Homework #2

Table of Contents

Jason Chiarulli

ENGR 108

Matlab for Engineers, Third Edition

Problem 4.1

```
a = [15 3 22; 3 8 5; 14 3 82];
b = [1; 5; 6];
c = [12 18 5 2];

% a)
d = a(:, 3) % Creates a matrix 'd' from matrix 'a' by
            % by using the colon operator and the number 3
            % to extract all the rows in column 3 of matrix
            % 'a'

% b)
e = [b, d] % Creates a two-dimensional matrix 'e' by combining
           % matrices 'b' and 'd'

% c)
f = [b; d] % Creates a one-dimensional matrix 'f' by combining
           % matrices 'b' and 'd'

% d)
g = [a; c(:, 1:3)] % Creates a matrix 'g' from matrix 'a'
                   % by using the colon operator to extract
                   % all the rows in columns 1 to 3 of matrix 'c'

% e)
h = [a(1,3) c(1,2) b(2,1)]
% Creates a matrix 'h' by extracting the following elements:
% first row third column of matrix 'a', first row second column
% of matrix 'c', and second row first column of matrix 'b'
```

```
d =
```

```
    22
     5
    82


e =

     1    22
     5     5
     6    82


f =

     1
     5
     6
    22
     5
    82


g =

    15     3    22
     3     8     5
    14     3    82
    12    18     5


h =

    22    18     5
```

Problem 4.4

```
% a)
% Import and save the data file Sensor.dat to the current directory
load Sensor.dat     % Load the sensor data file

amount_of_time_data = size(Sensor,1)
% Determines the row size of the imported matrix 'Sensor' which is
% equal to the amount of time data

amount_of_sensor_data = size(Sensor,1)*(size(Sensor,2) - 1)
% Determines the column size of the imported matrix 'Sensor' which is
% subtracted by 1 since the first column is composed of time data
% The new column size value is then multiplied by the row size of the
% imported matrix which results in the amount of sensor data

% b & c)
```

```
column = 2; % Creates and initializes an integer column
x = 1; % Creates and initializes an integer x
while column <= size(Sensor,2)
% executes loop while the column is less than or equal to the column
% size of the matrix 'Sensor'

   [max_value, row_max_occurred] = max(Sensor(:,column));
   % Finds the largest value and its location in each column
   % containing sensor data in the matrix 'Sensor'

   time_max_occurred = Sensor(row_max_occurred);
   % Determines the time at which the max occurred by taking the
   % location and passing it as an argument to the 'Sensor' matrix

   Max(x,:) = max_value;
   % Stores all of the max sensor data in a matrix

   Max_Time(x,:) = time_max_occurred;
   % Stores all of the max time data in a matrix

   [min_value, row_min_occurred] = min(Sensor(:,column));
   % Finds the smallest value and its location in each column
   % containing sensor data in the matrix 'Sensor'

   time_min_occurred = Sensor(row_min_occurred);
   % Determines the time at which the min occurred by taking the
   % location and passing it as an argument to the 'Sensor' matrix

   Min(x,:) = min_value;
   % Stores all of the min time data in a matrix

   Min_Time(x,:) = time_min_occurred;
   % Stores all of the min time data in a matrix

   Mean(x,:) = mean(Sensor(:,column));
   % Calculates the mean of each row containing sensor data and stores
   % the mean data in matrix

   Std(x,:) = std(Sensor(:,column));
   % Calculates the standard deviation of each row containing sensor
   % data and stores the standard deviation data in matrix

   column = column + 1;
   % Increments column by one each time the loop executes

   x = x + 1;
   % Increments x by one each time the loop executes

end % ends the loop

Mean_of_all_Sensor_Data = mean(Mean);
% Calculates the mean of all of the sensor data

Sensor_Data = Sensor(:,2:6);
```

```
% Creates a matrix by extracting the data from all the rows in columns
% 2 to 6 of matrix 'Sensor'

Std_of_all_Sensor_Data = std(Sensor_Data(:));
% Calculates the standard deviation of all of the sensor data

Sensors = {'Sensor 1'; 'Sensor 2'; 'Sensor 3'; 'Sensor 4';
           'Sensor 5'};
% Creates labels for each row in the Max_Min_Time_Sensor_Data_Table
% and the Mean_Std_Sensor_Data_Table

Max_Min_Time_Sensor_Data_Table = table(Max,Max_Time,Min,Min_Time,...
    'RowNames',Sensors)
% Populates a table with the min, max, and time ocurred sensor column
% data and with rownames

Mean_Std_Sensor_Data_Table = table(Mean,Std, 'Rownames',Sensors)
% Populates a table with the mean and standard deviation sensor column
% data and with rownames

Analyzed_Sensors_Data_Table = table(Mean_of_all_Sensor_Data,...
    Std_of_all_Sensor_Data)
% Populates a table with the anaylzed data for all of the sensor data


amount_of_time_data =

    20


amount_of_sensor_data =

   100


Max_Min_Time_Sensor_Data_Table =

  5×4 table

                Max      Max_Time     Min      Min_Time
               ______    ________    ______    ________

    Sensor 1   77.395      7.5       63.403        3
    Sensor 2   77.676      8.5       63.263      9.5
    Sensor 3   77.965      4.5       2.7644      2.5
    Sensor 4   80.561      1.5       63.094        4
    Sensor 5   76.183      4.5       58.374      5.5


Mean_Std_Sensor_Data_Table =

  5×2 table

                Mean      Std
```

```
                 ______     ______

    Sensor 1    69.726    4.5471
    Sensor 2    69.101    3.9223
    Sensor 3    65.374    15.357
    Sensor 4     71.23     5.719
    Sensor 5    68.265    5.2412


Analyzed_Sensors_Data_Table =

  1×2 table

    Mean_of_all_Sensor_Data    Std_of_all_Sensor_Data
    _______________________    ______________________

    68.739                     8.2182
```

Problem 4.8

```
T = 100:100:1000; % Creates a temperature vector 'T' that starts at
                  % 100K and ends at 1000K with spacing equal to 100

P = 100:100:1000; % Creates a pressure vector 'P' that starts at
                  % 1000kPa and ends at 1000kPa with spacing equal
                  % to 100

R = 0.2870; % Universal gas constant measured in kJ/(kgK)

[new_P, new_T] = meshgrid(P,T); % Maps the vectors 'T' and 'P' into a
                                % two-dimensional array

v = R.*new_T./new_P  % Calculates the volume 'v' (m^3/kg) from the
                     % ideal gas law by using array multiplication
                     % and array division


v =

  Columns 1 through 7

    0.2870    0.1435    0.0957    0.0717    0.0574    0.0478    0.0410
    0.5740    0.2870    0.1913    0.1435    0.1148    0.0957    0.0820
    0.8610    0.4305    0.2870    0.2152    0.1722    0.1435    0.1230
    1.1480    0.5740    0.3827    0.2870    0.2296    0.1913    0.1640
    1.4350    0.7175    0.4783    0.3588    0.2870    0.2392    0.2050
    1.7220    0.8610    0.5740    0.4305    0.3444    0.2870    0.2460
    2.0090    1.0045    0.6697    0.5022    0.4018    0.3348    0.2870
    2.2960    1.1480    0.7653    0.5740    0.4592    0.3827    0.3280
    2.5830    1.2915    0.8610    0.6457    0.5166    0.4305    0.3690
    2.8700    1.4350    0.9567    0.7175    0.5740    0.4783    0.4100
```

```
  Columns 8 through 10

    0.0359    0.0319    0.0287
    0.0717    0.0638    0.0574
    0.1076    0.0957    0.0861
    0.1435    0.1276    0.1148
    0.1794    0.1594    0.1435
    0.2152    0.1913    0.1722
    0.2511    0.2232    0.2009
    0.2870    0.2551    0.2296
    0.3229    0.2870    0.2583
    0.3588    0.3189    0.2870
```

Problem 4.10

```
Magic_Matrix = magic(6) % Creates a 6x6 'magic matrix'

% a)
sum_of_the_rows = sum(Magic_Matrix') % Calculates the sum of each row
                                     % in the 'magic' matrix

%b)
sum_of_the_columns = sum(Magic_Matrix) % Calculates the sum of each
                                       % column in the 'magic' matrix

% c)
sum_of_the_diagonal = sum(diag(Magic_Matrix))
% Calculates the sum of the main diagonal of the 'magic' matrix

sum_of_the_diagonal_from_left_to_right =
 sum(diag(fliplr(Magic_Matrix)))
% Calculates the sum of the diagonal from the lower left to the upper
% right of the 'magic' matrix


Magic_Matrix =

    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11


sum_of_the_rows =

   111   111   111   111   111   111


sum_of_the_columns =
```

```
   111   111   111   111   111   111


sum_of_the_diagonal =

   111


sum_of_the_diagonal_from_left_to_right =

   111
```

Problem 10.7

```
% The products of A*B and B*A are not equal in either (a) or (b) since
% matrix multiplication is not commutative

% a)
A = [12 4; 3 -5];
B = [2 12; 0 0];
C = A*B
D = B*A

% b)
A = [1 3 5; 2 4 6];
B = [-2 4; 3 8; 12 -2];
C = A*B
D = B*A


C =

    24   144
     6    36


D =

    60   -52
     0     0


C =

    67    18
    80    28


D =

     6    10    14
    19    41    63
```

```
     8    28    48
```

Problem 10.18

```
% The systems of equations in (a), (b), and (c) were all solved using
% both matrix left division e.g. X = A\B and the inverse matrix method
% e.g. X = inv(A)*B

% a)
A = [-2 1; 1 1];
B = [3; 10];
X1 = inv(A)*B
X2 = A\B

% b)
A = [5 3 -1; 3 2 1; 4 -1 3];
B = [10; 4; 12];
X1 = inv(A)*B
X2 = A\B

% c)
A = [3 1 1 1; 1 -3 7 1; 2 2 -3 4; 1 1 1 1];
B = [24; 12; 17; 0];
X1 = inv(A)*B
X2 = A\B


X1 =

    2.3333
    7.6667


X2 =

    2.3333
    7.6667


X1 =

    3.1613
   -2.2581
   -0.9677


X2 =

    3.1613
   -2.2581
   -0.9677
```

```
X1 =

   12.0000
   -8.2500
   -3.5000
   -0.2500


X2 =

   12.0000
   -8.2500
   -3.5000
   -0.2500
```

Probelm 10.19

```
% The following system of equations was solved with both matrix left
% division e.g. X = A\B and the inverse matrix method e.g.
% X = inv(A)*B.
% The tic and toc functions were used to time the execution of each
% technique. The tic and toc functions confirmed that the matrix left
% division method was faster than the inverse matrix method.

A = [3 4 2 -1 1 7 1; 2 -2 3 -4 5 2 8;
     1 2 3 1 2 4 6; 5 10 4 3 9 -2 1;
     3 2 -2 -4 -5 -6 7; -2 9 1 3 -3 5 1;
     1 -2 -8 4 2 4 5];
B = [42; 32; 12; -5; 10; 18; 17];
tic
X1 = inv(A)*B
toc
tic
X2 = A\B
toc


X1 =

   -0.1890
    2.5459
   -3.2806
   -6.7578
    1.3212
    4.3194
    0.6294

Elapsed time is 0.000214 seconds.

X2 =
```

```
   -0.1890
    2.5459
   -3.2806
   -6.7578
    1.3212
    4.3194
    0.6294

Elapsed time is 0.000071 seconds.
```

Published with MATLAB® R2017a