# Optimization in Practice with MATLAB:

## for Engineering Students and Professionals

## Solution Manual

ACHILLE MESSAC

# Table of Contents

CHAPTER 1

# MATLAB as our computation tool Solutions

## 1.1

(d) See Fig. 1.1.

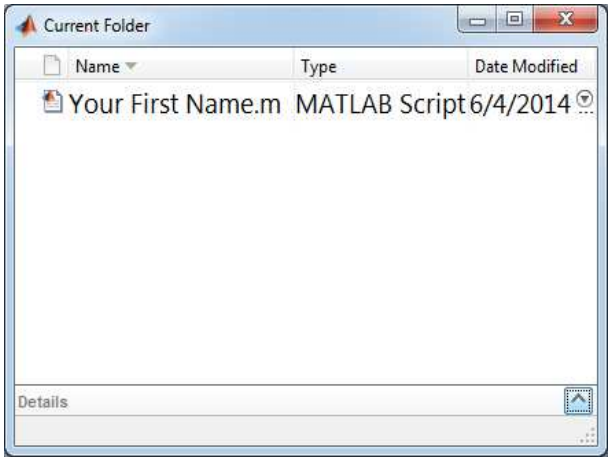


Figure 1.1: Screen shot for Problem 1.1(d)

---

## 1.2

(a) `>> z = [1/6 3/4 2/9]`

(b) `format` command can be used to change the format and the spacing of the numeric output display in the command window.

(c) To print the complete contents of the Command Window, select `File` → `Print`. To print only a selection, first highlight the selection in the Command Window and then select `File` → `Print Selection`.

(d) `format rat, format short`, `format long, format short e` are some format commands. More options are available in MATLAB help. `format loose, format compact` can be used to change spacing of the printed output.

(e)
```
% clear workspace
clear;
clc;
% define array
z = [1/6 3/4 2/9];
% show ans in the command window
display(z);

format compact
display(z);

format rat
display(z);

format long
display(z);

format short e
display(z);
```

---

## 1.3

(a) A = [1 2; 3 4; 5 6; 7 8]
B = [1 2 3 4; 5 6 7 8]

(b) Highlight the desired lines, right click and select `Create M-file`. See the M file in part (f).

(c)
```
clear;clc;
A = [
     1     2;
     3     4;
     5     6;
     7     8
     ];
B = [
     1     2     3     4;
     5     6     7     8
     ];
display(A*B);
display(B*A);
```
Note that $AB \neq BA$. Matrix multiplication is generally not commutative.

(e) See M file in part (f).

```
A = [
     0     1;
     9    -1;
     3     2;
     1     0
```

```
    ];
B = [
     3    -1     0     2;
     5     4     2     1
    ];
disp(A*B);
disp(B*A);
```

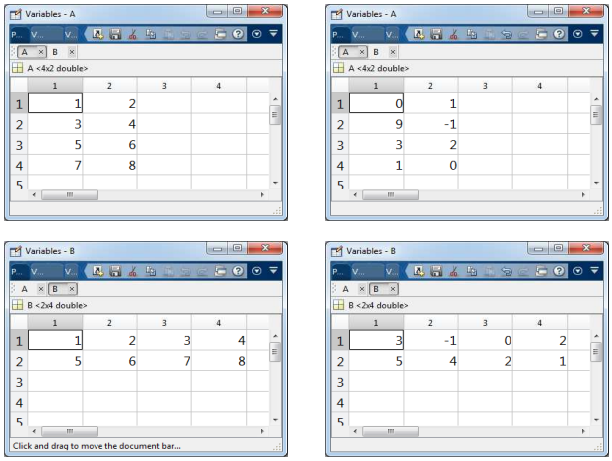


Figure 1.2: Screen shot for 1.3(f)

(f) See Fig. 1.2 for screen shots of the array editor before and after modification. The M-file before modification is given below:

```
A=[1 2;3 4;5 6;7 8]
B=[1 2 3 4;5 6 7 8]
AB=A*B
BA=B*A
```

The M-file after modification of $A$ and $B$ is given below:

```
A=[0 1;9 -1;3 2;1 0]
B=[3 -1 0 2;5 4 2 1]
AB=A*B
BA=B*A
```

---

## 1.4

(a)
```
clear;clc;close all
n = 100;
x = 0:10/n:10;
```

(b)
```
y1 = sin(x);
y2 = exp(x);
y3 = x.^2+2*x+1;
y4 = x.^3+5;
```

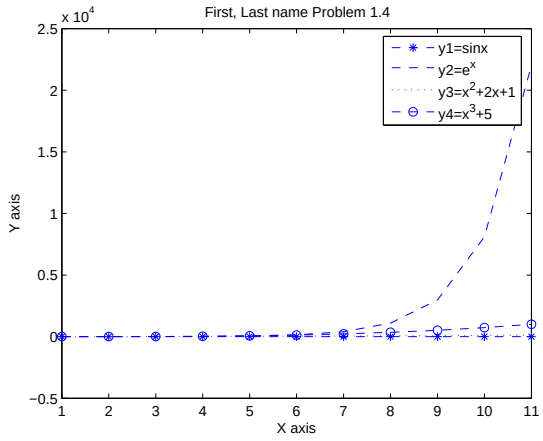


Figure 1.3: Plot of the 4 functions for Problem 1.4 (e)

(c) `save loadmat.mat`

(d) `load loadmat.mat`

(e)
```
h = sprintf('%1.0f points', n);
figure(1);
hold on;
plot(x,y1,'o-');
xlabel('x');
ylabel('y1');
legend(h);
title('Plot of y1=sinx');
figure(2);
plot(x,y2,':');
xlabel('x');
ylabel('y2');
legend(h);
title('Plot of y2=e^x');
figure(3);
plot(x,y3,'-');
xlabel('x');
ylabel('y3');
legend(h);
title('Plot of y3=x^2+2*x+1');
figure(4);
plot(x,y4,'*-');
xlabel('x');
ylabel('y4');
legend(h);
title('Plot of y4=x^3+5');
```
See the plot in Fig. 1.3.

(f) The number of elements in arrays `x` and `y` govern the smoothness of the curve. As we include more and more points in the array, the curve obtained will be smoother. See the plots in Figure

1.4.

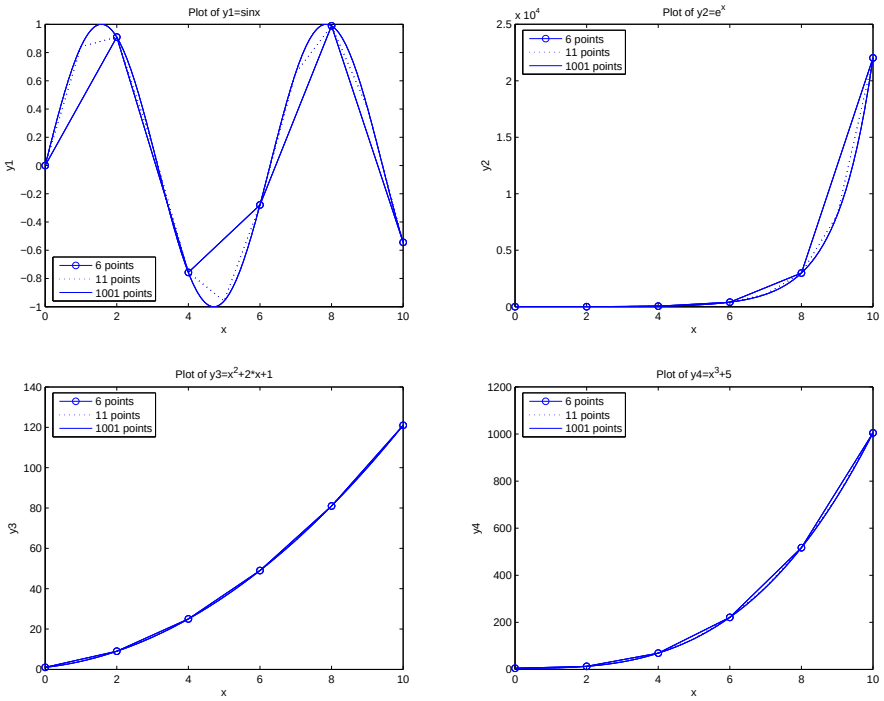


Figure 1.4: Problem 1.4

---

## 1.5

(a)
```
A=[2 4 6; 3 5 1; 7 5 9];
B=[1 3 6];
C=[5; 7; 2; 0];
F=[3;17;12;-2];
H=[5 7 4 -2; 3 12 -6 14];
D=[A;B];
```

```
D =
     2     4     6
     3     5     1
     7     5     9
     1     3     6
```

(b)
```
E=[D C];
```

```
E =
     2     4     6     5
     3     5     1     7
     7     5     9     2
     1     3     6     0
```

(c)
```
det(E)
ans =
   470
```

(d)
```
inv(E)
ans =
   -0.0702   -0.0064    0.1979   -0.2255
   -0.4191    0.3255   -0.0915    0.5021
    0.2213   -0.1617    0.0128   -0.0468
    0.2979   -0.0638   -0.0213   -0.2553
```

(e)
```
E'
ans =
     2     3     7     1
     4     5     5     3
     6     1     9     6
     5     7     2     0
```

(h)
```
E*E
ans =
    63    73   100    50
    35    63    74    52
    94   104   140    88
    53    49    63    38
```

```
F*F
```

not possible, matrix dimensions do not agree.

```
H*H
```

not possible, matrix dimensions do not agree.

```
E*F
ans =
   136
    92
   210
   126
```

```
F*E
```

not possible, matrix dimensions do not agree.

```
H*E
ans =
    57    69    61    82
    14    84    60    87
```

```
E*H
```

not possible, matrix dimensions do not agree.

```
F*H
```

not possible, matrix dimensions do not agree.

```
H*F
ans =
   186
   113
```

## 1.6

```
clear;clc;
A=[2 4 6; 3 5 1; 7 5 9];
[V, e] = eig(A);
display(e);     % eigenvalue (diagonal entries)
display(V);     % eigenvector (per column)
```

The diagonal elements of `d` denote the eigen values, and the columns of `v` denote the eigen vectors of `A`.

---

## 1.7

(a)
```
clear;clc;
A=[12 14 16 40; 32 15 11 1; 7 25 19 10];
B=[9 1 36 4; 19 0 -31 2];
C=[7; 5; 7; 2; 0];
F=[16; 3; 17; 12;-2];
H=[5 7 4 -2 -1; -9 3 12 -6 14];
D=[A;B];
D =
    12    14    16    40
    32    15    11     1
     7    25    19    10
     9     1     3     6
    19     0   -31     2
```

(b)
```
>> E=[D C]
E =
    12    14    16    40     7
    32    15    11     1     5
     7    25    19    10     7
     9     1     3     6     2
    19     0   -31     2     0
```

(c)
```
>> det(E)
ans =
     -676510
```

(d)
```
>> inv(E)
ans =
    0.0152    0.0482   -0.0354   -0.0496   -0.0016
    0.0355    0.0497    0.0003   -0.2495    0.0120
    0.0122    0.0304   -0.0236   -0.0360   -0.0331
    0.0450    0.0132   -0.0297   -0.0868    0.0013
   -0.2394   -0.3269    0.2838    1.1622    0.0468
```

(e)
```
>> E'
ans =
    12    32     7     9    19
    14    15    25     1     0
    16    11    19     3   -31
```

```
    40     1    10     6     2
     7     5     7     2     0
```

(h)
```
>> E*E
ans =
        1197         818         553         908         346
        1045         949         734        1421         378
        1240         958         561         569         327
         253         222         168         431         101
          29        -507        -279         462         -80
```

`FF` not possible, matrix dimensions do not agree.

`HH` not possible, matrix dimensions do not agree.

```
>> E*F
ans =
   972
   746
   616
   266
  -199
```

`FE` not possible, matrix dimensions do not agree.

```
>> H*E
ans =
   275   273   258   233    94
   284   213  -335  -245    24
```

`EH` not possible, matrix dimensions do not agree.

`FH` not possible, matrix dimensions do not agree.

```
>> H*F
ans =
   147
   -31
```

---

## 1.8

```
% clear workspace
clear;clc;
% see the values in the VARIABLE editor (workspace)
%% part 1
A=[2 4 6; 3 5 1; 7 5 9];
[V1,e1]=eig(A);
%% part 2
A=[12 14 16 40; 32 15 11 1; 7 25 19 10];
B=[9 1 36 4; 19 0 -31 2];
C=[7; 5; 7; 2; 0];
F=[16; 3; 17; 12;-2];
H=[5 7 4 -2 -1; -9 3 12 -6 14];
D=[A;B];
E=[D C];
```

```
% this is what E should be in the workspace
% E = [12 14 16 40 7; 32 15 11 1 5; 7 25 19 10 7; 9 1 3 6 2; 19 0 -31 2 0]
[V2,e2]=eig(E);
```

---

## 1.9

(a)
```
clear;clc;
A=[12 16 4; 23 1 21; 9 10 1];
B=[2 7 14; 3 11 2; -9 10 12];
X = A^(-1);
Y = B^(-1);
>> A*B
ans =
    36   300   248
  -140   382   576
    39   183   158
```

(b)
```
>>B*A
ans =
   311   179   169
   307    79   245
   230   -14   186
```

(c) $AB \neq BA$. Matrix multiplication is not generally commutative.

(d)
```
>> X=inv(A)
X =
   -0.2025    0.0233    0.3217
    0.1609   -0.0233   -0.1550
    0.2141    0.0233   -0.3450
```

(e)
```
>> Y=inv(B)
Y =
    0.0678    0.0339   -0.0847
   -0.0327    0.0908    0.0230
    0.0781   -0.0502    0.0006
```

(f)
```
>> A*X
ans =
    1.0000    0.0000         0
         0    1.0000   -0.0000
   -0.0000    0.0000    1.0000
```

(g)
```
>> B*Y
ans =
    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
         0         0    1.0000
```

(h) Parts (g) and (h) have the same answers, the identity matrix. It is true because $AX = AA^{-1} = I$ and $BY = BB^{-1} = I$

## 1.10

(a)
```
clear;clc;
A=[12 16 4; 23 1 21; 9 10 1];
B=[2 7 14;3 11 2; -9 10 12];
C=[43 12; 13 12];
D=[1 2 3; 4 5 6];
```

(1)
```
>> A+B
ans =
    14    23    18
    26    12    23
     0    20    13
```

(2) `A+C` not possible, `A` and `C` not the same order.

(3) `A+D` not possible, `A` and `D` not the same order.

(4) `B+C` not possible, `B` and `C` not the same order.

(5) `B+D` not possible, `B` and `D` not the same order.

(6) `C+D` not possible, `C` and `D` not the same order.

(c) See above

(1)
```
>> A+B
ans =
    14    23    18
    26    12    23
     0    20    13
```

(2)
```
>> B+A
ans =
    14    23    18
    26    12    23
     0    20    13
```

(3) `A+B` and `B+A` are equal. Matrix addition is commutative.

---

## 1.11

(a)
```
clear;clc;
A=[12 16 4; 23 1 21; 9 10 1];
B=[2 7 14;3 11 2; -9 10 12];
>> D=A*B
D =
    36   300   248
  -140   382   576
    39   183   158
```

(b)
```
>> E=A'
ans =
    12    23     9
    16     1    10
     4    21     1
```

(c)
```
>> F=B'
F =
     2     3    -9
     7    11    10
    14     2    12
```

(d)
```
>> G=D'
G =
    36  -140    39
   300   382   183
   248   576   158
```

(e) (1)
```
>> E*F
ans =
   311   307   230
   179    79   -14
   169   245   186
```

(2)
```
>> F*E
ans =
    36  -140    39
   300   382   183
   248   576   158
```

(f) Part (2), `F*E` is the same as matrix D.

(g) Recall the property `(AB)'=B'A'`

---

## 1.12

(a)
```
>> a=[3 4;4 2];
>> b=[12;10];
>> x=inv(a)*b
x =
    1.6000
    1.8000
```

(b)
```
>> a=[3 4;4 0];
>> b=[12;10];
>> x=inv(a)*b
x =
    2.5000
    1.1250
```

(c)
```
>> a=[-4 1;4 3];
>> b=[14;10];
>> x=inv(a)*b
x =
```

```
    -2
     6
```

(d)
```
>> a=[13 12;-4 7;11 -13];
>> b=[-6;-73;157];
>> x=pinv(a)*b
x =
    6.0000
   -7.0000
```

Note the usage of the command `pinv(a)` in this problem. Since `a` is not square, we use its pseudoinverse.

(e)
```
>> a=[2 3 -1;4 -2 1;1 5 -2];
>> b=[8;5;9];
>> x=inv(a)*b
x =
     2
     1
    -1
```

(f)
```
>> a=[4 -8 3;-1 2 -5;3 -6 1];
>> b=[16;-21;7];
>> x=pinv(a)*b
x =
    0.2000
   -0.4000
    4.0000
```

(g)
```
>> a=[2 3 1 -11;5 -2 5 -4;1 -1 3 -3;3 4 -7 2];
>> b=[1;5;3;-7];
>> x=pinv(a)*b
x =
   -0.0000
   -0.4286
    0.7143
   -0.1429
```

---

## 1.13

```
% Generating 25 elements in A using for loop logic
for i=1:1:25
    A(i)=i;
end
```

---

## 1.14

```
%Test if two numbers are greater than zero
function fun(n,m)
```

```
if ((n>0) && (m>0))
    display('Both');
elseif ((n<=0) && (m<=0))
    display('None');
else
    display('Other');
end
```

---

## 1.15

(a)
```
a=ones(1,50);
```

(b)
```
for i=1:1:50
    if rem(i,2)==0 a(i)=2;
    end
end
```

(c)
```
for i=1:1:50
    if rem(i,3)==0 a(i)=3;
    end
end
The final output:
a =
     1     2     3     2     1     3     1     2     3     2
     1     3     1     2     3     2     1     3     1     2
     3     2     1     3     1     2     3     2     1     3
     1     2     3     2     1     3     1     2     3     2
     1     3     1     2     3     2     1     3     1     2
```

---

## 1.16

See Fig. 1.5.

```
% clear workspace
clear;clc;

% contour plots
[x,y]=meshgrid(1:0.5:2,3:0.5:5);
z = (3*x.^2+4*y.^2);

% subplots of contour plots
subplot(2,3,1)
[c,h] = contour(x,y,z,4);
clabel(c,h)
xlabel('x')
ylabel('Function value')
title('Two Contours')
```

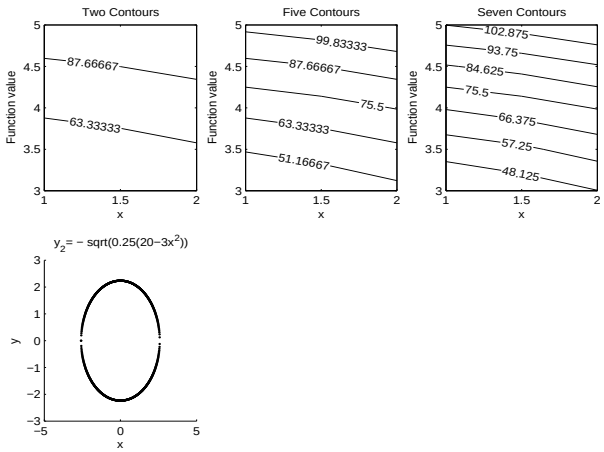


Figure 1.5: Contour Plots and Scatter plot for Problem 1.16

```
subplot(2,3,2)
[c2,h2] = contour(x,y,z,7);
clabel(c2,h2)
xlabel('x')
ylabel('Function value')
title('Five Contours')

subplot(2,3,3)
[c3,h3] = contour(x,y,z,9);
clabel(c3,h3)
xlabel('x')
ylabel('Function value')
title('Seven Contours')

% scatter plot
x1 = -sqrt(20/3):0.01:sqrt(20/3);
y1 = sqrt(1/4*(20-3*x1.^2));
y2 = -sqrt(1/4*(20-3*x1.^2));
% subplot of scatter plots
subplot(2,3,4)
scatter(x1,y1,'.')
xlabel('x')
ylabel('y')
title('y_1=sqrt(0.25(20-3x^2))')
hold on
scatter(x1,y2,'.')
xlabel('x')
ylabel('y')
title('y_2= -sqrt(0.25(20-3x^2))')
```

## 1.17

See Fig. 1.6.

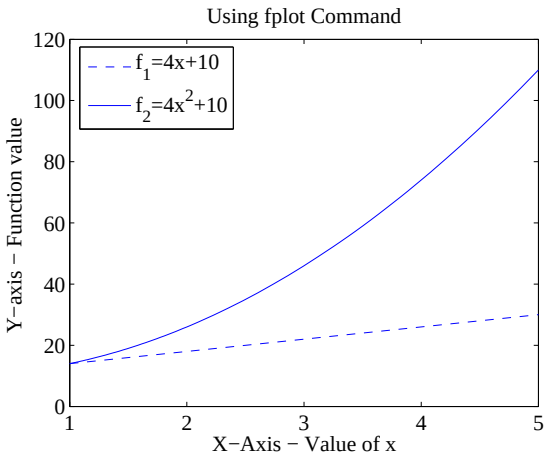


Figure 1.6: Plot of the 2 functions for Problem 1.17

```
% clear workspace
clear;clc;close all
% plot within specified limits
fplot('4*x+10',[1 5],':')
hold on
fplot('4*x^2+10',[1 5])
set(gca,'FontName','Times','FontSize',15)
xlabel('X-Axis - Value of x')
ylabel('Y-axis - Function value ')
legend('f_1=4x+10','f_2=4x^2+10',2)
title('Using fplot Command')
```

---

## 1.18

See Figs. 1.7 and 1.8.

```
% clear workspace
clear;clc;close all
fplot('4*x+10',[0 3.5],'-')
hold on
fplot('4*x^2+10',[0 3.5],':')
hold on
fplot('sin(x)',[0 3.5],'-.')
legend('y_1=4x+10','y_2=4x^2+10','y_3=sinx',2)
xlabel('X-Axis -Value of x')
ylabel('Y-Axis - Function value')
title('Problem 1.19 -Part 1')
figure(2)
% create subplots
```

```
subplot(3,1,1);
fplot('4*x+10',[0 3.5],'-');
xlabel('Value of x')
ylabel('Function value')
title('Plot 1 - y_1=4x+10')
legend('y_1=4x+10',2)

subplot(3,1,2);
fplot('4*x^2+10',[0 3.5],':');
xlabel('Value of x')
ylabel('Function value')
title('Plot 2 - y_2=4x^2+10')
legend('y_2=4x^2+10',2)

subplot(3,1,3);
fplot('sin(x)',[0 3.5],'-.')
xlabel('Value of x')
ylabel('Function value')
title('Plot 3 - y_3=sinx')
legend('y_3=sinx',2)
```

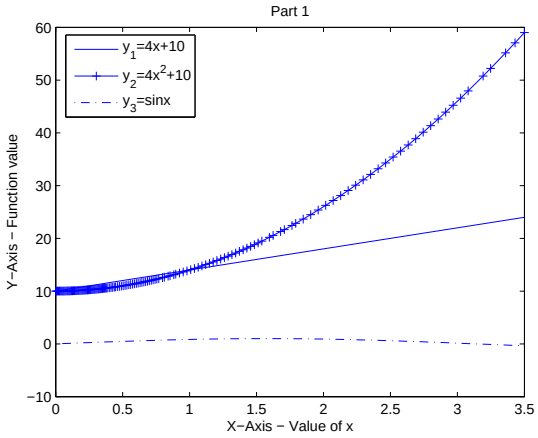


Figure 1.7: Plot of the 3 functions for Problem 1.18

## 1.19

See Figs. 1.9 and 1.10.

```
% clear workspace
clear;clc;close all
% define peak function
Z=peaks;
% mesh plot
figure(1)
```

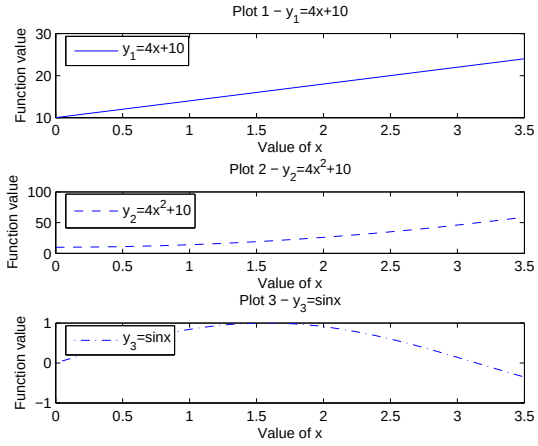


Figure 1.8: Subplots of the 3 functions for Problem 1.18

```
mesh(Z)
xlabel('X - axis')
ylabel('Y - axis')
zlabel('Z - axis')
title('3D plot of peaks using mesh')
legend('legend')
text(0,60,10,'First name')
text(0,0,0,'Last name')
% surface plot
figure(2)
surf(Z)
xlabel('X - axis')
ylabel('Y - axis')
zlabel('Z -axis')
title('3D plot of peaks using surf')
legend('legend')
text(0,60,10,'First name')
text(0,0,0,'')
```

---

## 1.20

(a) getSquared.m

```
% clear workspace
clear;clc;
%% for part c
% define input
xinput = 5;
% Call the function
xoutput = getSquared(xinput);
% Display result on screen
```

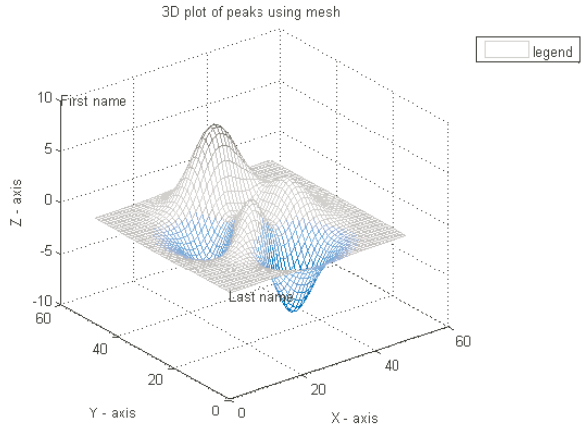


Figure 1.9: 3-D plot using mesh for Problem 1.19

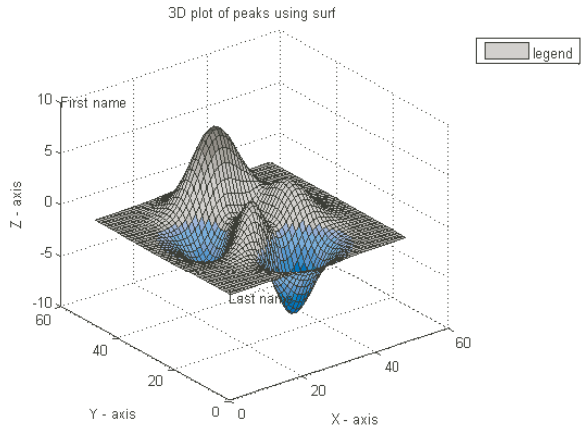


Figure 1.10: 3-D plot using surf for Problem 1.19

```
fprintf('Input = %d, Output = %d\n',xinput,xoutput);
```

(b) getSquared.m

```
% This function returns the squared value of input
function y = getSquared(x)
y = x^2;
```

(c)
```
% define input
xinput = 5;     % non-negative input
% xinput = -5;  % negative input
% Call the function
xoutput = getSquaredSpecial(xinput);
% Display result on screen
fprintf('Input = %d, Output = %d\n',xinput,xoutput);
```

(d) getsquaredspecial.m

```
% This function checks whether the input >= 0
% if yes, return the squared value of x
% otherwise, return -1
function y = getSquaredSpecial(x)
if x >=0
    y = x^2;
else
    f = -1;
end
```

---

## 1.21

(a,c)
```
% This function calculates the sum
% of the first x integers from 1
function y = sumHundred(x)
% initialize sum
sum = 0;
% check if x is between 1 and 100
if x < 1 || x > 100
    disp('Input not between 1 and 100');
    y = -1;
else
% Calculate sum of the first x integers
    for i = 1:1:x
        sum = sum + i;
    end
y = sum;
end
```

(d)
```
% This function calculates the sum of all even integers from 1 to x
function y = sumEvenHundred(x)
% initialize sum
sum = 0;
if x < 1 || x > 100
    disp('Input not between 1 and 100')
    y = -1;
else
    % Calculate sum of the first inpt integers
    for i = 1:1:x
        % Check if i is even
        if rem(i,2) == 0
            sum = sum + i;
        else
            continue;
        end
    end
y = sum;
end
```

## 1.22

(a)
```
% Function to calculate the sum of
% two input numbers
function opt = hiddenSum(a,b) opt = a + b;
```

(b)
```
clear;clc;
%% part b
% script to test the function "hiddenSum"
% specify inputs
x = 23; y = 54;
z = hiddenSum(x,y);
display(z);
```

(c) (1)
```
function hiddenSum_c(a,b)
    global fc
    fc = a+b;
```

(2)
```
% script to test the function "hiddenSum_c"
    % specify inputs
    x = 23; y = 54;
    global fc
    hiddenSum_c(x,y);
    z = fc;
    display(z);
```

(d) (1)
```
function hiddenSum_d
    global fd a b;
    fd = a + b;
```

(2)
```
% script to test the function "hiddenSum_d"
    % specify inputs
    x = 23; y = 54;
    global fd a b
    a = x;
    b = y;
    hiddenSum_d();
    z = fd;
    display(z);
```