MATLAB: A Practical Introduction to Programming and Problem Solving

Fourth Edition

PRACTICE PROBLEM SOLUTIONS

Stormy Attaway

College of Engineering

Boston University

**Chapter 1**

*Practice 1.1*

Think about what the results would be for the following expressions, and then type them in to verify your answers:

>> 1\2

ans =

2

>> - 5 ^ 2

ans =

-25

>> (-5) ^ 2

ans =

25

>> 10-6/2

ans =

7

>> 5\*4/2\*3

ans =

30

*Practice 1.2*

Generate a random

* real number in the range (0,1)

rand

* real number in the range (0, 100)

rand\*100

* real number in the range (20, 35)

rand\*(35-20)+20

* integer in the inclusive range from 1 to 100

randi(100)

* integer in the inclusive range from 20 to 35

randi([20, 35])

*Practice 1.3*

Think about what would be produced by the following expressions, and then type them in to verify your answers.

>> 3 == 5 + 2

ans =

0

>> 'b' < 'a' + 1

ans =

0

>> 10 > 5 + 2

ans =

1

>> (10 > 5) + 2

ans =

3

>> 'c' == 'd' - 1 && 2 < 4

ans =

1

>> 'c' == 'd' - 1 || 2 > 4

ans =

1

>>xor('c' == 'd' - 1, 2 > 4)

ans =

1

>>xor('c' == 'd' - 1, 2 < 4)

ans =

0

>>10 > 5 > 2

ans =

0

*Practice 1.4*

* Calculate the range of integers that can be stored in the types **int16**and **uint16**. Use **intmin** and **intmax** to verify your results.

>> 2^16

ans =

65536

>> 2^15

ans =

32768

>>intmin('int16')

ans =

-32768

>>intmax('int16')

ans =

32767

>>intmin('uint16')

ans =

0

>>intmax('uint16')

ans =

65535

* Enter an assignment statement and view the type of the variable in the Workspace Window. Then, change its type and view it again. View it also using **whos**.

>>clear

>>mynumber = 3\*11;

>>whos

Name Size Bytes Class Attributes

mynumber 1x1 8 double

>>mynumber = int32(mynumber)

mynumber =

33

>>whos

Name Size Bytes Class Attributes

mynumber 1x1 4 int32

*Practice 1.5*

* Find the numerical equivalent of the character 'x'.
* Find the character equivalent of 107.

>>double('x')

ans =

120

>>char(107)

ans =

k

*Practice 1.6*

Use the **help** function to find out what the rounding functions **fix**, **floor**, **ceil**, and **round** do. Experiment with them by passing different values to the functions, including some negative, some positive, some with fractions less than 0.5 and some greater. *It is very important when testing functions that you thoroughly test by trying different kinds of arguments!*

**Chapter 2**

*Practice 2.1*

Think about what would be produced by the following sequence of statements and expressions, and then type them in to verify your answers:

pvec = 3:2:10

pvec(2) = 15

pvec(7) = 33

pvec([2:4 7])

linspace(5,11,3)

logspace(2,4,3)

>>pvec = 3:2:10

pvec =

3 5 7 9

>>pvec(2) = 15

pvec =

3 15 7 9

>>pvec(7) = 33

pvec =

3 15 7 9 0 0 33

>>pvec([2:4 7])

ans =

15 7 9 33

>>linspace(5,11,3)

ans =

5 8 11

>>logspace(2, 4, 3)

ans =

100 1000 10000

*Practice 2.2*

Think about what would be produced by the following sequence of statements and expressions, and then type them in to verify your answers.

mat = [1:3; 44 9 2; 5:-1:3]

mat(3,2)

mat(2,:)

size(mat)

mat(:,4) = [8;11;33]

numel(mat)

v = mat(3,:)

v(v(2))

v(1) = []

reshape(mat,2,6)

>>mat = [1:3; 44 9 2; 5:-1:3]

mat =

1 2 3

44 9 2

5 4 3

>>mat(3,2)

ans =

4

>>mat(2,:)

ans =

44 9 2

>>size(mat)

ans =

3 3

>>mat(:,4) = [8;11;33]

mat =

1 2 3 8

44 9 2 11

5 4 3 33

>>numel(mat)

ans =

12

>>v = mat(3,:)

v =

5 4 3 33

>>v(v(2))

ans =

33

>>v(1) = []

v =

4 3 33

>>reshape(mat,2,6)

ans =

1 5 9 3 3 11

44 2 4 2 8 33

###### Practice 2.3

Create a vector variable and subtract 3 from every element in it.

Create a matrix variable and divide every element by 3.

Create a matrix variable and square every element.

>>vec = [4 11 32 -5 0 9]

vec =

4 11 32 -5 0 9

>>vec - 3

ans =

1 8 29 -8 -3 6

>>

>>mat = randi(30,2,4)

mat =

24 4 19 8

27 27 3 16

>>mat/3

ans =

8.0000 1.3333 6.3333 2.6667

9.0000 9.0000 1.0000 5.3333

>>mat .^ 2

ans =

576 16 361 64

729 729 9 256

*Practice 2.4*

Modify the result seen in the previous Quick Question. Instead of deleting the “bad” elements, retain only the “good” ones. (Hint: Do it two ways, using **find** and using a logical vector with the expression *vec >= 0*).

>>vec = [11 -5 33 2 8 -4 25]

vec =

11 -5 33 2 8 -4 25

>>pos = find(vec >= 0)

pos =

1 3 4 5 7

>>res = vec(pos)

res =

11 33 2 8 25

>>vec(vec>=0)

ans =

11 33 2 8 25

*Practice 2.5*

When two matrices have the same dimensions and are square, both array and matrix multiplication can be performed on them. For the following two matrices, perform A.\*B, A\*B, and B\*A by hand and then verify the results in MATLAB.

A B



>>A .\* B

ans =

1 8

-3 0

>> A \* B

ans =

-3 2

0 6

>> B \* A

ans =

7 10

-1 -4

**Chapter 3**

*Practice 3.1*