

Matlab offers many facilities for problem solving. The following solutions are provided to questions that do not provide complete Matlab code. They are mainly intended to demonstrate features mentioned in the book. They are not unique, other more elegant and more efficient solutions may well exist.

```
% CHAPTER 1
% credit card
capital = 1000; APR = 17.5;
interest = capital * ((APR/12)/100 )

% Pythagoras
a = 12; b = 5;
c = sqrt(a^2 + b^2)

% Roots of quadratic
a = 3; b = -13; c = 4;
root1 = (-b + sqrt(b^2 - 4*a*c))/(2*a)
root2 = (-b - sqrt(b^2 - 4*a*c))/(2*a)

% Cosine rule
a = 5; b = 7;
c = sqrt(a^2 + b^2 - 2*a*b*cos(pi/6))

% Vectors and matrix
x = [ 4 10 -1 0]
y = [ -5.3 -2 0.9 1]'
B = [ 1 7.3 -5.6 2; 1.4 8 -3 0; -2 6.3 1 -2]

% roots function
roots([ 3 -13 4])

% assign values
a = B(1,2)
b = B(3,4)

% form vector
v1 = B(:,2)
v2 = B(1,:)

% vector norm
v = [3 4 5 -1]; sqrt(v*v')
norm(v)

% systems of equations
A = [ 2 1; 3 -2]; b = [5; 4]; x = A\b;
a = x(1)
c = x(2)
A = [3 2 -1; 2 -2 3; 1 -1 -1]; b = [3; 9; 2]'; x = A\b;
a = x(1)
c = x(2)
d = x(3)

% sum of even numbers 2..10
addnumbers = 0;
for i = 2:2:10;
```

```

    addnumbers = addnumbers + i;
end;
addnumbers

% sum of vector elements
b = [2.5 5 9 -11]; eltsun = 0;
for i = 1:4;
    eltsun = eltsun + b(i);
end;
eltsun

% approximation to square root of 2
x = 1;
for n = 1:6;
    x = x/2 + 1/x;
end;
x

% Fibonacci
F(1) = 0; % set first two
F(2) = 1; % Fibonacci numbers
for n = 2:20;
    F(n+1) = F(n) + F(n-1); % next Fibonacci number
    ratio = F(n + 1)/F(n);
end;
(1 + sqrt(5))/2 %compare with Golden ratio

% test the discriminant
a = 3; b = 2; c = 1; % test values
if (b^2 - 4*a*c) > 0; roots = 2
else if (b^2 - 4*a*c) < 0; roots = 0
    else
        roots = 1
    end
end;

% while loop, vector elements
v = [ 2 4 -4 3 0 1 2 0 1]; %test vector
elesum = 0; i = 1; % initialise variables
while v(i) ~= 0;
    elesum = elesum + v(i);
    i = i + 1; % increment the element index
end;
elesum

% square root of 2 to 4 decimal places
x = 1; % current estimate
diff = 1; % to let the next step proceed
while diff > 0.00005;
    xn = x/2 + 1/x; % find next estimate, xn
    diff = abs(xn - x);
    x = xn; % update current estimate
end;
x

% correct number, three attempts
n=input('Enter the number of data values \n');

```

```

attempt = 1; % first attempt
reply=input('Is that correct, answer Y or N\n','s');
while ~strcmp(reply,'Y') && attempt < 3;
n=input('Enter the number of data values\n');
attempt = attempt + 1; % another attempt
reply=input('Is that correct, answer Y or N\n','s');
end;
if strcmp(reply,'Y');
disp('Thank you');
else
disp('Incorrect number');
end;

% correct string, two attempts
teststring = 'ABC123';
reply=input('Enter the password \n','s'); % string input
attempt = 1; % first attempt
OK = strcmp(reply,teststring); set OK to true or not true
while ~OK && attempt < 2;
    reply=input('Try again\n','s');
    OK = strcmp(reply,teststring);
    attempt = attempt + 1; % another attempt
end;
if OK
disp('Password accepted');
else
disp('Incorrect password');
end;

% FtoC
function [c] = FtoC(f) % this code to be stored
    c = (5/9)*(f - 32); % in M-file FtoC.m

% plot y = sin(x), y = cos(x)
x = linspace(0,2*pi,40);
y = sin(x);
plot(x,y,'r'); % red sine curve
hold;
y = cos(x);
plot(x,y,'b'); % blue cosine curve

% format using \n
ratio = 22/7;
s = sprintf( ...
'pi to 6 decimals is %8.6f \n 22/7 is %8.6f',pi,ratio);
disp(s);

% Matrix display
A = [ 1 2 4.56; 3 4 5.0; 3 29 4.567];
s= sprintf('%3d %3d %6.2f\n%3d %3d %5.1f\n%3d %3d %6.3f' ...
,A');
disp(s);

% textscan
% assume the file mydata.txt is as listed below
% Node 27 ux1 1.0934 uy 3.0e+005

```

```

% Node 28 ux1 2.0934 uy 3.0123e+005
% Node 29 ux1 3.0934 uy 3.0e+005
% Node 30 ux1 4.0934 uy 3.456e+005
%
fid = fopen('mydata.txt') %open mydata for reading
data = textscan(fid, '%*s %*d %*s %f %*s %f')
ux1 = data{1} %retrieve ux1 values
uy = data{2} %retrieve uy values

% CHAPTER 2
% Exercise 2.3
A = [ 1 2 3 1; 2 4 -1 1; 1 3 -1 2; -3 -1 -3 2];
b = [8 1 1 -7]';
[L U] = lu(A);
y = L\b;
x = U\y;
A*x % compare with b

% Exercise 2.4
format long
A = [-0.6210 0.0956 -0.0445 0.8747 ;
      0.4328 -0.0624 8.8101 -1.0393;
      -0.0004 -0.0621 5.3852 -0.3897;
      3.6066 0.6536 0.8460 -0.2000]
b = [ 5.3814 -1.0393 -0.3897 -0.0005]'
A\b
A= single(A); b = single(b);
A\b
format short % restore the default value

% Exercise 2.9 (iterative solution)
x = 0.5*ones(10,1); % initial estimate
for k = 1:5;
    x(1) = 6*(1 - x(2));
    for i = 2:9;
        x(i) = 6*( 1 - x(i-1) - x(i+1));
    end;
    x(10) = 6*(1 - x(9));
    s = sprintf('p1=%3.2f,p2=%3.2f,p3=%3.2f,p4=%3.2f,p5=%3.2f',x);
    disp(s);
end;

% Exercise 2.10 (iterative solution)
p = zeros(1,5); % current estimate
diff = 1; % let the next statement proceed
while diff >= 0.005 % stopping condition
    q = p; % save current estimate
    p(1) = p(2)/2;
    p(2) = (p(3)+p(1))/2;
    p(3) = (p(4)+p(2))/2;
    p(4) = (p(5)+p(3))/2;
    p(5) = (1+p(4))/2;
    diff = abs(max(abs(p-q))); % largest p and q difference
end;
s = sprintf('p1=%3.2f,p2=%3.2f,p3=%3.2f,p4=%3.2f,p5=%3.2f',p);
disp(s);

```

```

% CHAPTER 3
% Exercise 3.2
low = 0; %initial lower bound
high = pi/2; %initial upper bound
% reduce the length of the interval until it is less than 0.00005
while high - low > 0.00005 ;
    mid = (high + low)/2 ; %mid point of current bounds
    if (low*sin(low)-cos(low))*(mid*sin(mid)-cos(mid)) ...
        < 0 %solution lies to the left
        high = mid;
    else %solution lies to the right
        low = mid;
    end;
    s = sprintf('solution lies in [%8.4f,%8.4f]',low, high);
    disp(s);
end;
% alternative solution using fzero
% function[result] = fsincos(x) % this code to be stored
% result = x*sin(x)-cos(x); % in M-file fsincos.m
solution = fzero(@fsincos, 1.0)

% Exercise 3.3
% assume function sq2(x) = x^2 -2 has been defined in M-file, sq2.m
low = 1;
high = 2;
xstar = 1; % let the next step proceed
while abs(sq2(xstar))>0.00005
    xstar = high - sq2(high)*(high - low)/(sq2(high) - sq2(low));
    if sq2(low)*sq2(xstar) < 0
        high = xstar;
    else
        low = xstar;
    end
end;
xstar

% Exercise 3.4
a = 1; % change to a = -1; b = -2;
b = 2; % to find the negative root
eps = 0.00005;
xstar = 1; % to let the next step proceed
while abs(sq2(xstar)) > eps % test using sq2
    xstar = b - sq2(b)*(a-b)/(sq2(a)-sq2(b));
    a = b;
    b = xstar;
    s = sprintf('current estimate %8.4f', xstar);
    disp(s);
end;

% Exercise 3.5
x = linspace(-3,3,100);
y = 4*x.*x.*(x-2) + 3*x -10;
plot(x,y);
% insert code from previous question, using a function such as
% function[polyvalue] = f(x)

```

```

% polyval = 4*x^3 -8*x^2 +3*x -10;
% stored in M-file, f.m
roots([4 -8 3 -10]); %check using Matlab function, roots

% Exercise 3.6
% Newton's method for finding  $a^p$ , solve  $x^{(1/p)} = a$ 
a = 2; p = 2/5; % test by evaluating  $2^{(5/2)}$ 
eps = 0.0000005; % to six decimal places
x = 1; % initial estimate
diff = 1; % let next step proceed
while diff > eps;
    x1 = x - (x^p - a)/(p*x^(p-1));
    diff = abs(x1-x);
    s = sprintf('current estimate %16.6f',x1);
    disp(s);
    x = x1;
end;
s = sprintf('Matlab value %16.6f', 2^(5/2));
disp(s)

% Exercise 3.7
% The following code to be stored in M-file , NewtonRaphson.m
% with code for a function fdash in M-file fdash.m
function[root] = NewtonRaphson(estimate, eps, limit)
x = estimate;
count = 0;
while diff > eps && count < limit
    if fdash(x) ~= 0
        root = x - f(x)/fdash(x);
        diff = abs(root-x);
        s = sprintf('latest estimate %16.8f',root);
        disp(s);
        x = root;
        count = count + 1;
    else
        s = sprintf('Method fails, zero derivative at %8.4f',x);
        disp(s);
        break;
    end;
end;

Exercise 3.8
% define M-file function f1 and f2 to evaluate
%  $x_1 \cos(x_2) + x_2 \cos(x_1) - 0.9$  (equation 1)
%  $x_1 \sin(x_2) + x_2 \sin(x_1) - 0.1$  (equation 2)
%
% Alternatively simple functions may be defined within
% a program using the @ symbol, thus
% f1 = @(x1,x2) x1*cos(x2) + x2*cos(x1) - 0.9;
% offers an alternative to using an M-file
%
x = [0 ; 1];
for i = 1:3
    % more iterations will be needed for greater accuracy
    % form the matrix of coefficients
    % define M-file functions diff1(i,j) to differentiate
    % equation i, i = 1,2 with respect to  $x_j$ , j = 1,2

```

```

A = [diff1(x,1) diff1(x,2) ; diff2(x,1) diff2(x,2)];
b = -[ f1(x) ; f2(x)];
d = A\b ;
x = x + d;
f1(x); % values of equations (1) and (2)
f2(x); % are expected to decrease
end;
% Matlab check
x0 = [0 1]';
options=optimset('Display','iter');
[x,fval] = fsolve(@sys2,x0,options)

```

### Exercise 3.10

```

x = [ 1 1 1]';
eps = 0.00005;
diff = eps + 1;
% define M-file functions f1,f2,f3 to evaluate
% 4(x1)^2 + 2x2 - 4
% x1 + 2(x2^2)+ 2x3 -4
% x2 + 2(x3^2)- 4
while (diff > eps);
    b = -[f1(x),f2(x),f3(x)]';
    % form the Jacobian
    Jmat = [ 8*x(1)    2    0; ...
            1    4*x(2)    2; ...
            0    1    4*x(3) ];
    d = Jmat\b;
    xlatest = x + d;
%compare latest estimates
    diff = norm(d);
    x = xlatest;
end;
x
% Matlab check
x0 = [1 1 1]';
options=optimset('Display','iter');
[x,fval] = fsolve(@sys,x0,options)

```

### % CHAPTER 4

#### % Exercise 4.1

```

T = [0:100:600];
Y = [ 200 190 185 178 170 160 150];
for i = 1:6
    diff1(i) = (Y(i+1) - Y(i))/100;
end;
for i = 1:5
    diff2(i) = (diff1(i+1) - diff1(i))/200;
end;
A = [ 7 sum(T); sum(T) T*T' ];
rhs = [ sum(Y); T*Y' ];
coeff = A\rhs
Ylinear = coeff(1) + coeff(2)*T
p = polyfit(T,Y,1)

```

#### % Exercise 4.2

```

%Gas problem 6 data points 5 intervals
%5 splines have 4 coefficients - need 20 equations.

```

```

% A(i,j) i = 1..20; j = 1..5;
x = [0.5 : 0.5 : 3];
y = [1.62 1 0.75 0.62 0.52 0.46];
% data 10 equations, numbers 1,2 5,6 9,10, 13,14 17,18
for j = 1:5; %data values, first two of each block
    k = (j-1)*4;
    b(k+1) = y(j);
    b(k+2) = y(j+1);
    for i = 1:4
        A(k + 1, k + i) = x(j)^(i-1);
        A(k + 2, k + i) = x(j+1)^(i-1);
    end;
end;
% equal derivatives
for j = 2 :5
    k1 = (j-2)*4; k2 = k1 + 4;
    for i = 1:3;
        % 1st derivatives equations 3,7,11,15
        A(k1 + 3, k1 + i + 1) = -i*x(j)^(i-1);
        A(k1 + 3, k2 + i + 1) = i*x(j)^(i-1);
    end;
    for i = 2:3;
        % 2nd derivatives equations 4,8,12,16
        A(k1 + 4, k1 + i + 1) = -i*(i-1)*x(j)^(i-2);
        A(k1 + 4, k2 + i + 1) = i*(i-1)*x(j)^(i-2);
    end;
end;
%end conditions
% 2a(3) + 6a(4)x = 0
A(19,3) = 2; A(19,4) = 6*x(1); b(19) = 0;
A(20,19) = 2; A(20,20) = 6*x(6); b(20) = 0;
p = A\b' % Spline coefficients (Table 4.9)

% Exercise 4.3
% use Spline coefficients, p from previous exercise
C = [ p(4:-1:1)'; p(8:-1:5)'; p(12:-1:9)'; p(16:-1:13)'; p(20:-1:17)'];
for interval = 1:5;
    xvalues = linspace(x(interval), x(interval+1), 20);
    poly = C(interval,:);
    yvalues = polyval(poly,xvalues);
    plot (xvalues,yvalues);
    if interval == 1; hold; end
end;

% Exercise 4.4
x=[0.5:0.5:3]; y=[1.62 1 0.75 0.62 0.52 0.46];
for i = 1:5;
    diff(1,i) = (y(i+1)-y(i))/(x(i+1) -x(i));
end ;
for i = 1:4;
    diff(2,i) = (diff(1,i+1)-diff(1,i))/(x(i+2) -x(i));
end ;
for i = 1:3;
    diff(3,i) = (diff(2,i+1)-diff(2,i))/(x(i+3) -x(i));
end ;
for i = 1:2;
    diff(4,i) = (diff(3,i+1)-diff(3,i))/(x(i+4) -x(i));
end ;

```

```

end ;
for i = 1:1;
    diff(5,i) = (diff(4,i+1)-diff(4,i))/(x(i+5) -x(i));
end ;
% coefficients appear in first column of diff
for i = 1:5;
    diff(1,i) = (y(i+1)-y(i))/(x(i+1) -x(i));
end ;
for j=2:5;
    for i = 1:6-j;
        diff(j,i) = (diff(j-1,i+1) - diff(j-1,i)) ...
            /(x(i+j)- x(i));
    end;
end;
diff

% Exercise 4.5
xaxis = linspace(0.5,3.0,40);
% evaluating the polynomial at the specified point
for i = 1:40
    pval = 1.62; xdiff = 1;
    for j = 1:5
        xdiff = xdiff*(xaxis(i) - x(j));
        pval = pval + xdiff* diff(j,1);
    end;
    yaxis(i) = pval;
end;
plot(xaxis,yaxis);
% plot over extended range
figure % open another plotting window
xaxis = linspace(0,5,40);
for i = 1:40
    pval = 1.62; xdiff = 1;
    for j = 1:5
        xdiff = xdiff*(xaxis(i) - x(j));
        pval = pval + xdiff* diff(j,1);
    end;
    yaxis(i) = pval;
end;
plot(xaxis,yaxis);

% Exercise 4.6
x = 0.5:0.5:3.0;
P(:,1) = [1.62 1 0.75 0.62 0.52 0.46]';
xstar = 1.75;
for j = 1:5;
    for i = 1:6-j;
        P(i,j+1) = ((xstar - x(i))*P(i+1,j) - (xstar - x(i+j))*P(i,j)) ...
            /(x(i+j) -x(i));
    end;
end;
P % Table 4.6 (in upper triangular form)

% Exercise 4.7
x = [ 0.197 0.139 0.068 0.0427 0.027 0.015 0.009 0.008];
v = [21.5 21 19 16.5 14.5 11 8.5 7];
A= [ x*x' -x*v' ; x*v' -v*v'];

```

```

rhs = [ v*(x.*x)'; (v.*v)*x'];
sol = A\rhs;
a = sol(1); b = sol(2);
for i = 1:8
    vx = a*x(i)/(b + x(i));
    s = sprintf('%11.3f %6.1f %11.2f',x(i),v(i),vx);
    disp(s);
end;

% CHAPTER 5
% Exercise 5.1
% The following code would be placed in M-file, trap.m
function [integral] = trap(x,f,n)
% Assume function values for the
% range of integration are stored in
% equally spaced values x(i),y(i) i=1..n
h = x(2)-x(1);
integral = h*(f(1)+f(n))/2;
for i = 2:n-1;
    integral = integral + h * f(i);
end;

%Exercise 5.2
for n = 2:14;
x = linspace(0,1,n);
y = x.*x; % test integral x^2 between 0 and 1, n points
est = trap(x,y,n); %estimated value
err = 1/3 - est; %actual error
p = -1/(6*(n-1)^2); %predicted error
s = sprintf( ...
'%2d, estimate %9.6f, actual err. %9.6f predicted %9.6f', ...
n,est,err,p);
disp(s);
end;

%Exercise 5.3
% The following code would be placed in M-file, Simpson.m
function [integral] = Simpson(x,f,n)
% Assume function values for the
% range of integration are stored in
% equally spaced values x(i),f(i) i=1..n
%check for n odd
if 2* floor(n/2) == n;
    disp('Error: number of mesh points must be odd');
    return;
else
    h = x(2)-x(1);
    integral = h*(f(1)+f(n))/3;
    for i = 3:2:n-1; % odd section
        integral = integral + 2 * (h/3) * f(i);
    end;
    for i = 2:2:n-1; % even section
        integral = integral + 4 * (h/3) * f(i);
    end;
end;
%
% End of function

```

```

%
for n = 3:2:10;
x = linspace(0,1,n);
y = x.*x; % test integral x^2 between 0 and 1
est = Simpson(x,y,n);
err = 1/3 - est;
p = 0; %predicted error
s = sprintf('%2d points, est. %12.6f, err. %12.6f %12.6f', ...
            n,est,err,p);
disp(s);
end;
%
for n = 3:2:10;
x = linspace(0,1,n);
y = x.*x.*x; % test integral x^3 between 0 and 1
est = Simpson(x,y,n);
err = 1/4 - est;
p = 0; %predicted error
s = sprintf('%2d points, est. %12.6f, err. %12.6f %12.6f', ...
            n,est,err,p);
disp(s);
end;
%
for n = 3:2:10;
x = linspace(0,1,n);
y = x.*x.*x.*x; % test integral x^4 between 0 and 1
est = Simpson(x,y,n); % estimate
err = 1/5 - est;      % actual error
h = x(2)- x(1);
p = - h^4*(24/180); % predicted error
s = sprintf('%2d, est. %10.6f, err. %10.6f predicted %10.6f', ...
            n,est,err,p);
disp(s);
end;

% Exercise 5.4
for n = 3:2:20;
x = linspace(0,1,n);
y = 1./(1 + x.*x);
S = Simpson(x,y,n);
q = quad(@f1,0,1); % Matlab quad function, assume function
tr = trap(x,y,n); % f1(x) = 1/(1+x^2) been defined in M-file f1.m
tz = trapz(x,y); % Matlab trapz function
s=sprintf('%2d, Simp.%,%9.4f, quad.%,%9.4f, trapz.%,%9.4f', ...
            n,S,q,tr,tz);
disp(s);
end;

% Exercise 5.5
% The following code would be placed in M-file, Gmesh.m
function [points] = Gmesh(n)
p(1) = {[1]};
p(2) = {[1 0]};
for i=1:n - 2;
p(i+2) = { ((2*i+1)/(i+1))* [p{i+1} 0] - (i/(i+1))*[0 0 p{i}]};
end;
points = roots(p{n});

```

```

% Exercise 5.6
x = Gmesh(6) % roots of Legendre polynomial, P_6
sign = 1;
for i = 1:5;
    A(i,:) = x.^(i-1);
    b(i) = (2/i)*sign;
    sign = 1 - sign;
end;
w = A\b' % weights

% Exercise 5.7
% continuing from the previous exercise
sum = 0;
for i = 1:5;
    sum = sum + w(i)* sin((pi/4)*(x(i)+ 1));
end;
sum = (pi/4)*sum

% Exercise 5.8
% The following code would be placed in M-file, Etrap.m
function [integral, error] = Etrap(a,b,f)
% assume f is defined in M-file f.m
x = linspace(a,b,2);
y = f(x);
I1 = trap(x,y,2);
x = linspace(a,b,3); % halve the interval
I2 = trap(x,y,3);
error = 4*(I2 - I1)/3;
integral = I1 + error;

% Exercise 5.9
% The following code would be placed in M-file, fact.m
function [Factorial] = fact(n)
    if n==0;
        Factorial = 1; %stopping condition
    else
        Factorial = n * fact(n-1);
    end;

% Exercise 5.10
% The following code would be placed in M-file, Atrap.m
function [Integral] = Atrap(a,b,f,eps)
m = a + b;
[i e] = Etrap(a,b,f);
if abs(e) < eps ;
    Integral = i;

else
    m = m/2; eps = eps/2;
    Integral = Atrap(a,(a+b)/2,f,eps)+ Atrap((a+b)/2,b,f,eps);
end;

% CHAPTER 6
% Exercise 6.1
% Assume the following function, fdash to evaluate f'(x)
% using a five-point formula has been established in M-file, fd.m

```

```

% function[res] = fdash(x,h)
% res = (1/(12*h))*(f(x-2*h)-8*f(x-h)+8*f(x+h)-f(x+2*h));
% and that a function, f(x) to evaluate x^2sin(1/x) has been
% established in M-file, f.m
x = 0.1; h = 0.5;
for i = 1:11
    f2 = (1/(12*h))*(fd(x-2*h,h)-8*fd(x-h,h)+8*fd(x+h,h)-fd(x+2*h,h));
    s = sprintf('h %f12.6, f''''(0.1) estimate %8.4f',h,f2);
    disp(s);
    h = h/2;
end;

% Exercise 6.2
% test values,
x = 0.345; h = 1.5567;
a = 1.4839; b = 2.4567; c = -56.43; d = 21.2; e = 0.34;
% 2-point applied to ax + b
f2 = (a*(x+h) + b - (a*x+b))/h;
e2 = a;
% 3-point applied to ax^2 + bx + c
f3 = (a*(x+h)^2 + b*(x+h) + c - (a*(x-h)^2 + b*(x-h) + c))/(2*h);
e3 = 2*a*x + b;
% 5-point applied to ax^4 + cx^3 + d^x + e
f5 = ...
( a*(x-2*h)^4 + c*(x-2*h)^3 + d*(x-2*h)+ e - ...
8*(a*(x-h)^4 + c*(x-h)^3 + d*(x-h)+ e) + ...
8*(a*(x+h)^4 + c*(x+h)^3 + d*(x+h)+ e) - ...
( a*(x+2*h)^4 + c*(x+2*h)^3 + d*(x+2*h)+ e) )/(12*h);
e5 = 4*a*x^3 + 3*c*x^2 + d;
s = sprintf('formula results %8.4f %8.4f %8.4f',f2,f3,f5);
disp(s);
s = sprintf('accurate results %8.4f %8.4f %8.4f',e2,e3,e5);
disp(s);

% Exercise 6.3
% assume test function x^2.sin(1/x) is established in M-file f.m
x = 0.25; h = 0.1;
d = 2*x*sin(1/x) - cos(1/x);
s = sprintf('correct value, %8.4f',d); disp(s);
s = sprintf ...
(' h, 3-point, 5-point, 1st one-sided, 2nd one-sided');
disp(s);
for i = 1:3
    % 3-point
    f3 = (f(x+h) - f(x-h))/(2*h);
    % 5-point
    f5 = (1/(12*h))*(f(x-2*h)-8*f(x-h)+8*f(x+h)-f(x+2*h));
    % one sided formula 1
    fm1 = (1/(2*h))*(-3*f(x) + 4*f(x+h) - f(x+2*h));
    % one sided formula 2
    fm2 = (1/(12*h))*(-25*f(x) + 48*f(x+h) - 36*f(x+2*h) + 16*f(x+3*h) ...
- 3*f(x+4*h));
    s = sprintf('%15.6f %8.4f %8.4f %8.4f %8.4f',h,f3,f5,fm1,fm2);
    disp(s);
    h = h/10;
end;

```

```

% Exercise 6.4(ii)
global n r
r = 1; % eps = 0.5e-6; f0 = zeros(1,10);
s = sprintf(' radius, derivatives of order 6..10');
disp(s);
for j = 1:5;
    if j<5 r = r/2;
    for n = 1:10;
        f0(n) = factorial(n)*(2/r^n)*real(quad(@chy,0,1,1.0e-12));
    end;
    s = sprintf('%7.4f %10.4f %12.4f %12.4f %12.4f %12.4f\n',r,f0(6:10));
    disp(s);
    if j==1; r = 0.5; end;
    if j==2; r = 0.25; end;
    if j==4; r =r/2; end;
end;
end;

% Exercise 6.5
syms f d x;
f = exp(x)/(sin(x)^3 + cos(x)^3);
display(f); % display the formula
x = 0;
d = diff(f); display(d); eval(d) % first derivative
for i = 1:4
    d = diff(d); eval(d) % derivatives of order 2..5
end;

% CHAPTER 7
% Exercise %7.1
figure
x = linspace(0,20,100);
axis ([0 20 0 10]);
y = 4*x;
plot(x,y);
hold;
y = 15-x;
axis ([0 20 0 20]);
plot(x,y);
y = (80-4*x)/9;
axis ([0 20 0 20]);
plot(x,y);
y = (10 - 5*x)/2;
axis ([0 20 0 20]);
plot(x,y,'r');
f = [-5 -2]; A = [ 1 1; 4 9; -4 1; -1 0; 0 -1];
b = [15 80 0 0 0];
gtext('feasible region'); % click on the display to insert
gtext('z = 10'); % the text
% solution using Matlab function
% change sign of resulting value for a maximum
[x, val] = linprog(f,A,b)

% Exercise 7.2(i)
figure

```

```

linspace(0,10,100);
axis ([0 10 0 10]);
y = 20 - 5*x;
plot(x,y);
hold;
y = (40 - 5*x)/2;
axis ([0 10 0 10]);
plot(x,y);
y = (18*x - 90)/5;
axis ([0 10 0 10]);
plot(x,y);
y = 10 -x ;
axis ([0 10 0 10]);
plot(x,y,'r');
f = [ -1 -1]; A = [-5 -1; -5 -2; 18 -5; -1 0; 0 -1];
b = [-20 -40 90 0 0];
[x, val] =linprog(f,A,b)

```

**% Exercise 7.3**

```

f = [ -200 -400];
A = [ 5 12; 10 7; 12 15];
b = [ 120 140 180];
[x, val] = linprog(f,A,b)

```

**% Exercise 7.4**

```

f = [ -200; -400];
A = [ 5 12; 10 7; 12 15];
b = [ 120; 140; 180];
[x val] = linprog(f,A,b)
% first sub-problem y <= 7
%f = [ -200 -400];
A = [5 12; 10 7; 12 15; 0 1; -1 0; 0 -1];
b = [120 140 180 7 0 0];
[x val] = linprog(f,A,b)
% second sub-problem y >= 8
A = [5 12; 10 7; 12 15; 0 -1; -1 0; 0 -1];
b = [120 140 180 -8 0 0];
[x val] = linprog(f,A,b)
%
%f = [ -200 -400];
% branch left x <= 6, y <= 7
A = [5 12; 10 7; 12 15; 1 0; 0 1; -1 0; 0 -1];
b = [120 140 180 6 7 0 0];
[x val] = linprog(f,A,b)
%
% x >= 7, y <= 7
A = [5 12; 10 7; 12 15; -1 0; 0 1; -1 0; 0 -1];
b = [120 140 180 -7 7 0 0];
[x val] = linprog(f,A,b)
%f = [ -200 -400];
% branch right x <= 4, y >= 8
A = [5 12; 10 7; 12 15; 1 0; 0 -1; -1 0; 0 -1];
b = [120 140 180 4 -8 0 0];
[x val] = linprog(f,A,b)
%
% x >= 5, y >= 8
A = [5 12; 10 7; 12 15; -1 0; 0 -1; -1 0; 0 -1];
b = [120 140 180 -5 -8 0 0];
[x val] = linprog(f,A,b)

```

```

%
%f = [ -200 -400];
%
%           x <= 4, y <= 8
A = [5 12; 10 7; 12 15; 1 0; 0 1; -1 0; 0 -1];
b = [120 140 180 4 8 0 0];
[x val] = linprog(f,A,b)
%
%           x <= 4, y >= 9
A = [5 12; 10 7; 12 15; 1 0; 0 -1; -1 0; 0 -1];
b = [120 140 180 4 -9 0 0];
[x val] = linprog(f,A,b)
%
%           x <= 2, y >= 9
%f = [ -200 -400];
A = [5 12; 10 7; 12 15; 1 0; 0 -1; -1 0; 0 -1];
b = [120 140 180 2 -9 0 0];
[x val] = linprog(f,A,b)
%
%           x <= 2, y <= 9
f = [ -200 -400];
A = [5 12; 10 7; 12 15; 1 0; 0 1; -1 0; 0 -1];
b = [120 140 180 2 9 0 0];
[x val] = linprog(f,A,b)

% Exercise 7.5
A = [ 5 12; 10 7; 12 15]
u = [ 1 2 4 8]
X = [5*u 12*u; 10*u 7*u; 12*u 15*u]
val = [-200*u -400*u]
b = [ 120 140 180]'
Z = bintprog(val,X,b);
Z(4:-1:1)' % x (binary)
Z(8:-1:5)' % y (binary)

% Exercise 7.6
Aequals = [ 2 0 4 1 0 0 ; 1 1 1 0 1 0; 0 -2 1 0 0 1; 0 0 0 0 1 0 ]
equals = [4 3 -2 0]
f = [ 0 0 0 0 1 -1]
A = eye(6); A=-A; A(5,5) = 0;
b = zeros(6,1);
f = [ 0 0 0 0 1 -1];
X = linprog(f,A,b,Aequals,equals)
% We now have a basic feasible solution
% x = [ 0 3 0 4 0 4 ]
% with which to start the simplex method

% Exercise 7.8
u = [1 2 4 8];
A = [5*u 12*u 12*u 0; 10*u 7*u 16*u 0; 12*u 15*u 10*u 0; ...
      0*u 1*u 0*u 16; 0*u 0*u 1*u -16];
b = [ 120; 140; 180; 16; 0];
f = [-200*u -400*u -550*u 0 ];
x = bintprog(f,A,b)
x1 = x(4:-1:1)' % x1 (binary)
x2 = x(8:-1:5)' % x2 (binary)
x3 = x(12:-1:9)' % x3 (binary)
d = x(13)

% Exercise 7.8
val = [ 1 1 1 1 1 1];

```

```

A = [ -1 -1 0 0 -1 -1 ; ...
      0 0 -1 0 -1 0 ; ...
      -1 0 -1 0 0 0 ; ...
      0 -1 0 -1 0 0 ; ...
      -1 0 0 0 0 -1 ; ...
      0 -1 -1 -1 0 -1 ; ...
      0 0 0 -1 -1 0 ]
b = [ -1 -1 -1 -1 -1 -1 -1];
x = bintprog(val,A,b) % 6 binary values d_1...d_6

% Exercise 7.9
% Store the variables in order
% 11 12 13 14 15 21 22 23 24 25 31..... 41..... 51.....
% variable (i,j) stored in position 5(i-1) + j
z = [ 0 44 45 35 32 ...
      44 0 93 72 68 ...
      45 93 0 57 66 ...
      35 72 57 0 36 ...
      32 68 66 36 0 ];
v = [ 0 0 0 0 0];
A = [0 1 1 1 1 v v v v;
      v 1 0 1 1 1 v v v;
      v v 1 1 0 1 1 v v;
      v v v 1 1 1 0 1 v;
      v v v v 1 1 1 1 0];
B = [0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 ; ...
      0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 ; ...
      0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 ; ...
      0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 ; ...
      0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 ];
C = [ A; B];
for i = 1:10
s = sprintf( '%d%d%d%d', C(i,:));
disp(s);
end;
b = ones(10,1);
[X value] = bintprog(z,[],[],C,b);
X(1:5)' % display solution in more readable form
X(6:10)'
X(11:15)'
X(16:20)'
X(21:25)'
value
% Extra if necessary
E = [0 0 -1 -1 -1 0 0 -1 -1 -1 v v v]; %depending on 1st solution
e = -1;
[X value] = bintprog(z, E,e,C,b);
X(1:5)' % display solution in more readable form
X(6:10)'
X(11:15)'
X(16:20)'
X(21:25)'
value

% CHAPTER 8
% Exercise 8.1

```

```

% The following code would be placed in M-file, golden.m
function [val min] = golden(fn,a,b,eps)
gamma = (-1 + sqrt(5))/2;
x = [a (a +(1-gamma)*(b-a)) (a + gamma*(b-a)) b];
for i =1:4; f(i) = fn(x(i)); end;
%check for monotonicity
while (f(1) > f(2) && f(2) > f(3) && f(3) > f(4) ) || ...
(f(1) < f(2) && f(2) < f(3) && f(3) < f(4) );
    disp(' not monotonic - extend boundary')
    if f(1) < f(2);
        a = a - 0.5;           %           val = a; min = f(1);
    else
        b = b + 0.5;           %           val = b; min = f(4);
    end;
    x = [a (a +(1-gamma)*(b-a)) (a + gamma*(b-a)) b];
    for i =1:4; f(i) = fn(x(i)); end;
end;
% end of check
while abs(f(2)-f(3)) > eps && abs(x(2)-x(3)) > eps;
x = [a (a +(1-gamma)*(b-a) ) (a + gamma*(b-a)) b];

for i =1:4; f(i) = fn(x(i)); end;
if f(1) > f(2) && f(2) < f(3);
    b = x(3);
end;
if f(2) > f(3) && f(3) < f(4);
    a = x(2);
end;
end;
val = (a + b)/2;
min = (f(1) + f(4))/2;

% Exercise 8.3
% The following code would be placed in M-file, Pline.m
function [val] = Pline(t)
global a b lambda1 lambda2
x = a + lambda1*t;
y = b + lambda2*t;
val = x + y + sqrt( (1/(x+y)^2)+ (y-x)^2);
%
% end of function definition
%
global a b lambda1 lambda2
a = 0.25; b = 1; lambda1 = 0.150845; lambda2 = -1.21704;
[t min] = golden(@Pline, -1, 1, 1.0e-8)

% Exercise 8.4
% Use the following form of Pline
function [val] = Pline(t)
global est grad
x = est(1)- t*grad(1);
y = est(2) - t*grad(2);
val = x + y + sqrt( (1/(x+y)^2)+ (y-x)^2);
%
% end of function definition
global est grad;
%
```

```

% Exercise 8.5
% The following code (a variant of Pline) would be placed
% in M-file, Hline.m
function [val] = Hline(t)
global est grad H
A = t*H*grad';
x = est(1)- A(1);
y = est(2)- A(2);
val = x + y +sqrt( (1/(x+y)^2)+ (y-x)^2);
%
% end of function definition
%
global est grad H;
est = [0.25 1] ; % initial estimate
% initial H matrix
H = [ 1 0; 0 1]; increment = 1; i = 1; plotxy = est;
disp(' Step x y P(x,y) Partial_x Partial_y')
while norm(increment,inf)> 0.1e-5
% obtain current function value and first partial derivatives
[f grad] = P(est);
s = sprintf('%2d %8.5f %8.5f %8.5f %8.5f %8.5f',i,est,f,grad);
i = i +1;
disp(s)
% find a local minimum of f(est - t H.grad)
[tvalue min] = fminsearch (@Hline, 0);
increment = -tvalue*H*grad';
est = est + increment';
% for plotting purposes save the estimates
plotxy = cat(1,plotxy,est);
% update H
[fnew gradnew] = P(est);
deltaf = (gradnew - grad)';
u = increment - H*deltaf;
alpha = 1/(u'*deltaf);
H = H + alpha*u*u';
end
% plotxy
plot(plotxy(:,1),plotxy(:,2),'-r');
axis( [ 0 1 0 1]);

% Exercise 8.6
% The following code would be placed in M-file, rankln.m
function [pos value] = rankln(P,estimate)
global est grad H myfun ;
myfun = P;
est = estimate;
n = max(size(estimate)); H = eye(n);
increment = 1; i = 1;
while norm(increment,inf)> 0.1e-5
% obtain current function value and first partial derivatives
[f grad] = P(est);
i = i +1;
% find a local minimum of f(est - t H.grad)
options = optimset('MaxIter',100000);
[tvalue min] = fminsearch (@searchline, 0,options);
increment = -tvalue*H*grad';

```

```

est = est + increment';
% update H
[fnew gradnew] = P(est);
deltaf = (gradnew - grad)';
u = increment - H*deltaf;
alpha = 1/(u'*deltaf);
H = H + alpha*u*u';
end
% The following code would be placed in M-file, searchline.m
function [val] = searchline(t)
global est grad myfun
A = t*H*grad';
[ val g] = myfun (est - A');
%
% End of general function definitions
%
% The following code would be placed in M-file, fxy.m
function [val grad ] = fxy(xy)
global sigma
x = xy(1); y =xy(2);
val = x^2 + x*y -1 + sigma*((x + y^2 - 2)^2);
grad = [(2*x + y + 2*sigma*(x + y^2 -2)) ...
        (x + 4*y*sigma*(x + y^2 -2))];
%
% end of function definition for Exercise 8.6
%
global sigma
estimate = [ 1 1];
disp('          sigma          x          y          f(x,y)  (x + y^2 -2)^2');
for i = 1:5;
    sigma = 10^(i-1);
    [ est value ] = rankln(@fxy,estimate);
    pen = (est(1) + est(2)^2 - 2)^2;
    s = sprintf(' %12d %8.4f %8.4f %8.4f %10.1e',sigma, est, value,
pen);
    disp(s);
end

% Example 8.7
% Define the function Lagrange in M-file, Lagrange.m
% with global variables sigma mu1 mu2
%
% Note that the following solves the problem as
% stated in the book. However Table 8.15 represents the
% solution with the constraints  $3(x_2)^2 + 2(x_3) - 1 = 0$ 
% and  $3(x_1) - x_2 + 4x_3 - 2 = 0$ 
%
global sigma mu1 mu2
disp(' sigma      mu1      mu2      x1      x2      x3      f(x1,x2,x3)')
Q = @Lagrange;
sigma =1; mu1 = 0; mu2 = 0;
for i = 1:3
    sigma = 2*sigma; xold = [ 0 0 0 ]; x = [ 1 1 1 ];
    while abs(max(xold -x)) > 1.0e-7
        xold = x;
        options = optimset('TolX',1e-8,'TolF',1e-8);
        x = fminsearch(Q,xold);
    end
end

```

```

        f = x(1)^4 + x(2) + x(3);
        mu1 = mu1 + 2*sigma*(x(1) + 3*x(2)^2 + 2*x(3) - 1);
        mu2 = mu2 + 2*sigma*(3*x(1) - x(2) + 4*x(3) - 2);
    end;
    s = sprintf(' %4d%9.5f %8.5f %8.5f %8.5f %8.5f %8.5f',...
        sigma, mu1,mu2, x, f); disp(s);
end;

% Exercise 8.8
% The following code would be placed in M-file, expab.m
function [val grad] = expab(xy)
    x = [0 1 2 3 4 5 6 7]; f = [ 2.5 1.7 1.3 1 0.8 0.6 0.4 0.2];
    a = xy(1); b = xy(2);
    res = a*(exp(-b*x))-f;
    sum1 = 2*sum(res.*exp(-b*x));
    sum2 = res*res';
    sum3 = 2*a*sum(res.*(-x.*exp(-b*x)));
    val = sum2;
    grad = [ sum1 sum3];
    %
    % end of function definition
    %
    [pos val] = rankln(@expab,[2.5 1]);
    x = [0 1 2 3 4 5 6 7]; f = [ 2.5 1.7 1.3 1 0.8 0.6 0.4 0.2];
    plot(x,f,'*');
    hold;
    xd = linspace(0,7,100);
    yd = pos(1)*exp(-pos(2)*xd);
    plot(xd,yd,'-b');
    disp(' x          data f(x)          least squares value')
    for i = 1:8;
        g = pos(1)*exp(-pos(2)*x(i));
        s = sprintf('%2d %11.1f %20.2f',x(i),f(i),g); disp(s);
    end;

% Exercise 8.9
% The following code would be placed in M-file, expab.m
function [val grad] = expab(xy)
    x = xy(1); y = xy(2);
    d1 = x*cos(y) + y*cos(x) - 0.9;
    d2 = x*sin(y) + y*sin(x) - 0.1;
    val = d1^2 + d2^2;
    grad = [ (2*d1*(cos(y) - y*sin(x)) + 2*d2*(sin(y) + y*cos(x))) ...
        (2*d1*(-x*sin(y) + cos(x)) + 2*d2*(x*cos(y) + sin(x))) ];
    %
    % end of function definition
    [pos val] = rankln(@expab,[0 1]) % using the rankln function
    [pos1 val1] = fminsearch(@expab,[0 1]) % using the Matlab function

% CHAPTER 9
% Exercise 9.1
% Using the following code in M-file, eul.m
% function next = eul(f,x,y, h )
%     next = y + h*f(x,y);
% and a function, f(x,y) in M-file, f.m to return 3x - y + 8
y = 3; h = 0.1; %initial y-value and step-length

```

```

disp('      x      Euler y      correct y      absolute error');
for i = 1:6
    x = (i-1)*0.1;
    correct = 3*x - 2*exp(-x) + 5; % correct y-value
    error = abs(y-correct);
    s = sprintf(' %5.1f %10.5f %10.5f %10.5f %10.5f',x,y,correct,error);
    disp(s);
    y = eul (@f,x,y, h) ; % Euler y-value
end;

% Exercise 9.2
% Using the following code in M-file, RK2.m
% function [next] = RK2(f,x,y,h)
%     k1 = h * f(x,y);
%     k2 = h * f(x + h/2, y + k1/2);
%     next = y + k2;
% and a function, f(x,y) in M-file, f.m to return 3x - y + 8
y = 3; h = 0.1;
disp('      x      Runge-Kutta y      correct y      absolute error');
for i = 1:6
    x = (i-1)*0.1;
    correct = 3*x - 2*exp(-x) + 5; % correct y-value
    error = abs(y-correct);
    s = sprintf(' %5.1f %10.5f %10.5f %10.5f %10.5f',x,y,correct,error);
    disp(s);
    y = RK2 (@f,x,y, h) ; % Euler y-value
end;

% Exercise 9.3
% part(i)
% As above using the following code in M-file, RK4.m
function [next] = RK4(f,x,y,h)
    k1 = h*f(x,y);
    k2 = h*f(x + h/2, y + k1/2);
    k3 = h*f(x + h/2, y + k2/2);
    k4 = h*f(x + h, y + k3);
    next = y + (1/6)*(k1 + 2*k2 + 2*k3 + k4);
% part (ii)
% Similar to (i) but with function, f(x,y) modified to
% evaluate -9.81 - 0.2*y^2 and repeated with shorter step-length

% Exercise 9.4
% set functions fxy(x,y) to evaluate sin(x) + y;
% and fw(x,w) to -w + cos(x) in M-files fxy.m and fw.m
% function[result] = fxy(x,y)
h = 0.1; w0 = 0; y0 = 0; x= 0;
disp('      x      w      y');
while x < 0.5;
    % solve dw/dx = sin(x) + y to give w
    w = w0 + RK4(@fxy,x,y0,h)-y0;
    % solve dy/dx = -w + cos(x) to give y
    y = y0 + RK4(@fwx,x,w0,h)-w0;
    x = x + h;
    s = sprintf('%10.1f%10.5f%10.5f',x,w,y); disp(s);
    w0 = w; y0 = y;
end;
% repeat for shorter step-lengths

```

```

% Exercise 9.6
% Finite Difference example
% set function funxy(x,y) to evaluate 2y;
% set function funxz(x,z) to z (=dy/dx);
h=0.0001; y0 = 1; x =0; z0 = -1.25643;
disp('      x      y      z (=dy/dx)');
s = sprintf('%10.5f %10.5f %10.5f',x,y0,z0); disp(s);
while x < 1.0 + h;
    % solve dz/dx = 2y to give z
    z = z0 + RK4(@funxy,x,y0,h) - y0;
    % solve dy/dx = z to give y
    y = y0 + RK4(@funxz,x,z0,h) - z0;
    x = x + h;
    % print at selected x values
    if abs(x-0.2)<h/2 || abs(x-0.4)<h/2 || abs(x-0.6)<h/2 ...
        || abs(x-0.8)<h/2 || abs(x-1.0)<h/2;
        s = sprintf('%10.5f %10.5f %10.5f',x,y,z); disp(s);
    end;
    y0 = y; z0 = z;
end;

% Exercise 9.7
h = 0.1; n =10; j = 1;
for k = 1:4; % four step lengths
v = ones(n,1); w = ones(1,n - 1);
A = diag(w,-1) + diag( -2*(1+h^2)*v,0) + diag(w,1);
A(n,n) = -(1+h^2);
b = zeros(n,1); b(1) = -1;
x = A\b; % current solution
for i = 1:10;
    res(i,k) = x(i*j); % store solutions in columns
end;
h = h/2; n = n *2; j=j*2;
end;
res

% CHAPTER 10
% Exercise 10.3% Exercise 10.3
% largest eigenvalue (absolute value)
A= [ 2 1 -1; 4 -3 8; 1 0 2];
x= ones(3,1); lambda = 1; lambda1 = 0;
% to allow for eigenvectors changing sign
% test for convergence using the eigenvector
while abs(lambda - lambda1) > 0.00005
    v = A*x; lambda1 = lambda;
    lambda = max(abs(v));
    x = v/lambda;
end;
s = sprintf('%8.4f',lambda); disp(s);
% smallest eigenvalue (absolute value)
x= ones(3,1); lambda = 1;
x = (lambda*A)\x; % fix to start the while loop
lambda = max(abs(x)); lambda1 = 0;
while abs(lambda - lambda1) > 0.0000005
    lambda1 = lambda;
    x = (lambda*A)\x;
end;

```

```

        lambda = max(abs(x));
    end;
    lambda = 1/lambda;
    s = sprintf('%8.4f',lambda); disp(s);
    % eigenvalue closest to -0.5
    x= ones(3,1); lambda = 1;
    B = A + 0.5*eye(3);
    x = (lambda*B)\x; % fix to start the while loop
    lambda = max(abs(x)); lambda1 = 0;
    while abs(lambda - lambda1) > 0.0000005
        lambda1 = lambda;
        x = (lambda*B)\x;
        lambda = max(abs(x));
    end;
    lambda = (1/lambda)-0.5;
    s = sprintf('%8.4f',lambda); disp(s);

% Exercise 10.4
A = [ 4 -1 2 1; 1 1 6 3; 0 0 -3 4; 0 0 0 7];
Aorig = A;
while abs(A(2,1)) > 1.0e-6;
% solve  $-A(1,1)*s + A(2,1)*c = 0$ ;  $s^2 + c^2 = 1$ ;
s = 1/sqrt(1 + (A(1,1)/A(2,1))^2);
c = s * A(1,1)/A(2,1);
Q = [ c s 0 0; -s c 0 0; 0 0 1 0; 0 0 0 1];
Qinverse = [ c -s 0 0; s c 0 0; 0 0 1 0; 0 0 0 1];
A = Q*A*Q'; disp(A);
end;
%eigenvalues
e = diag(A); disp('Eigenvalues'),disp(e);
%eigenvectors
for i = 1:4
X = Aorig - e(i)*diag(ones(1,4));
[1 u] = lu(X); det(X);
if i ~= 3;
b = [ 0 0 0 1]';
u(4,4) = 1;
x = u\b; z = (x/norm(x))';
end;
if i == 3;
R = [u(1,1:2); u(2,1:2) ]; b = -[ u(1,3); u(2,3)];
x = [R\b ; 1; 0]; z = (x/norm(x))';
end;
disp('eigenvector'); disp(z);
end;

% Exercise 10.5
A = [3 5 -4 4; -1 -3 1 -4; 4 -2 3 5; 3 -5 5 4];
Aorig = A;
for i = 1:20 %Main
Inv = diag(ones(4,1));
% knock out A(4,1)
% solve  $-A(3,1)*s + A(4,1)*c = 0$ ;  $s^2 + c^2 = 1$ ;
s = 1/sqrt(1 + (A(3,1)/A(4,1))^2);
c = s * A(3,1)/A(4,1);
Q = [ 1 0 0 0; 0 1 0 0; 0 0 c s; 0 0 -s c];
Qinv = [ 1 0 0 0; 0 1 0 0; 0 0 c -s; 0 0 s c];
Inv=Inv*Qinv;

```

```

Q * Qinv;
A = Q*A;
% knock out A(3,1)
% solve -A(2,1)*s + A(3,1)*c = 0; s^2 + c^2 = 1;
s = 1/sqrt(1 + (A(2,1)/A(3,1))^2);
c = s * A(2,1)/A(3,1);
Q = [ 1 0 0 0; 0 c s 0; 0 -s c 0; 0 0 0 1];
Qinv = [ 1 0 0 0; 0 c -s 0; 0 s c 0; 0 0 0 1];
Inv = Inv*Qinv;
Q * Qinv;
A = Q*A;
% knock out A(2,1)
% solve -A(1,1)*s + A(2,1)*c = 0; s^2 + c^2 = 1;
s = 1/sqrt(1 + (A(1,1)/A(2,1))^2);
c = s * A(1,1)/A(2,1);
Q = [ c s 0 0; -s c 0 0; 0 0 1 0; 0 0 0 1];
Qinv = [ c -s 0 0; s c 0 0; 0 0 1 0; 0 0 0 1];
Inv = Inv*Qinv;
Q * Qinv;
A = Q*A;
% knock out A(4,2)
% solve -A(3,2)*s + A(4,2)*c = 0; s^2 + c^2 = 1;
s = 1/sqrt(1 + (A(3,2)/A(4,2))^2);
c = s * A(3,2)/A(4,2);
Q = [ 1 0 0 0; 0 1 0 0; 0 0 c s; 0 0 -s c ];
Qinv = [1 0 0 0; 0 1 0 0; 0 0 c -s; 0 0 s c ];
Inv = Inv*Qinv;
Q * Qinv;
A = Q*A;
% knock out A(3,2)
% solve -A(2,2)*s + A(3,2)*c = 0; s^2 + c^2 = 1;
s = 1/sqrt(1 + (A(2,2)/A(3,2))^2);
c = s * A(2,2)/A(3,2);
Q = [ 1 0 0 0; 0 c s 0; 0 -s c 0; 0 0 0 1];
Qinv = [1 0 0 0; 0 c -s 0; 0 s c 0; 0 0 0 1];
Q * Qinv;
Inv = Inv*Qinv;
A = Q*A;
% knock out A(4,3)
% solve -A(3,3)*s + A(4,3)*c = 0; s^2 + c^2 = 1;
s = 1/sqrt(1 + (A(3,3)/A(4,3))^2);
c = s * A(3,3)/A(4,3);
Q = [ 1 0 0 0; 0 1 0 0; 0 0 c s; 0 0 -s c ];
Qinv = [1 0 0 0; 0 1 0 0; 0 0 c -s; 0 0 s c ];
Q * Qinv;
Inv = Inv*Qinv;
A = Q*A;
A = A*Inv;
s = sprintf(' %7.4f %7.4f %7.4f %7.4f',diag(A)); disp(s);
end;
% using Matlab
A = [3 5 -4 4; -1 -3 1 -4; 4 -2 3 5; 3 -5 5 4];
[Q R] = qr(A);
Q*R % check A = Q*R, Q'*Q = I

% CHAPTER 11
% Exercise 11.1

```

```

% example based on marks for programming and engineering
e = [ 15 21 22 30 33 37 39 43 43 47 50 56 60 63 67 70 76 79];
p = [ 41 42 48 47 49 49 50 52 52 14 34 35 37 55 60 67 29 63];
[n cen] = hist(p,10);
bar(cen,n,'w')
colormap([ 1 1 1]); %change default blue fill to white
axis([10 100 6])
set(gca,'YTick',0:1:6)
gtext('Programming')
xlabel('Mark')
ylabel('Frequency')
disp('mean, programming'); disp(mean(p));
disp('standard deviation, programming'); disp(std(p));
disp('programming/engineering');
X = cov(p,e); disp('covariance matrix'); disp(X);
cc = X(1,2)/(sqrt(X(1,1)*X(2,2)));
disp('correlation coefficient'); disp(cc);

% Exercise 11.2
x = 0:100:600;
y = [200 190 185 178 170 160 150];
n = 7;
SX = sum(x); SY = sum(y); SXX = sum(x.*x); SXY = sum(x.*y);
A = [n SX; SX SXX];
b = [SY; SXY];
c = A\b;
s = 0; disp(' data x and y   least squares y');
for i = 1:n
    est(i) = c(1) + c(2)*x(i);
    s = s +(y(i) - est(i))^2;
    str = sprintf('%6.2f  %6.2f  %8.4f',x(i),y(i),est(i));
    disp(str);
end;
s = sqrt(s/(n-2)); sb = s/sqrt(SXX - n*(SX/n)^2);
str = sprintf('standard error %6.4f, sigma_b %6.4f',s,sb);
disp(str);

% Exercise 11.3
seed = 123456789; mult = 7^5; m = 2^31-1; % set the generator
n(1) = seed;
for i = 1:19;
    n(i+1) = mod(mult*n(i),m);
end;
q = n/m % range [0 1]
q1 = (2*q -1) % range [-1 1]
q2 = floor( (100-5)*q1 + 5 ) % integer range [5 100]

% Exercise 11.4
seed = 123456789; mult = 7^5; m = 2^31-1; % set the generator
disp('power of 10   estimate')
for index = 5:9
    in = 0; out = 0;
    for n = 1 : 10^index
        seed = mod(mult*seed,m);
        x = seed/m; % range [0 1]
        seed = mod(mult*seed,m);
        y = seed/m; % range [0 1]
    end
end

```

```

    if sqrt(y^2+x^2) < 1
        in = in + 1;
    else
        out = out + 1;
    end;
end;
ratio = in/(in+out); f= sprintf('%1d %18.4f',index,ratio);
disp(f);
end;

% Exercise 11.5
count=0; count1=0; count2=0; count3=0; nocount=0;
disp(' power of 10      Area(i)      Area(ii)      Area(iii)');
for j = 5:9;
    n = 10^(j);
while count < n
    count = count + 1;
    x = random('unif',-3.0,3.0);
    y = random('unif',0,1.0);
    if y < exp( -(2*x^2));
        if abs(x) <= 0.5; count3 = count3 + 1; end;
        if abs(x) <= 1.0; count2 = count2 + 1; end;
        if abs(x) <= 1.5; count1 = count1 + 1; end;
        if abs(x) > 1.5; nocount = nocount + 1; end;
    else;
        nocount = nocount + 1;
    end;
end;
Ar = 6; % area of rectangle
A0 = Ar*nocount/count; A1 = Ar*count1/count;
A2 = Ar*count2/count; A3 = Ar*count3/count;
sum = A0 + A1;
s = sprintf('%8d %11.2f %11.2f%11.2f ',j,A1,A2,A3); disp(s);
end;

```

