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

$$\lambda^2 - 0.15 = 0$$
$$\lambda = \pm\sqrt{0.15}$$

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,

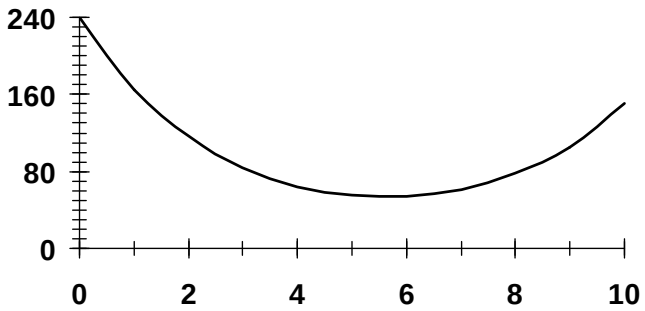
$$240 = A + B$$
$$150 = 48.08563A + 0.020796B$$

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:

$$\frac{dT}{dx} = z$$

$$\frac{dz}{dx} = 0.15T$$

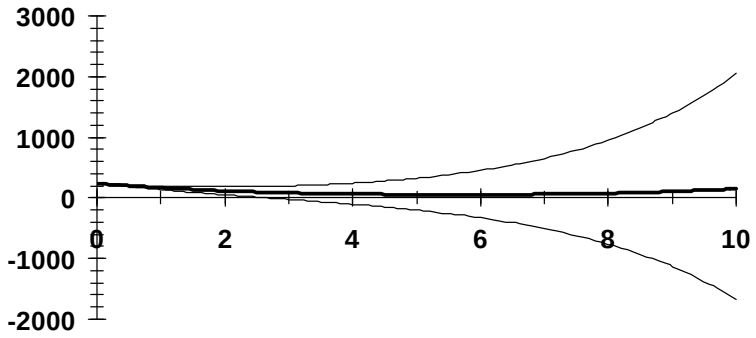
The solution was then generated on the Excel spreadsheet using the Heun method (without iteration) with a step-size of 0.01. An initial condition of $z = -120$ was chosen for the first shot. The first few calculation results are shown below.

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

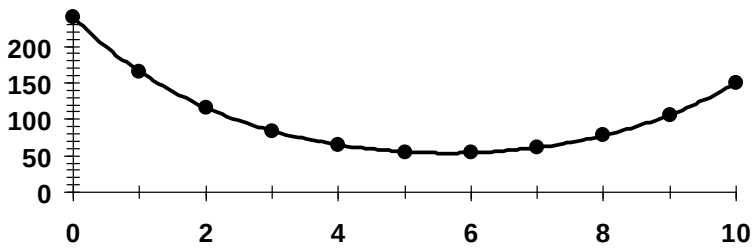
The resulting value at $x = 10$ was $T(10) = -1671.817$. A second shot using an initial condition of $z(0) = -60$ was attempted with the result at $x = 10$ of $T(10) = 2047.766$. These values can then be used to derive the correct initial condition,

$$z(0) = -120 + \frac{-60 + 120}{2047.766 - (-1671.817)}(150 - (-1671.817)) = -90.6126$$

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

$$2.15T_1 - T_2 = 240$$

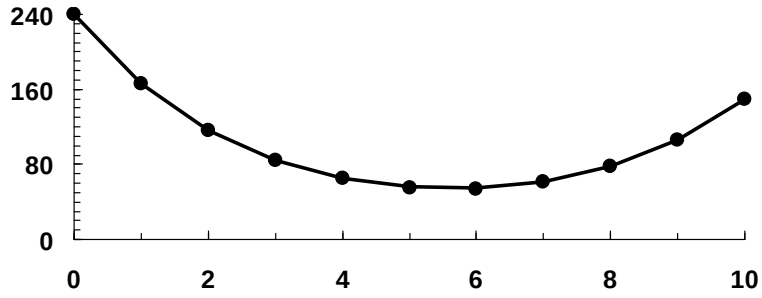
and the last node would be

$$-T_9 + 2.15T_{10} = 150$$

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,

$$\frac{dy}{dx} = z$$
$$\frac{dz}{dx} = \frac{2z + y - x}{7}$$

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

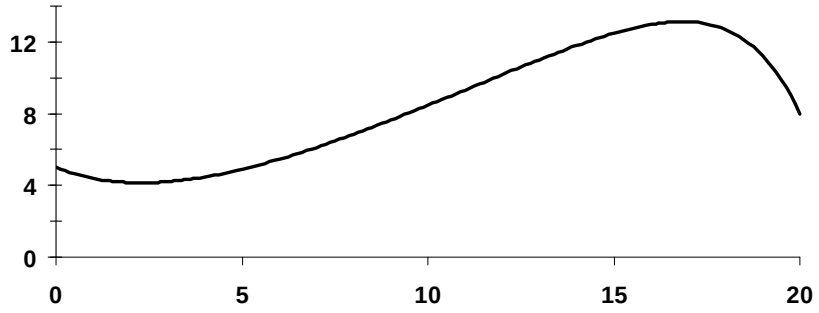
$z(0)$	–1	–0.5
$y(20)$	–11,837.64486	22,712.34615

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,

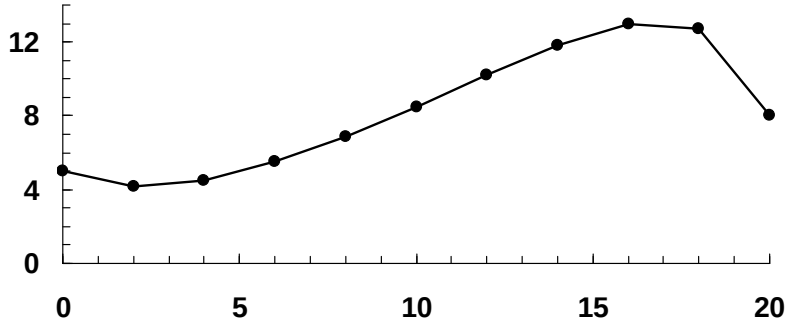
$$7\frac{y_{i+1}-2y_i+y_{i-1}}{\Delta x^2}-2\frac{y_{i+1}-y_{i-1}}{2\Delta x}-y_i+x_i=0$$

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



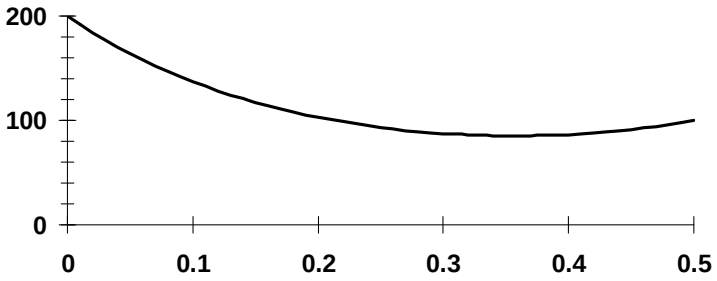
27.6 The second-order ODE can be expressed as the following pair of first-order ODEs,

$$\frac{dT}{dx} = z$$

$$\frac{dz}{dx} = 1\times10^{7}(T+273)^{4} - 4(150-T)$$

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

$$\frac{d^2T}{dx^2} - 1\times10^7 (T_b + 273)^4 - 4\times10^7 (T_b + 273)^3 (T - T_b) + 5(150 - T) = 0$$

Substituting T_b = 150 and collecting terms gives

$$\frac{d^2T}{dx^2} - 34.27479T + 1939.659 = 0$$

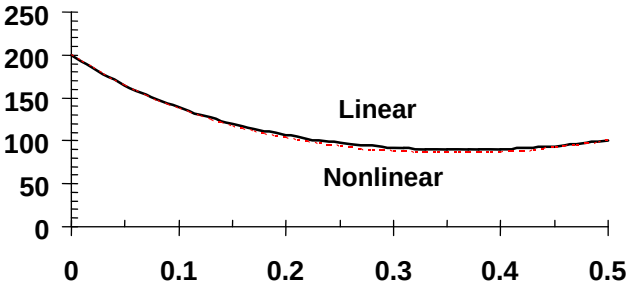
Substituting a centered-difference approximation of the second derivative gives

$$-T_{i-1} + (2 + 34.27479\Delta x^2)T_i - T_{i+1} = 1939.659\Delta x^2$$

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

$$\begin{aligned}
\left(\frac{2k_1}{m_1} - \omega^2\right)A_1 \quad & -\frac{k_1}{m_1}A_2 & & = 0 \\
-\frac{k_1}{m_1}A_1 \quad & +\left(\frac{2k_1}{m_1} - \omega^2\right)A_2 \quad & -\frac{k_1}{m_1}A_3 \quad & = 0 \\
& -\frac{k_1}{m_1}A_2 \quad & +\left(\frac{2k_1}{m_1} - \omega^2\right)A_3 \quad & = 0
\end{aligned}$$

Substituting m = 40 kg and k = 240 gives

$$\begin{aligned}
\left(12 - \omega^2\right)A_1 \quad & -6A_2 & & = 0 \\
-6A_1 \quad & +\left(12 - \omega^2\right)A_2 \quad & -6A_3 \quad & = 0 \\
& -6A_2 \quad & +\left(12 - \omega^2\right)A_3 \quad & = 0
\end{aligned}$$

The determinant is

$$-\omega^6 + 36\omega^4 - 360\omega^2 + 864 = 0$$

which can be solved for ω^2 = 20.4853, 12, and 3.5147 s^{-2}. Therefore the frequencies are ω = 4.526, 3.464, and 1.875 s^{-1}. Substituting these values into the original equations yields for ω^2 = 20.4853,

$$A_1 = -0.707A_2 = A_3$$

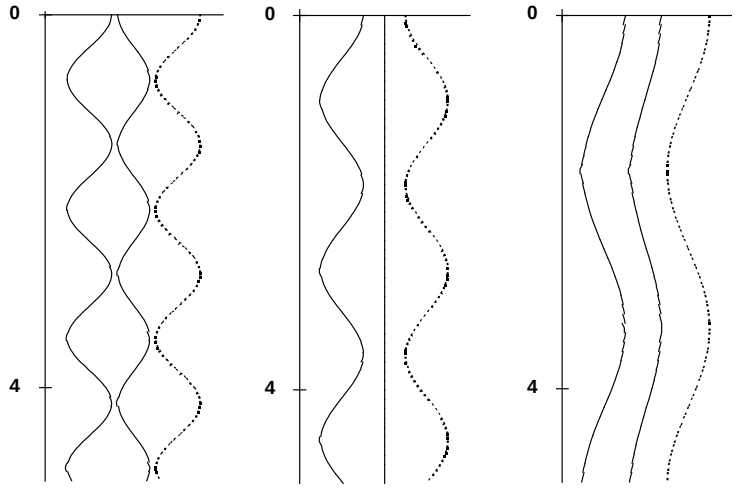
for ω^2 = 12

$$A_1 = -A_3, \text{ and } = A_2 = 0$$

for ω^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

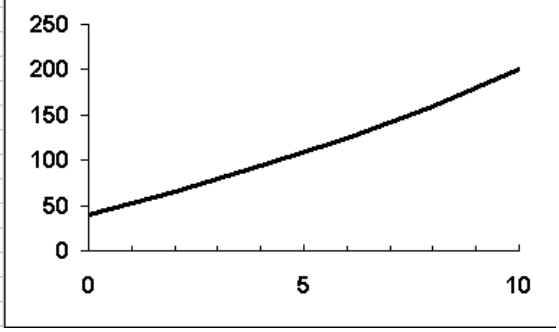
Thus, after four iterations, the estimate of the lowest eigenvalue is 1/(3.398537) = 0.294244 with an eigenvector of [0.0411696 1 –0.8070225].

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
  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
```

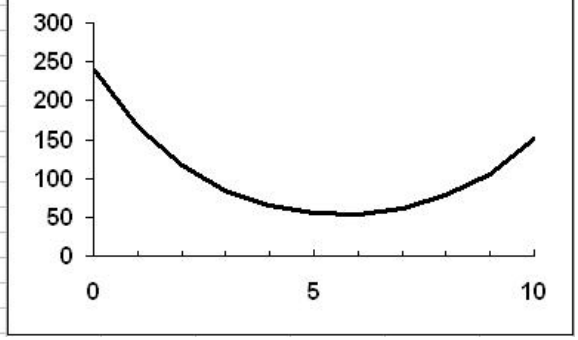
```
  yp(i, m) = y(i)
Next i
Do
  xend = x + xout
  If xend > xf Then xend = xf
  h = dx
  Do
    If xend - x < h Then h = xend - x
    Call RK4(x, y, n, h)
    If x >= xend Then Exit Do
  Loop
  m = m + 1
  xp(m) = x
  For i = 1 To n
    yp(i, m) = y(i)
  Next i
  If x >= xf Then Exit Do
Loop
End Sub

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
```

	A	B	C	D	E	F	G	H	I	J	K	L
1	Prob. 27.13											
2												
3	t	T	z									
4	0	40	12.69067		RUN							
5	2	65.95189	13.348									
6	4	93.74796	14.54101									
7	6	124.5037	16.3176									
8	8	159.4536	18.74904									
9	10	200	21.93294									
10												
11												
12												
13												
14												

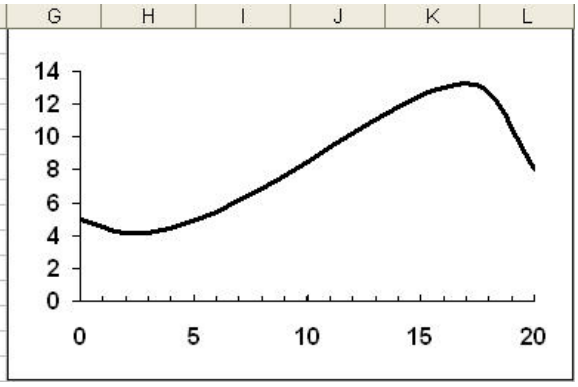


27.14

	A	B	C	D	E	F	G	H	I	J	K	L
1	Prob. 27.2											
2												
3	t	T	z									
4	0	240	-90.6146		RUN							
5	1	165.33	-60.5896									
6	2	115.7703	-39.767									
7	3	83.79376	-24.9843									
8	4	64.5439	-13.9961									
9	5	55.09699	-5.13363									
10	6	54.01824	2.949131									
11	7	61.14383	11.47982									
12	8	77.556	21.75409									
13	9	105.7475	35.33241									
14	10	150	54.27706									



	A	B	C	D	E	F	G	H	I	J	K	L
1	Prob. 27.4											
2												
3	t	T	z									
4	0	5	-0.82857		RUN							
5	2	4.151209	-0.08474									
6	4	4.460771	0.355953									
7	6	5.455657	0.615941									
8	8	6.851982	0.766008									
9	10	8.4714	0.84268									
10	12	10.17836	0.851334									
11	14	11.80357	0.746539									
12	16	12.98116	0.356171									
13	18	12.70164	-0.8553									
14	20	8	-4.49965									



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

$$-\left(a-0.5b\Delta x\right)y_{i-1}+\left(2a-c\Delta x^2\right)y_i-\left(a+0.5b\Delta x\right)y_{i+1}=f(x_i)\Delta x^2$$

Dividing by Δx^2,

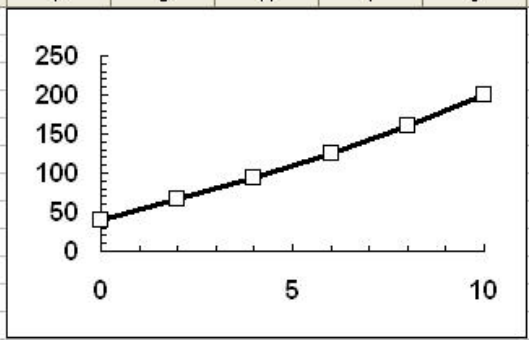
$$-\left(a/\Delta x^2-0.5b/\Delta x\right)y_{i-1}+\left(2a/\Delta x^2-c\right)y_i-\left(a/\Delta x^2+0.5b/\Delta x\right)y_{i+1}=f(x_i)$$

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
  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)
```

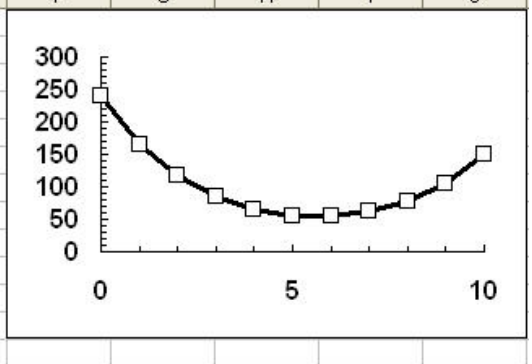
```
Sheets("Sheet1").Select
Range("a5:b105").ClearContents
Range("a5").Select
For i = 0 To ns
  ActiveCell.Value = x(i)
  ActiveCell.Offset(0, 1).Select
  ActiveCell.Value = y(i)
  ActiveCell.Offset(1, -1).Select
Next i
Range("a5").Select
End Sub
Sub Tridiag(e, f, g, r, n, x)
Dim k As Integer
For k = 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
```

	A	B	C	D	E	F	G	H	I	J
1	Example 27.3									
2										
3										
4	x	y								
5	0	40								
6	2	65.96983		RUN						
7	4	93.77846								
8	6	124.5382								
9	8	159.4795								
10	10	200								
11										
12										

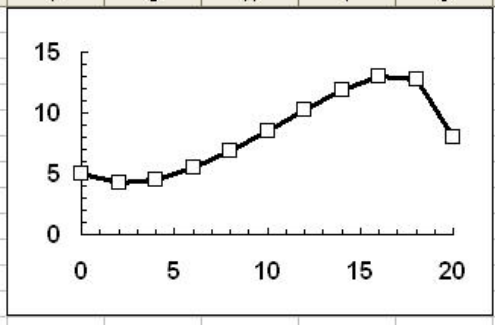


27.16

	A	B	C	D	E	F	G	H	I	J
1	Problem 27.3									
2										
3										
4	x	y								
5	0	240								
6	1	165.7573		RUN						
7	2	116.3782								
8	3	84.45583								
9	4	65.20183								
10	5	55.72811								
11	6	54.61361								
12	7	61.69114								
13	8	78.02235								
14	9	106.0569								
15	10	150								



	A	B	C	D	E	F	G	H	I	J
1	Problem 27.5									
2										
3										
4	x	y								
5	0	5								
6	2	4.199592		RUN						
7	4	4.518531								
8	6	5.507445								
9	8	6.893447								
10	10	8.503007								
11	12	10.20262								
12	14	11.82402								
13	16	13.00176								
14	18	12.7231								
15	20	8								



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**		RUN	
4	**eigenvalue:**	**1.777778**			
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, 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 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):
%   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

$$R = \frac{T_{i-1} - 2T_i + T_{i+1}}{(\Delta x)^2} - 1\times10^{-7}(T_i + 273)^4 + 4(150 - T_i)$$

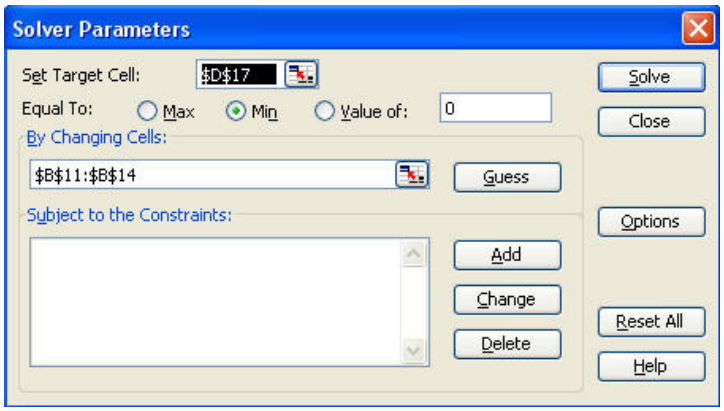
where R = the residual, which is equal to zero when the equation is satisfied. Next, a spreadsheet can be set up as below. Guesses for T can be entered in cells B11:B14. Then, the residual equation can be written in cells C11:C14 and referenced to the temperatures in column B. The square of the R's can 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				
17			SSR	502680505

= sum(D11:D14)

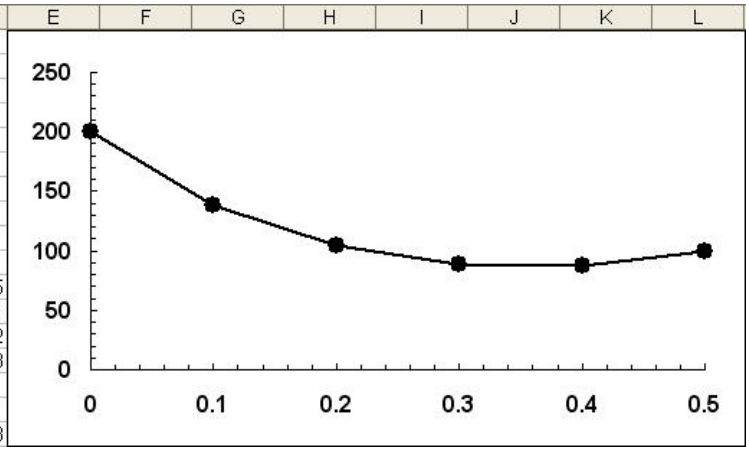
=(B10-2*B11+B12)/$B$7^2-$B$2*(B11+273)^4+$B$3*($B$4-B11)

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.

	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	137.9163	0.022505	0.0005065
12	0.2	103.8606	0.044834	0.0020101
13	0.3	88.13052	0.056053	0.003142
14	0.4	86.93437	0.049626	0.0024628
15	0.5	100		
16				
17			SSR	0.0081213



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)];
```

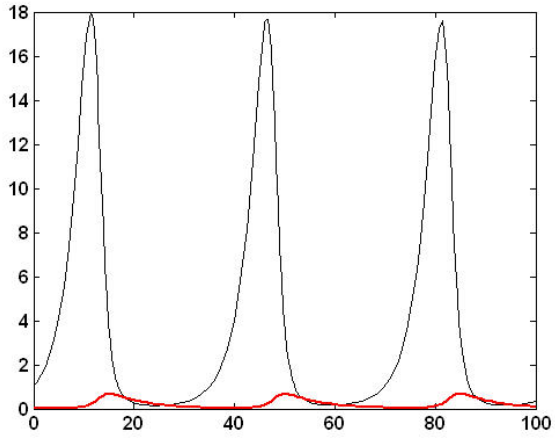
Then, the following MATLAB session is used to generate the solution:

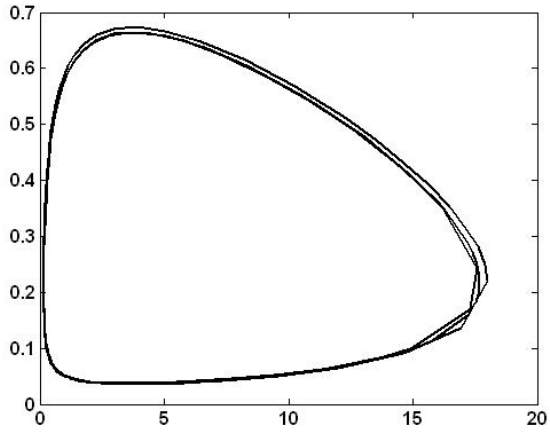
```
>> [t,y]=ode45(@predprey,[0 100],[1;.05]);
```

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

$$\frac{dx}{dt} = z$$

$$\frac{dz}{dt} = -8z - 1200x$$

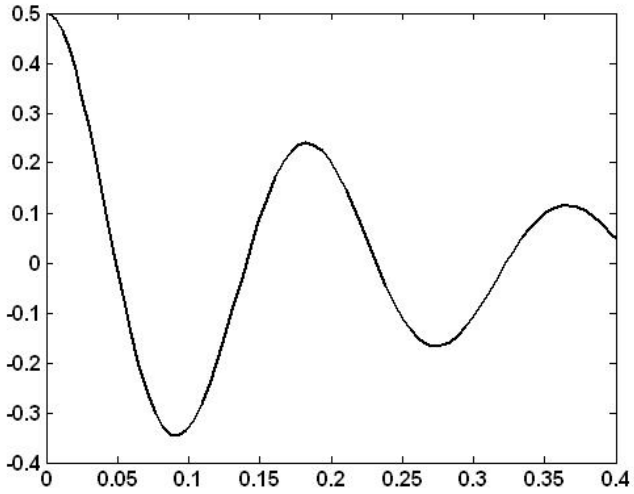
Next, we create an M-file to hold the ODEs:

```
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
```

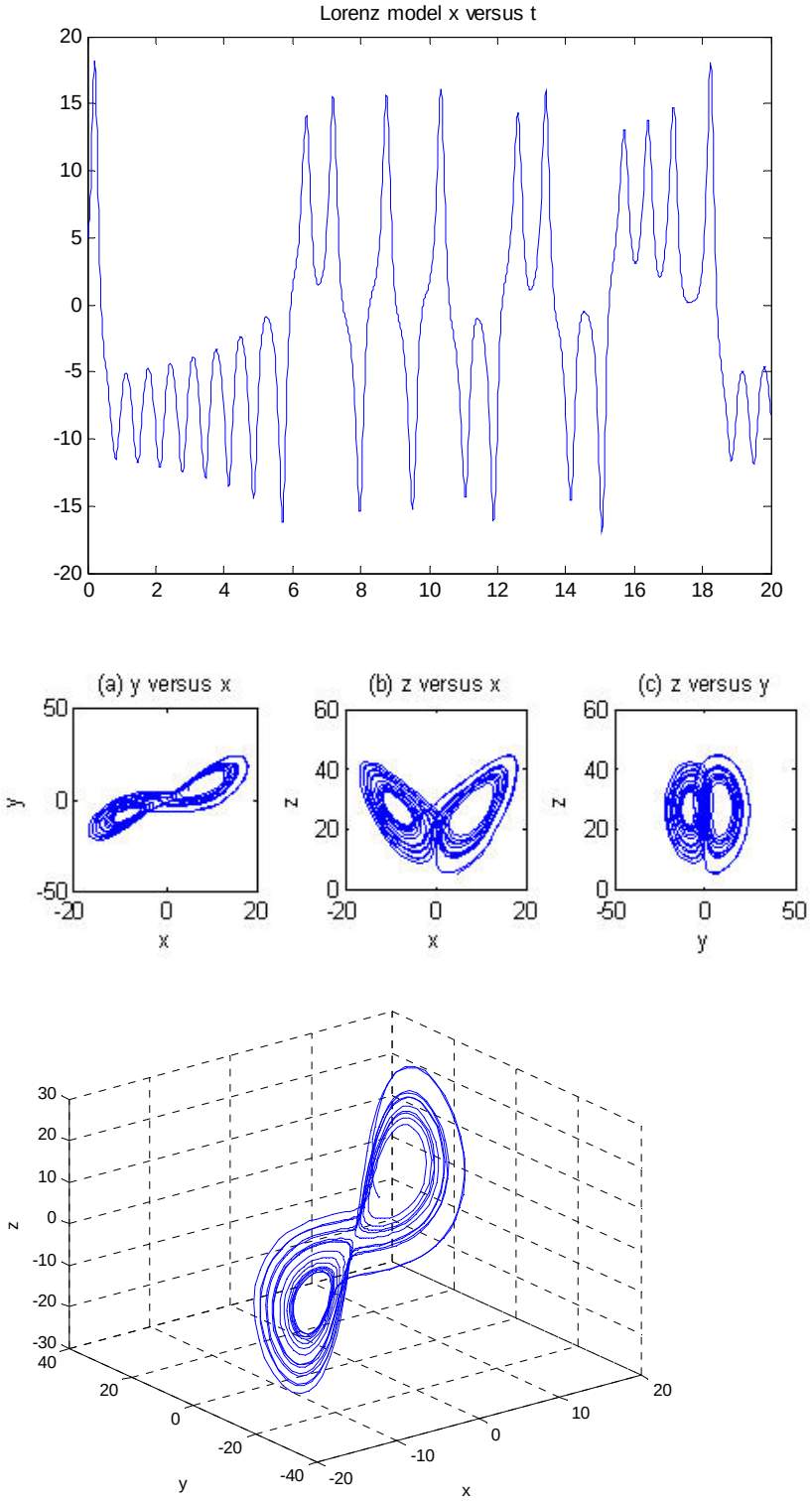
27.22 Here is a MATLAB function to hold the differential equations:

```
function yp=lorenz(t,y,sigma,b,r)
yp=[-sigma*y(1)+sigma*y(2);r*y(1)-y(2)-y(1)*y(3);-b*y(3)+y(1)*y(2)];
```

Here is a script to solve the equations and develop both time series and two and three dimensional phase plane plots of the solutions.

```
tspan=[0 20];y0=[5 5 5];
sigma=10;b=8/3;r=28;
[t y] = ode45(@lorenz,tspan,y0,[],sigma,b,r);
plot(t,y(:,1))
title('Lorenz model x versus t');
pause
subplot(1,3,1);plot(y(:,1),y(:,2))
xlabel('x');ylabel('y')
axis square;title('(a) y versus x')
subplot(1,3,2);plot(y(:,1),y(:,3))
xlabel('x');ylabel('z')
axis square;title('(b) z versus x')
subplot(1,3,3);plot(y(:,2),y(:,3))
xlabel('y');ylabel('z')
axis square;title('(c) z versus y')
pause
subplot(1,1,1)
plot3(y(:,1),y(:,2),y(:,2))
xlabel('x');ylabel('y');zlabel('z');grid
```

Here are the resulting plots:



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 = l; and at $x = 2$, $i = n$

$$\Delta x = \frac{2-0}{n-1}$$

2. Finite Difference Equation

$$\frac{d^2u}{dx^2} + 6\frac{du}{dx} - u = 2$$

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:
$a_i = 1 - 3\Delta x$ $b_i = -2 - \Delta x^2$ $c_i = 1 + 3\Delta x$ $d_i = 2\Delta x^2$

3. End point equations

<u>$i = 2$:</u>

$$[1-3(\Delta x)]10 + [-2-\Delta x^2]u_2 + [1+3(\Delta x)]u_3 = 2\Delta x^2$$

Coefficients:
$a_2 = 0$ $b_2 = -2 - \Delta x^2$ $c_2 = 1 + 3\Delta x$ $d_2 = 2\Delta x^2 - 10(1 - 3(\Delta x))$

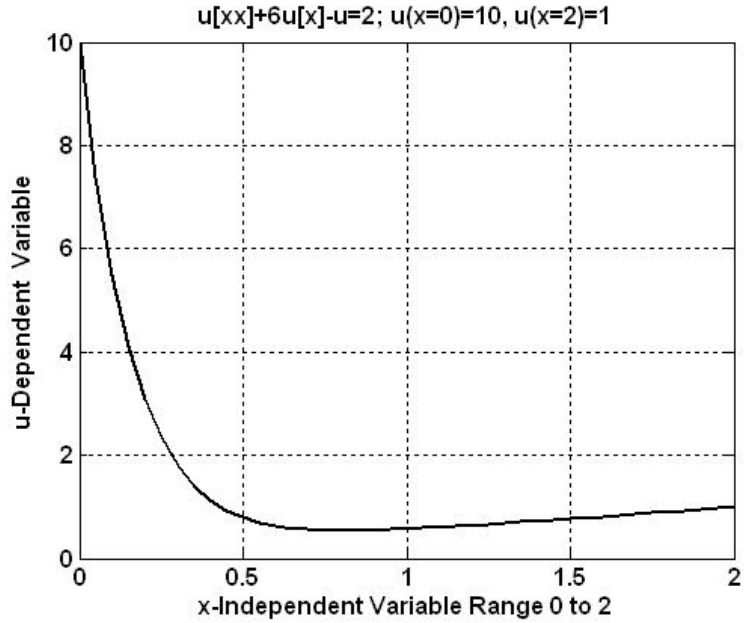
<u>$i = n - 1$:</u>

$$[1-3(\Delta x)]u_{n-2} + [-2-\Delta x^2]u_{n-1} + [1+3(\Delta x)]1 = 2\Delta x^2$$

Coefficients:
$a_2 = 1 - 3\Delta x$ $b_2 = -2 - \Delta x^2$ $c_2 = 0$ $d_2 = 2\Delta x^2 - (1 - 3(\Delta x))$

```
% Boundary Value Problem
% u[xx]+6u[x]-u=2
% BC: u(x=0)=10  u(x=2)=1
% i=spatial index from 1 to n
% numbering for points is i=l 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

1. Divide the radial coordinate into *n* finite points.

$$\Delta r = \frac{1}{n-1}$$

2. 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:

$$-\left[1 - \frac{1}{2(i-1)}\right]T_2 + 2T_1 - \left[1 + \frac{1}{2(i-1)}\right]T_2 = \Delta r^2 S$$

Collecting terms gives

$$2T_1 - 2T_2 = \Delta r^2 S$$

6. End point equation at $i = n - 1$

$$T(r = 1) = 1$$

$$-\left[1 - \frac{1}{2(i-1)}\right] T_{n-2} + 2T_{n-1} = \Delta r^2 S + \left[1 + \frac{1}{2(i-1)}\right]$$

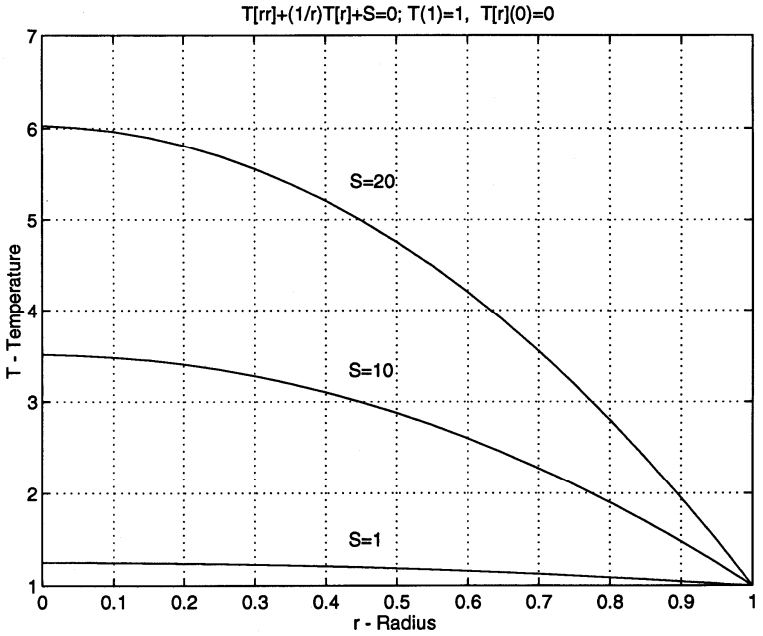
7. Solve the resulting tridiagonal system of algebraic equations using the Thomas Algorithm.

8. Following program in MATLAB.

```
% Solution of the ODE Boundary Value Problem
% T[rr]+(l/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 resu1ting 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: $k_1 = k_4 = 15$ N/m, $k_2 = k_3 = 35$ N/m, and $m_1 = m_2 = m_3 = 1.5$ kg

$$\left[\frac{k}{m}\right]=\begin{bmatrix} 33.33333 & -23.33333 & 0 \\ -23.33333 & 46.66667 & -23.33333 \\ 0 & -23.33333 & 33.33333 \end{bmatrix}$$

Solve for the eigenva1ues/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.

$m\omega_1^2 = 0 \Rightarrow \omega_1 = 0$ (Hz) 1st mode of oscillation

$m\omega_2^2 = 6 \Rightarrow \omega_2 = \sqrt{6}$ (Hz) 2nd mode

$m\omega_3^2 = 6 \Rightarrow \omega_3 = \sqrt{6}$ (Hz) 3rd mode

27.27 (a) The exact solution is

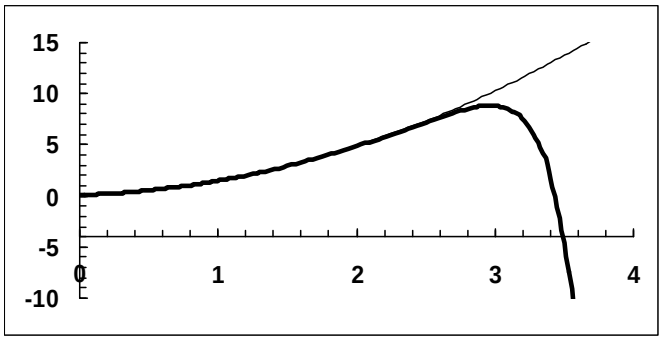
$$y = Ae^{5t} + t^2 + 0.4t + 0.08$$

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)

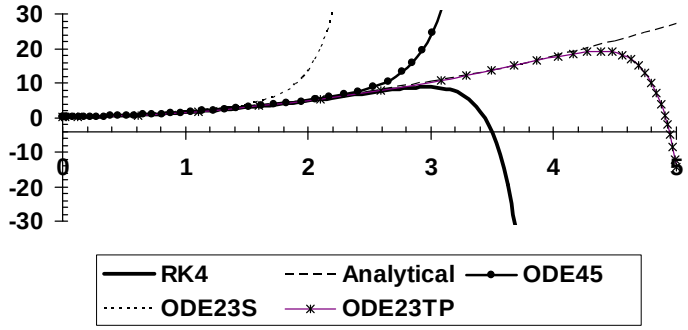
```
function yp=dy(t,y)
yp=5*(y-t^2);
>> tspan=[0,5];
>> y0=0.08;
>> [t,y]=ode45('dy1',tspan,y0);
```

(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

$$\frac{dT}{dx} = z \qquad \frac{dz}{dx} = -25$$

(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

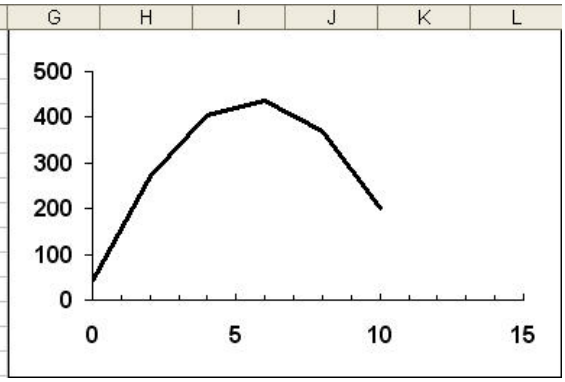
$z(0)$	–1	–0.5
$T(10)$	–1220	–1215

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:

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Prob 27.28 (Shooting method)**											
2												
3	**t**	**T**	**z**									
4	**0**	**40**	**141**		**RUN**							
5	**2**	**272**	**91**									
6	**4**	**404**	**41**									
7	**6**	**436**	**-9**									
8	**8**	**368**	**-59**									
9	**10**	**200**	**-109**									
10												
11												
12												
13												
14												



(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

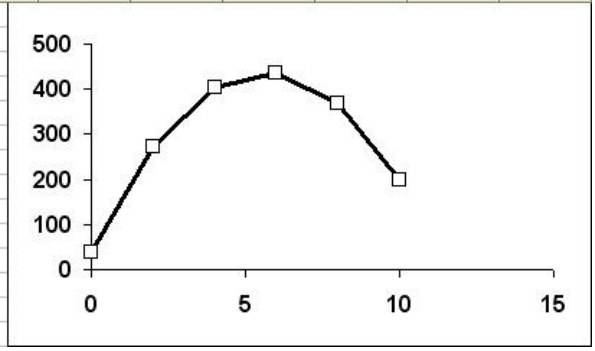
$$-T_{i+1} + 2T_i - T_{i-1} = 100$$

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:

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Prob 27.28 (Finite Difference)**											
2												
3	**x**	**y**										
4	**0**	**40**										
5	**2**	**272**										
6	**4**	**404**		**RUN**								
7	**6**	**436**										
8	**8**	**368**										
9	**10**	**200**										
10												
11												
12												
13												
14												



27.29 First, the 2nd-order ODE can be reexpressed as the following system of 1st-order ODE's

$$\frac{dT}{dx} = z \qquad \frac{dz}{dx} = -(0.12x^3 - 2.4x^2 + 12x)$$

(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

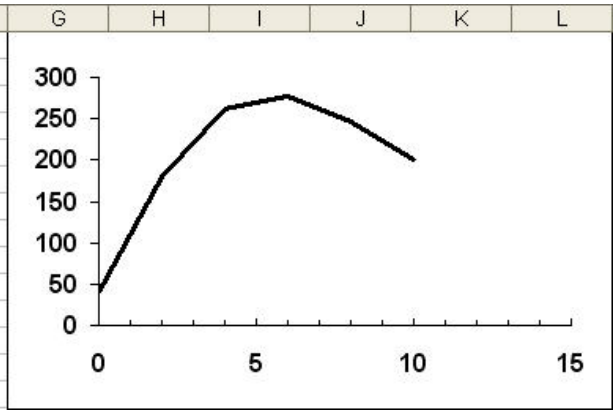
$z(0)$	–1	–0.5
$T(10)$	–570	–565

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:

	A	B	C	D	E	F
1	Prob 27.29 (Shooting method)					
2						
3	t	T	z			
4	0	40	76		RUN	
5	2	179.008	57.92			
6	4	261.056	23.52			
7	6	276.544	-6.08			
8	8	246.592	-21.28			
9	10	200	-24			
10						
11						
12						
13						
14						



(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

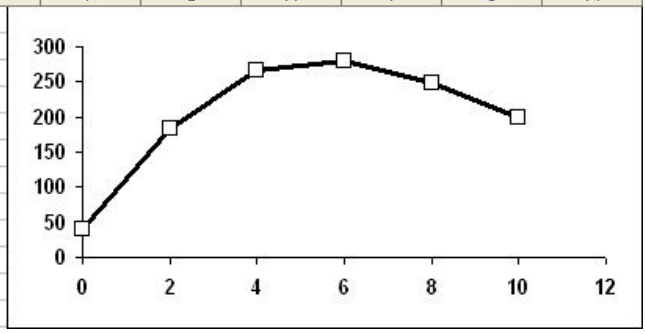
$$-T_{i+1} + 2T_i - T_{i-1} = \Delta x^2(0.12x_i^3 - 2.4x_i^2 + 12x_i)$$

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:

	A	B	C	D	E	F	G	H	I	J	K
1	Prob 27.29 (Finite Difference)										
2											
3	x	y									
4	0	40									
5	2	184.128									
6	4	266.816		RUN							
7	6	280.384									
8	8	247.872									
9	10	200									
10											
11											
12											



27.30 The second-order equation can be reexpressed as a pair of first-order equations:

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

A function can be developed to hold these equations,

```
function dy=prob2221sys(x,y)
g=9.81;c=12.5;m=70;
dy=[y(2);g-c/m*y(2)];
```

The solution was then generated with the following script. Note that we have generated a plot of all the shots as well as the analytical solution.

```
ti=0;tf=12;
za1=0;za2=50;xa=0;xb=500;
[t1,x1]=ode45(@prob2221sys,[ti tf],[xa za1]);
xb1=x1(length(x1));
[t2,x2]=ode45(@prob2221sys,[ti tf],[xa za2]);
xb2=x2(length(x2));
za=za1+(za2-za1)/(xb2-xb1)*(xb-xb1);
[t,x]=ode45(@prob2221sys,[ti tf],[xa za]);
plot(t,x(:,1),t1,x1(:,1),'--',t2,x2(:,1),'--')
disp('results:')
fprintf('1st shot:  za1 = %8.4g xb1 = %8.4g\n',za1,xb1)
fprintf('2nd shot:  za2 = %8.4g xb2 = %8.4g\n',za2,xb2)
fprintf('Final shot: za = %8.4g   x = %8.4g\n',za,x(length(x)))
fprintf('\n     t         x        dx/dt\n')
disp([t x])
```

The results are

```
results:
1st shot:    za1 =        0 xb1 =    387.7
2nd shot:    za2 =       50 xb2 =    634.8
Final shot:  za =    22.72   x =      500
     t         x        dx/dt
         0         0   22.7224
    1.2000   31.1280   28.9359
    2.4000   68.9674   33.9508
    3.6000  112.2237   37.9985
    4.8000  159.8520   41.2654
```

```
 6.0000  211.0091   43.9023
 7.2000  265.0143   46.0305
 8.4000  321.3183   47.7482
 9.6000  379.4776   49.1346
10.8000  439.1345   50.2536
12.0000  500.0000   51.1567
```

27.31 In order to include a derivative boundary condition at the left end of the rod, we have to introduce a node, T_0 for which the following finite difference equation can be written,

$$-T_{-1} + 2.04T_0 - T_1 = 0.8$$

As described on p. 785 of the text, a zero derivative at this node can be introduced by setting $T_{-1} = T_1$, which converts the equation to

$$2.04T_0 - 2T_1 = 0.8$$

The system of equations to be solved is therefore

$$\begin{bmatrix} 2.04 & -2 & 0 & 0 & 0 \\ -1 & 2.04 & -1 & 0 & 0 \\ 0 & -1 & 2.04 & -1 & 0 \\ 0 & 0 & -1 & 2.04 & -1 \\ 0 & 0 & 0 & -1 & 2.04 \end{bmatrix} \begin{Bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 0.8 \\ 0.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{Bmatrix}$$

The solution as generated with MATLAB is

```
>> A=[2.04 -2 0 0 0;-1 2.04 -1 0 0;0 -1 2.04 -1 0;0 0 -1 2.04 -1;0 0 0 -1 2.04];
>> b=[0.8 0.8 0.8 0.8 200.8]';
>> T=A\b;
>> T=[T; 200]'
T =
  136.7972  139.1331  146.2344  158.3851  176.0711  200.0000
>> x=[0:2:10]';
>> plot(x,T','o-')
```