}$$

Inserting boundary conditions:

$$\begin{bmatrix} 10 & -10 & & & & \\ -10 & 20 & -10 & & & \\ & -10 & 20 & -10 & & \\ & & -10 & 20 & -10 & \\ & & & -10 & 20 & \\ & & & & -10 & -100 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ \frac{dT}{dx} \Big|_6 \end{Bmatrix} = \begin{Bmatrix} 125 \\ 300 \\ 300 \\ 300 \\ 1300 \\ -850 \end{Bmatrix}$$

The solution can then be obtained with MATLAB:

```
>> C=[10 -10 0 0 0 0;
-10 20 -10 0 0 0;
0 -10 20 -10 0 0;
0 0 -10 20 -10 0;
0 0 0 -10 20 0;
0 0 0 -10 -100];
>> b=[125 300 300 300 1300 -850]';
>> x=C\b

x =
462.5000
450.0000
407.5000
335.0000
232.5000
```

-14.7500

31.13

Element 1:

Node 1:

$$\sum q_k + f(x) = 0$$

$$\left(-kA \frac{dT}{dx} \Big|_1 + kA \left(\frac{1}{x_2 - x_1} \right) (T_2 - T_1) \right) + \int_{x_1}^{x_2} N_1 f(x) dx$$

$$\left(-100 \frac{dT}{dx} \Big|_1 + \frac{95}{10} (T_2 - T_1) \right) + \int_0^{10} \left(\frac{10-x}{10} \right) 30 dx$$

$$9.5(T_1 - T_2) = -100 \frac{dT}{dx} \Big|_1 + 150$$

Node 2:

$$\left(kA \frac{dT}{dx} \Big|_2 - kA \left(\frac{1}{x_2 - x_1} \right) (T_2 - T_1) \right) + \int_{x_1}^{x_2} N_2 f(x) dx$$

$$\left(90 \frac{dT}{dx} \Big|_1 + \frac{95}{10} (T_2 - T_1) \right) + \int_0^{10} \left(\frac{x-0}{10} \right) 30 dx$$

$$-9.5(T_1 - T_2) = 90 \frac{dT}{dx} \Big|_2 + 150$$

The other node equations can be derived similarly,

Element 2:

Node 2:

$$8.5(T_2 - T_3) = -90 \frac{dT}{dx} \Big|_2 + 150$$

Node 3:

$$-8.5(T_2 - T_3) = 80 \frac{dT}{dx} \Big|_3 + 150$$

Element 3:

Node 3:

$$7.5(T_3 - T_4) = -80 \frac{dT}{dx} \Big|_3 + 150$$

Node 4:

$$-7.5(T_3 - T_4) = 70 \frac{dT}{dx} \Big|_4 + 150$$

Element 4:

Node 4:

$$6.5(T_4 - T_5) = -70 \frac{dT}{dx} \Big|_4 + 150$$

Node 5:

$$-6.5(T_4 - T_5) = 60 \frac{dT}{dx} \Big|_5 + 150$$

Element 5:

Node 5:

$$5.5(T_5 - T_6) = -60 \frac{dT}{dx} \Big|_5 + 150$$

Node 6:

$$-5.5(T_5 - T_6) = 50 \frac{dT}{dx} \Big|_6 + 150$$

Equation assembly:

$$\begin{bmatrix} 9.5 & -9.5 & 0 & 0 & 0 & 0 \\ -9.5 & 18 & -8.5 & 0 & 0 & 0 \\ 0 & -8.5 & 16 & -7.5 & 0 & 0 \\ 0 & 0 & -7.5 & 14 & -6.5 & 0 \\ 0 & 0 & 0 & -6.5 & 12 & 0 \\ 0 & 0 & 0 & 0 & -5.5 & -50 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{Bmatrix} = \begin{Bmatrix} -100 \frac{dT}{dx} \Big|_1 + 150 \\ 300 \\ 300 \\ 300 \\ 300 \\ 50 \frac{dT}{dx} \Big|_6 + 150 \end{Bmatrix}$$

Inserting boundary conditions:

$$\begin{bmatrix} 100 & -9.5 & 0 & 0 & 0 & 0 \\ 18 & -8.5 & 0 & 0 & 0 & 0 \\ -8.5 & 16 & -7.5 & 0 & 0 & 0 \\ -7.5 & 14 & -6.5 & 0 & 0 & 0 \\ -6.5 & 12 & 0 & 0 & 0 & 0 \\ -5.5 & -50 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \frac{dT}{dx} \Big|_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ \frac{dT}{dx} \Big|_6 \end{Bmatrix} = \begin{Bmatrix} -800 \\ 1250 \\ 300 \\ 300 \\ 575 \\ -125 \end{Bmatrix}$$

The solution can then be obtained with MATLAB:

```
>> C=[100 -9.5 0 0 0 0;
0 18 -8.5 0 0 0;
0 -8.5 16 -7.5 0 0;
0 0 -7.5 14 -6.5 0;
0 0 0 -6.5 12 0;
0 0 0 0 -5.5 -50];
>> b=[-800 1250 300 300 575 -125]';
>> x=C\b
```

x =

7.5982
164.1913
200.6404
201.9494
157.3059
-14.8037

CHAPTER 32

32.1 First equation

$$6.075c_0 - 3.2c_1 = 262.5$$

Middle equations ($i = 1$ to 8)

$$-2.1c_{i-1} + 3.45c_i - 1.1c_{i+1} = 0$$

Last equation

$$-3.2c_8 + 3.45c_9 = 0$$

The solution is

$c_0 = 76.53$	$c_5 = 29.61$
$c_1 = 63.25$	$c_6 = 24.62$
$c_2 = 52.28$	$c_7 = 20.69$
$c_3 = 43.22$	$c_8 = 17.88$
$c_4 = 35.75$	$c_9 = 16.58$

32.2 Element equation: (See solution for Prob. 31.4 for derivation of element equation.)

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

where

$$a_{11} = \frac{D}{x_2 - x_1} - \frac{U}{2} + \frac{k}{2}(x_2 - x_1) \quad a_{12} = \frac{-D}{x_2 - x_1} + \frac{U}{2} \quad a_{21} = \frac{-D}{x_2 - x_1} - \frac{U}{2}$$

$$a_{22} = \frac{D}{x_2 - x_1} + \frac{U}{2} + \frac{k}{2}(x_2 - x_1)$$

$$b_1 = -D \frac{dc}{dx}(x_1) \quad b_2 = D \frac{dc}{dx}(x_2)$$

Substituting the parameter values:

$$a_{11} = \frac{2}{2.5} - \frac{1}{2} + \frac{0.2}{2} (2.5) = 0.55 \quad a_{12} = \frac{-2}{2.5} + \frac{1}{2} = -0.3$$

$$a_{21} = \frac{-2}{2.5} - \frac{1}{2} = -1.3 \quad a_{22} = \frac{2}{2.5} + \frac{1}{2} + \frac{0.2}{2} (2.5) = 1.55$$

$$b_1 = -2 \frac{dc}{dx}(x_1) \quad b_2 = 2 \frac{dc}{dx}(x_2)$$

Assembly:

$$\begin{bmatrix} 0.55 & -0.3 & & & \\ -1.3 & 2.1 & -0.3 & & \\ & -1.3 & 2.1 & -0.3 & \\ & & -1.3 & 2.1 & -0.3 \\ & & & -1.3 & 1.55 \end{bmatrix} \begin{Bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} -2 \frac{dc}{dx}(x_1) \\ 0 \\ 0 \\ 0 \\ 2 \frac{dc}{dx}(x_2) \end{Bmatrix}$$

Boundary conditions:

A mass balance at the inlet can be written as:

$$Uc_{\text{in}} = Uc_0 - D \frac{dc}{dx}(0)$$

which can be solved for

$$-D \frac{dc}{dx}(0) = Uc_{\text{in}} - Uc_0$$

Substitute into first equation

$$0.55c_0 - 0.3c_1 = 100 - c_0$$

$$1.55c_0 - 0.3c_1 = 100$$

Outlet:

$$\frac{dc}{dx}(10) = 0$$

Therefore, the system of equations to be solved is

$$\begin{bmatrix} 1.55 & -0.3 & & & \\ -1.3 & 2.1 & -0.3 & & \\ & -1.3 & 2.1 & -0.3 & \\ & & -1.3 & 2.1 & -0.3 \\ & & & -1.3 & 1.55 \end{bmatrix} \begin{Bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 100 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

Solution:

$$c_0 = 74.4 \quad c_1 = 51.08 \quad c_2 = 35.15 \quad c_3 = 24.72 \quad c_4 = 20.74$$

32.3 According to Fick's first law, the diffusive flux is

$$J(x) = -D \frac{dc}{dx}(x)$$

where $J(x)$ = flux at position x . If c has units of g/m^3 , D has units of m^2/d and x is measured in m, flux has units of $\text{g}/\text{m}^2/\text{d}$. In addition, there will be an advective flux which can be calculated as

$$J(x) = Uc(x)$$

Finite divided differences can be used to approximate the derivatives. For example, for the point at the beginning of the tank, a forward difference can be used to calculate

$$\frac{dc}{dx}(0) \approx \frac{52.47 - 76.44}{2.5} = -0.9588 \frac{\text{g}/\text{m}^3}{\text{m}}$$

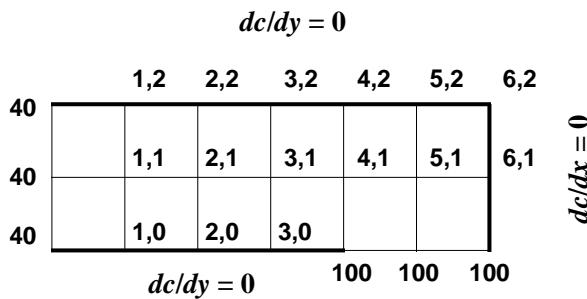
Thus, the flux at the head of the tank is

$$J(0) = -2(-0.9588) + 1(76.44) = 19.176 + 76.44 = 95.616 \frac{\text{g}/\text{m}^3}{\text{m}}$$

The remainder of the values can be calculated in a similar fashion using centered (middle nodes) and backward differences (the end node):

<i>c</i>	<i>dc/dx</i>	<i>J-diff</i>	<i>J-adv</i>	<i>J</i>
76.44	-9.588	19.176	76.44	95.616
52.47	-8.076	16.152	52.47	68.622
36.06	-5.484	10.968	36.06	47.028
25.05	-3.394	6.788	25.05	31.838
19.09	-2.384	4.768	19.09	23.858

32.4 Segmentation scheme:



Nodes 1,1 through 5,1

$$0.5 \frac{c_{i+1,j} - 2c_{i,j} + c_{i-1,j}}{5^2} + 0.5 \frac{c_{i,j+1} - 2c_{i,j} + c_{i,j-1}}{5^2} - 0.1c_{i,j}$$

Collecting terms gives

$$0.18c_{i,j} - 0.02c_{i+1,j} - 0.02c_{i-1,j} - 0.02c_{i,j+1} - 0.02c_{i,j-1} = 0$$

Node 6,1 would be modified to reflect the no flow condition in x and the Dirichlet condition at 6,0:

$$0.18c_{6,1} - 0.04c_{5,1} - 0.02c_{6,2} - 0.02(100) = 0$$

The nodes along the upper edge (1,2 through 5,2) would be generally written to reflect the no-flow condition in y as

$$0.18c_{i,j} - 0.02c_{i+1,j} - 0.02c_{i-1,j} - 0.04c_{i,j-1} = 0$$

The node at the upper right edge (6,2) would be generally written to reflect the no-flow condition in x and y as

$$0.18c_{6,2} - 0.04c_{5,2} - 0.04c_{6,1} = 0$$

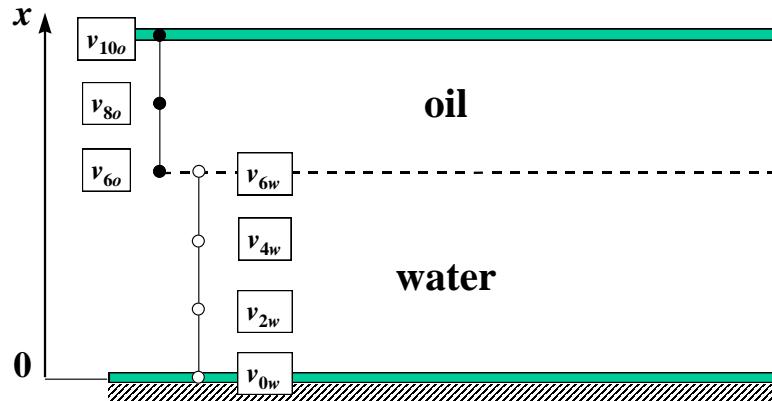
Finally, the nodes along the lower edge (1,0 through 3,0) would be generally written to reflect the no-flow condition in y as

$$0.18c_{i,j} - 0.02c_{i+1,j} - 0.02c_{i-1,j} - 0.04c_{i,j+1} = 0$$

These equations can be solved for

40	5.881926	1.096415	1.216395	3.59368	4.13942	4.220145
40	5.920461	1.384706	3.128728	13.49365	14.72048	14.85123
40	6.017519	2.316748	12.0638	100	100	100

32.5 For simplicity, we will use a very coarse grid to solve this problem. Thus, we place nodes as in the following diagram.



A simple explicit solution can be developed by substituting finite-differences for the second derivative terms in the motion equations. This is done for the three non-boundary nodes,

$$\frac{dv_{2w}}{dt} = \mu_w \frac{v_{0w} - 2v_{2w} + v_{4w}}{\Delta x^2}$$

$$\frac{dv_{4w}}{dt} = \mu_w \frac{v_{2w} - 2v_{4w} + v_{6w}}{\Delta x^2}$$

$$\frac{dv_{8o}}{dt} = \mu_o \frac{v_{6o} - 2v_{8o} + v_{10o}}{\Delta x^2}$$

These three equations have 7 unknowns ($v_{0w}, v_{2w}, v_{4w}, v_{6w}, v_{6o}, v_{8o}, v_{10o}$). The boundary conditions at the plates effectively specify $v_{0w} = 0$ and $v_{10o} = 8$. The former is called a “no slip” condition because it specifies that the velocity at the lower plate is zero.

The relationships at the oil-water interface can be used to eliminate two of the remaining unknowns. The first condition states that

$$v_{6o} = v_{6w} \quad (i)$$

The second can be rearranged to yield

$$v_{6w} = \frac{\mu_o v_{8o} + \mu_w v_{4w}}{\mu_o + \mu_w} \quad (ii)$$

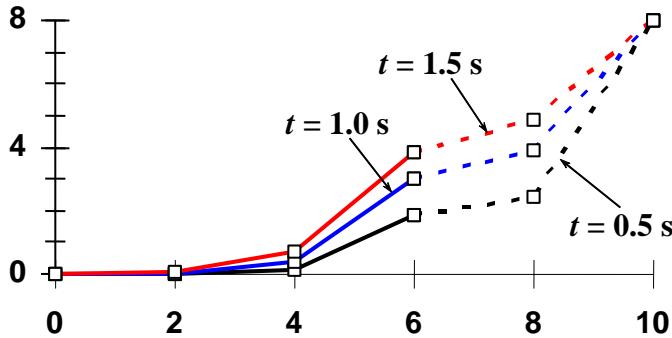
These, along with the wall boundary conditions can be substituted into the differential equations

$$\frac{dv_{2w}}{dt} = \mu_w \frac{-2v_{2w} + v_{4w}}{\Delta x^2}$$

$$\frac{dv_{4w}}{dt} = \mu_w \frac{v_{2w} - 2v_{4w} + \frac{\mu_o v_{8o} + \mu_w v_{4w}}{\mu_o + \mu_w}}{\Delta x^2}$$

$$\frac{dv_{8o}}{dt} = \mu_o \frac{\frac{\mu_o v_{8o} + \mu_w v_{4w}}{\mu_o + \mu_w} - 2v_{8o} + 8}{\Delta x^2}$$

These equations can now be integrated to determine the velocities as a function of time. Equations (i) and (ii) can be used to determine v_{6o} and v_{6w} . The results are plotted below:



32.6 Using a similar approach to Sec. 32.2, the nodal equation can be developed as:

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = -0.29353$$

This equation can then be written for the interior nodes and solved for

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.24461 & -0.34245 & -0.34245 & -0.24461 & 0 \\ 0 & -0.34245 & -0.48922 & -0.48922 & -0.34245 & 0 \\ 0 & -0.34245 & -0.48922 & -0.48922 & -0.34245 & 0 \\ 0 & -0.24461 & -0.34245 & -0.34245 & -0.24461 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

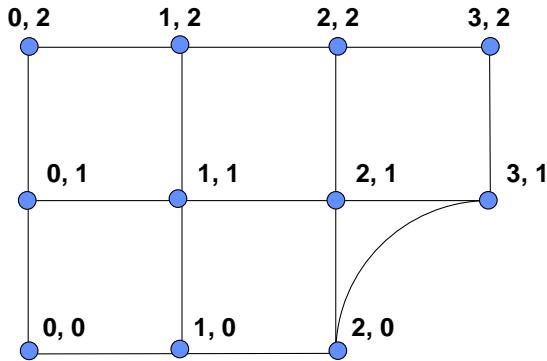
These results can then be used as input to the right-hand side of Eq. 32.14. The nodal equation is

$$4z_{i,j} - z_{i-1,j} - z_{i+1,j} - z_{i,j-1} - z_{i,j+1} = u_{i,j} \Delta x^2$$

If this equation is written for every interior node, the resulting simultaneous equations can be solved for

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.04436 & 0.06914 & 0.06914 & 0.04436 & 0 \\ 0 & 0.06914 & 0.10828 & 0.10828 & 0.06914 & 0 \\ 0 & 0.06914 & 0.10828 & 0.10828 & 0.06914 & 0 \\ 0 & 0.04436 & 0.06914 & 0.06914 & 0.04436 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

32.7 Grid scheme



All nodes in the above scheme can be modeled with the following general difference equation

$$\frac{h_{i+1,j} - 2h_{i,j} + h_{i-1,j}}{\Delta x^2} + \frac{h_{i,j+1} - 2h_{i,j} + h_{i,j-1}}{\Delta y^2} = 0$$

Node 0,0:

$$\frac{h_{1,0} - 2h_{0,0} + h_{-1,0}}{\Delta x^2} + \frac{h_{0,1} - 2h_{0,0} + h_{0,-1}}{\Delta y^2} = 0$$

The external nodes can be approximated with finite differences

$$\frac{dh}{dy} = \frac{h_{0,1} - h_{0,-1}}{2\Delta y}$$

$$h_{0,-1} = h_{0,1} - 2\Delta y \frac{dh}{dy} = h_{0,1}$$

$$\frac{dh}{dx} = \frac{h_{1,0} - h_{-1,0}}{2\Delta x}$$

$$h_{-1,0} = h_{1,0} - 2\Delta x \frac{dh}{dx} = h_{1,0} - 2(1)(1) = h_{1,0} - 2$$

which can be substituted into the difference equation to give

$$\frac{2h_{1,0} - 2h_{0,0} - 2}{\Delta x^2} + \frac{2h_{0,1} - 2h_{0,0}}{\Delta y^2} = 0$$

$$4h_{0,0} - 2h_{1,0} - 2h_{0,1} = -2$$

Node 1,0:

$$4h_{1,0} - 2h_{1,1} - h_{0,0} - h_{2,0} = 0$$

Node 2,0:

$$4h_{2,0} - 2h_{1,0} - 2h_{2,1} = 0$$

Node 0,1:

$$4h_{0,1} - 2h_{1,1} - h_{0,0} - h_{0,2} = -2$$

Node 1,1:

$$4h_{1,1} - h_{1,0} - h_{0,1} - h_{1,2} - h_{2,1} = 0$$

Node 2,1:

$$4h_{2,1} - h_{1,1} - h_{2,2} - h_{2,0} = 20$$

Node 0,2:

$$4h_{0,2} - 2h_{0,1} - 2h_{1,2} = -2$$

Node 1,2:

$$4h_{1,2} - h_{0,2} - h_{2,2} - 2h_{1,1} = 0$$

Node 2,2:

$$4h_{2,2} - h_{1,2} - 2h_{2,1} = 20$$

The equations can be solved simultaneously for

16.3372	17.37748	18.55022	20
16.29691	17.31126	18.4117	20
16.22792	17.15894	17.78532	

More refined results can be obtained by using finer grid spacing.

32.8 The fluxes can be determined using finite divided differences as

dh/dx			
1.04029	1.10651	1.31126	1.44978
1.01435	1.05740	1.34437	1.58830
0.93102	0.77870	0.62638	
dh/dy			
0.04029	0.06623	0.13852	0.00000
0.05464	0.10927	0.38245	0.00000
0.06898	0.15232	0.62638	
dh/dn			

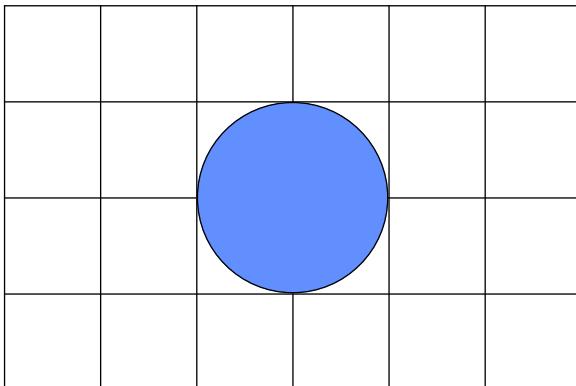
1.04107	1.10849	1.31855	1.44978
1.01582	1.06303	1.39771	1.58830
0.93357	0.79345	0.88583	
θ (radians)			
0.03871	0.05978	0.10525	0.00000
0.05381	0.10297	0.27716	0.00000
0.07396	0.19317	0.78540	
θ (degrees)			
2.21777	3.42509	6.03034	0.00000
3.08314	5.90002	15.88014	0.00000
4.23765	11.06765	45.00000	
Velocity			
-5.205E-04	-5.542E-04	-6.593E-04	-7.249E-04
-5.079E-04	-5.315E-04	-6.989E-04	-7.942E-04
-4.668E-04	-3.967E-04	-4.429E-04	

32.9 Because of the equi-spaced grid, the domain can be modeled with simple Laplacians.

The resulting solution is

25	40	40	30		
10	21.87149	24.04033	20	15	
10	13.44564	14.28983	12.63401	10	7.5
10	7.62124	7.039322	6.246222	5.311556	5
5	0	0	0	0	2.5

32.10 A convenient segmentation scheme can be developed as



Simple Laplacians reflecting the boundary conditions can be developed and solved for

99.56378	99.49429	99.13184	97.64677	91.45524	81.4052	70
99.63328	99.64076	99.69315	100	93.3845	82.08277	70
99.6878	99.74233	100	100	100	83.54139	70
99.63328	99.64076	99.69315	100	93.3845	82.08277	70
99.56378	99.49429	99.13184	97.64677	91.45524	81.4052	70

32.11 The system to be solved is

$$\begin{bmatrix} 2.25 & -1.5 & & \\ -1.5 & 2 & -0.5 & \\ & -0.5 & 2.5 & -2 \\ & & -2 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.5 \end{Bmatrix}$$

which can be solved for $x_1 = 2$, $x_2 = 3$, $x_3 = 6$, and $x_4 = 6.75$.

32.12 The system to be solved is

$$\begin{bmatrix} 0.75 & -0.5 & & & \\ -0.5 & 2 & -1.5 & & \\ & -1.5 & 2.25 & -0.75 & \\ & & -0.75 & 1.75 & -1 \\ & & & -1 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{Bmatrix}$$

which can be solved for $x_1 = 8$, $x_2 = 12$, $x_3 = 13.33333$, $x_4 = 16$, and $x_5 = 18$.

32.13 Substituting the Crank-Nicolson finite difference analogues to the derivatives

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{2} \left[\frac{u_{i+1,n+1} - u_{i,n+1} + u_{i-1,n+1}}{\Delta x^2} + \frac{u_{i+1,n} - u_{i,n} + u_{i-1,n}}{\Delta x^2} \right]$$

$$\frac{\partial u}{\partial t} = \frac{u_{i,n+1} - u_{i,n}}{\Delta t}$$

into the governing equations gives the following finite difference equations:

$$[1] u_{i-1,n+1} + \left[-2 - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n+1} + [1] u_{i+1,n+1} = -u_{i-1,n} + \left[2 - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n} - u_{i+1,n} \quad 0 \leq x \leq \frac{1}{2}$$

$$[r] u_{i-1,n+1} + \left[-2r - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n+1} + [r] u_{i+1,n+1} = -ru_{i-1,n} + \left[2r - 2 \frac{\Delta x^2}{\Delta t} \right] u_{i,n} - ru_{i+1,n} \quad \frac{1}{2} \leq x \leq 1$$

Substitute for the end point boundary conditions to get the end point finite difference equations. Substitute the first order Crank Nicolson analogues to the derivatives

$$\frac{\partial u}{\partial r} = \frac{1}{2} \left[\frac{u_{i+1,n+1} - u_{i-1,n+1}}{2\Delta r} + \frac{u_{i+1,n} - u_{i-1,n}}{2\Delta r} \right]$$

into the midpoint boundary condition and get

$$u^a_{L+1,n+1} + u^a_{L+1,n} + r(u^b_{L+1,n+1} + u^b_{L+1,n}) = u_{L-1,n+1} + u_{L=1,n} + r(u_{L+1,n+1} + u_{L+1,n})$$

where u^a and u^b are fictitious points located in the opposite side of the midpoint from their half. Write out the two finite difference equations from above for the point $i = L$ (the midpoint) and then combine these two equations with the midpoint boundary condition to obtain the midpoint finite difference equation:

$$[2]u_{L-1,n+1} + \left[-2(1+r) - 4\frac{\Delta x^2}{\Delta t} \right]u_{L,n+1} + [2]u_{L+1,n+1} = -2u_{L-1,n} + \left[2(1+r) - 4\frac{\Delta x^2}{\Delta t} \right]u_{L,n} - (1+r)u_{L+1,n}$$

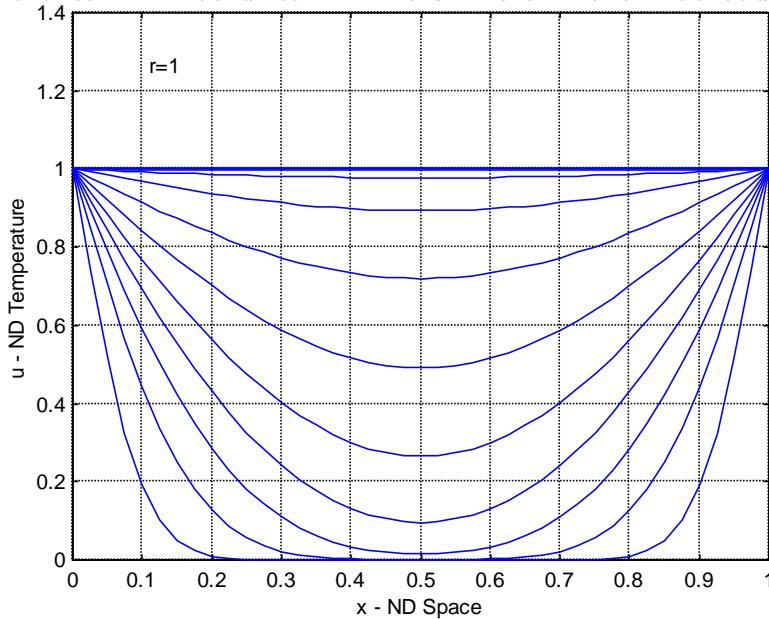
```
%PDE Parabolic Problem - Transient Heat conduction in a composite rod
% u[xx]=u[t]          0<xx<0.5
% r(u[xx])=u[t]        0.5<xx<1
% BC u(0,t)=1  u(1,t)=1
% u[x]=r(u[x])        x=0.5
% IC   u(x,0)=0        0<x<1
% i=spatial index, from 1 to imax
% R = no. of x points (R=21 for 20 dx spaces)
% n=time index from 1 to N
% N = no. of time steps,
% Crank-Nicolson Formulation
R=41;      %(imax must be odd for point L to be correct)
N=69;       % last time step = nmax+1
L=(R-1)/2+1; % L = midpoint of point no. (for R=41, L=21)
% Constants
r=0.01;
dx=1/(R-1);
dx2=dx*dx;
dt=dx2;           % Setting dt to dx2 for good stability and results
% Independent space variable
x=0:dx:1;
% Sizing matrices
u=zeros(R,N+1); t=zeros(1,N+1);
a=zeros(1,R); b=zeros(1,R);
c=zeros(1,R); d=zeros(1,R);
ba=zeros(1,R); ga=zeros(1,R);
up=zeros(1,R);
% Boundary Conditions at t=0
u(1,1)=1;
u(R,1)=1;
% Time step loop
% n=1 represents 0 time, next time = n+1
t(1)=0;
for n=1:N
    t(n+1)=t(n)+dt;
    % Boundary conditions & Constants
    u(1,n+1)=1;
    u(R,n+1)=1;
    dx2dt=dx2/dt;
    % coefficients
    b(2)=-2-2*dx2dt;
    c(2)=1;
    d(2)=(2-2*dx2dt)*u(2,n)-u(3,n)-2;
    for i=3:L-1
        a(i)=1;
        b(i)=-2-2*dx2dt;
        c(i)=1;
        d(i)=-u(i-1,n)+(2-2*dx2dt)*u(i,n)-u(i+1,n);
    end
    a(L)=2;
```

```

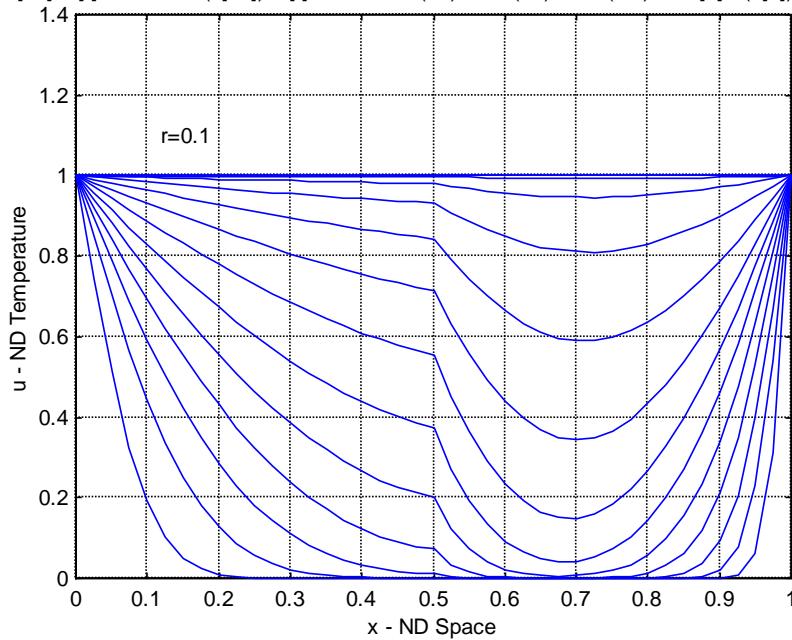
b(L)=-2*(1+r)-4*dx2dt;
c(L)=2*r;
d(L)=-2*u(L-1,n)+(2*(1+r)-4*dx2dt)*u(L,n)-2*r*u(L+1,n);
for i=L+1:R-2
    a(i)=r;
    b(i)=-2*r-2*dx2dt;
    c(i)=r;
    d(i)=-r*u(i-1,n)+(2*r-2*dx2dt)*u(i,n)-r*u(i+1,n);
end
a(R-1)=r;
b(R-1)=-2*r-2*dx2dt;
d(R-1)=-r*u(R-2,n)+(2*r-2*dx2dt)*u(R-1,n)-2*r;
% Solution by Thomas Algorithm
ba(2)=b(2);
ga(2)=d(2)/b(2);
for i=3:R-1
    ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
    ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
end
% Back substitution step
u(R-1,n+1)=ga(R-1);
for i=R-2:-1:2
    u(i,n+1)=ga(i)-c(i)*u(i+1,n+1)/ba(i);
end
dt=1.1*dt;
end
% end of time step loop
% Plot
% Storing plot value of u as up, at every 5 time steps
% j=time index
% i=space index
for j=5:5:N+1
    for i=1:R
        up(i)=u(i,j);
    end
    plot(x,up)
    hold on
end
grid
title('u[xx]=u[t] 0<x<0.5; r(u[xx])=u[t] 0.5<x<1; u(0,t)=1, u(1,t)=1,
u(x,0)=0; u[x]=r(u[x]) x=0.5')
xlabel('x - ND Space')
ylabel('u - ND Temperature')
hold off
gtext('r=0.01')

```

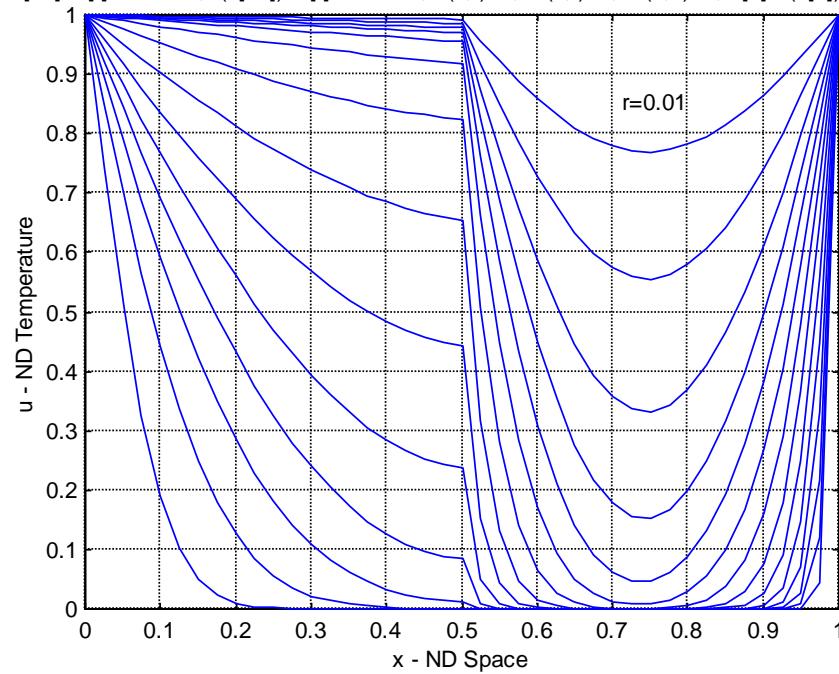
$$u[xx]=u[t] \quad 0 < x < 0.5; \quad r(u[xx])=u[t] \quad 0.5 < x < 1; \quad u(0,t)=1, \quad u(1,t)=1, \quad u(x,0)=0; \quad u[x]=r(u[x]) \quad x=0.5$$



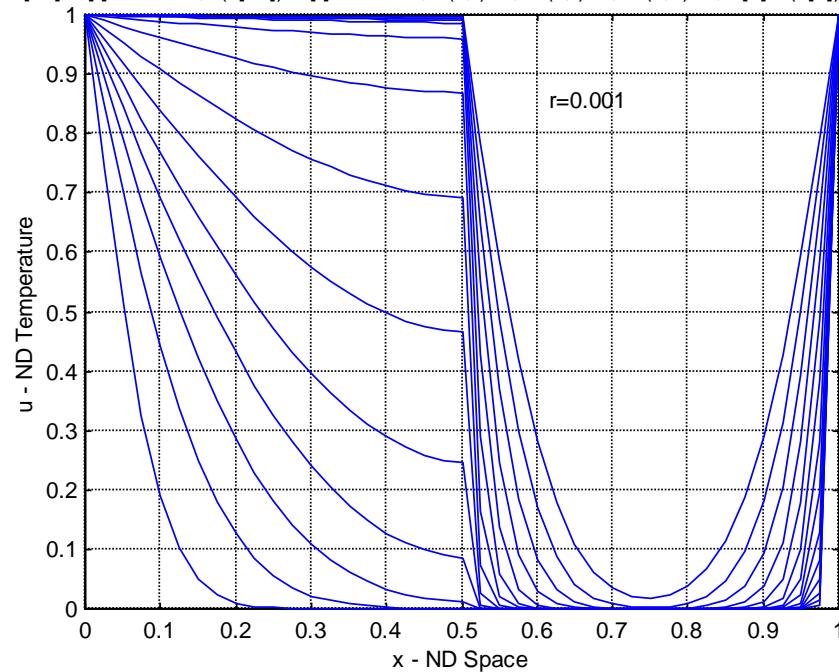
$$u[xx]=u[t] \quad 0 < x < 0.5; \quad r(u[xx])=u[t] \quad 0.5 < x < 1; \quad u(0,t)=1, \quad u(1,t)=1, \quad u(x,0)=0; \quad u[x]=r(u[x]) \quad x=0.5$$

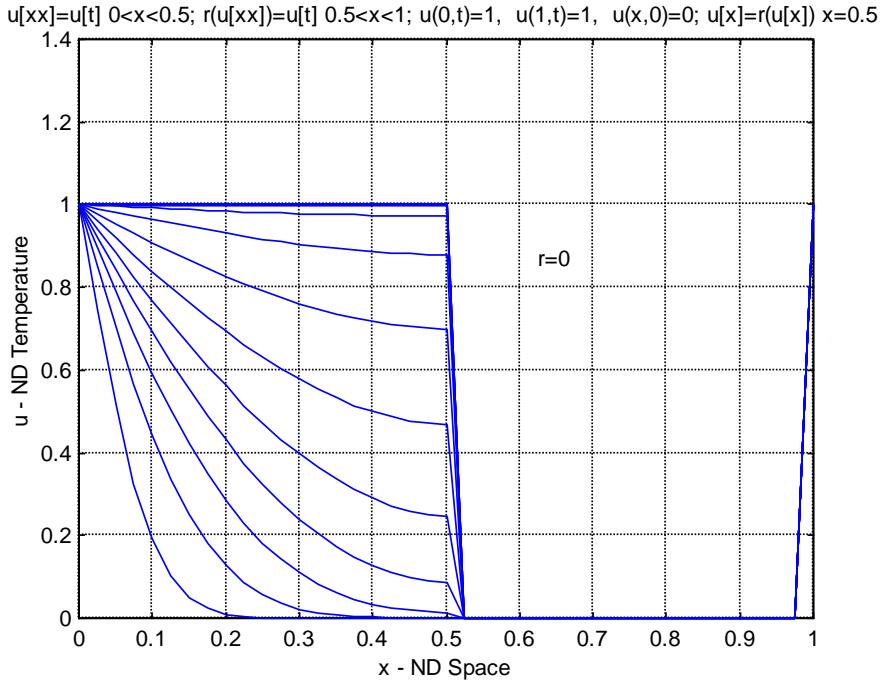


$$u[xx]=u[t] \quad 0 < x < 0.5; \quad r(u[xx])=u[t] \quad 0.5 < x < 1; \quad u(0,t)=1, \quad u(1,t)=1, \quad u(x,0)=0; \quad u[x]=r(u[x]) \quad x=0.5$$



$$u[xx]=u[t] \quad 0 < x < 0.5; \quad r(u[xx])=u[t] \quad 0.5 < x < 1; \quad u(0,t)=1, \quad u(1,t)=1, \quad u(x,0)=0; \quad u[x]=r(u[x]) \quad x=0.5$$





32.14

```
% PDE Parabolic Problem - Heat conduction in a rod
%  $u_{xx} + u_{yy} = u_t$ 
% BC:  $u(0,y,t) = 0$   $u(l,y,t) = 1$ 
% BC:  $u(x,0,t) = 0$   $u(x,l,t) = 1$ 
% IC:  $u(x,y,0) = 0$   $0 \leq x \leq l$   $0 \leq y \leq l$ 
% Crank-Nicolson Formulation
% ADI Solution Method
% Intermediate value of  $u$  stored as  $u_l$ 
% MATLAB version with multi-dimensional arrays
% i=spatial index in x-direction, from 1 to R
% j=spatial index in y-direction, from 1 to S
% n=time index from 1 to N
R=21; % last x-point
S=21; % last y-point
N=20; % last time step = N+1
% Constants
dx=1/(R-1);
dx2=dx*dx;
dy=1/(S-1);
dy2=dy*dy;
dxdy=dx2/dy2;
dydx=dy2/dx2;
dt=dx2; %setting dt to dx2 vfor good stability and results
% Independent space variables
x=0:dx:1;
y=0:dy:1;
% Sizing matrices
u=zeros(R,S,N); u1=zeros(R,S);
%  $u(l,j,n)$ =present time,  $u1$ =first pass intermediate results
%  $u(i,j,n+1)$ =next time step
t=zeros(1,N+1);
a=zeros(1,R); b=zeros(1,R); c=zeros(1,R); d=zeros(1,R);
ba=zeros(1,R); ga=zeros(1,R);
```

```

% Boundary conditions
for n=1:N
    for i=1:R
        u(i,S,n)=1;
    end
    for j=1:S
        u(R,j,n)=1;
    end
end
% Intermediate values
for i=1:R
    u1(i,S)=1;
end
for j=1:S
    u1(R,j)=1;
end
% Plot initial conditions
mesh(x,y,u(:,:,1))
title('u[xx]+u[yy]=u[t];u(0,y,t)=0,u(1,y,t)=1,u(x,0,t)=0,u(x,1,t)=1,u(x,y,0)=0')
xlabel('x-coordinate');ylabel('y-coordinate');zlabel('u-Temperature');
pause
% ****
% Time step loop
%: n=1 represents 0 time, n+1 = next time step
t(1)=0;
for n=1:N
    t(n+1)=t(n)+2*dt;
    % First pass in x-direction ****
    % first time step - intermediate value at u1(i,j) are calculated
    % Constants
    dx2dt=dx2/dt;
    % Coefficients
    for j=2:S-1
        b(2)=-2-dx2dt;
        c(2)=1;
        d(2)=-dxdy*u(2,j-1,n)+(2*dxdy-dx2dt)*u(2,j,n)-dxdy*u(2,j+1,n);
        for i=3:R-2
            a(i)=1;
            b(i)=-2-dx2dt;
            c(i)=1;
            d(i)=-dxdy*u(i,j-1,n)+(2*dxdy-dx2dt)*u(i,j,n)-dxdy*u(i,j+1,n);
        end
        a(R-1)=1;
        b(R-1)=-2-dx2dt;
        d(R-1)=-1-dxdy*u(i,j-1,n)+(2*dxdy-dx2dt)*u(i,j,n)-dxdy*u(i,j+1,n);
        % Solution by Thomas algorithm
        ba(2)=b(2);
        ga(2)=d(2)/ba(2);
        for i=3:R-1
            ba(i)=b(i)-a(i)*c(i-1)/ba(i-1);
            ga(i)=(d(i)-a(i)*ga(i-1))/ba(i);
        end
        % Back substitution step
        u1(R-1,j)=ga(R-1);
        for i=R-2:-1:2
            u1(i,j)=ga(i)-c(i)*u1(i+1,j)/ba(i);
        end
    end
    % second pass in y-direction ****
    % second time step - final values at u(i,j,n+1) are calculated
    dy2dt=dy2/dt;
    % coefficients

```

```

for i=2:R-1
    b(2)=-2-dy2dt;
    c(2)=1;
    d(2)=-dydx*u1(i-1,2)+(2*dydx-dy2dt)*u1(i,2)-dydx*u1(i+1,2);
    for j=3:S-2
        a(j)=1;
        b(j)=-2-dy2dt;
        c(j)=1;
        d(j)=-dydx*u1(i-1,j)+(2*dydx-dy2dt)*u1(i,j)-dydx*u1(i+1,j);
    end
    a(S-1)=1;
    b(S-1)=-2-dy2dt;
    d(S-1)=-1-dydx*u1(i-1,S-1)+(2*dydx-dy2dt)*u1(i,S-1)-dydx*u1(i+1,S-1);
    % Solution by Thomas algorithm
    ba(2)=b(2);
    ga(2)=d(2)/ba(2);
    for j=3:S-1
        ba(j)=b(j)-a(j)*c(j-1)/ba(j-1);
        ga(j)=(d(j)-a(j)*ga(j-1))/ba(j);
    end
    % Back substitution step
    u(i,S-1,n+1)=ga(S-1);
    for j=S-2:-1:2
        u(i,j,n+1)=ga(j)-c(j)*u(i,j+1,n+1)/ba(j);
    end
end
% dt can be incremented at this point if desired as dt=1.1*dt
end      % end time step loop
%*****
% Plot results
mesh(x,y,u(:,:,10))
title('u[xx]+u[yy]=u[t];u(0,y,t)=0,u(1,y,t)=1,u(x,0,t)=0,u(x,1,t)=1,u(x,y,0)=0')
xlabel('x-coordinate');ylabel('y-coordinate');zlabel('u-Temperature');
pause
t(10)
mesh(x,y,u(:,:,20))
title('u[xx]+u[yy]=u[t];u(0,y,t)=0,u(1,y,t)=1,u(x,0,t)=0,u(x,1,t)=1,u(x,y,0)=0')
xlabel('x-coordinate');ylabel('y-coordinate');zlabel('u-Temperature');
pause
t(20)

```

