MATLAB PORTFOLIO

Bella Barbera
December 14th 2017
Section A
Harrison Davis, Lianna Klinger, Brooke Bendarke, Callan Kennedy

Homework 7_8

Overview: To analyze a dataset of strings embedded into an RGB image file, where each character represents an intensity measurement of that pixel's blue signal. In part one, the file black_white is read into the script and its size and midpoints are calculated and stored as variables so that the image data can be stored in four different quadrants as variables. In part two the four quadrant variables and their length are defined. The for loop uses a subplot to create a 2by2 plot of each quadrant, where imshow displays each quadrant. In part three qstr is sorts the cell array into alphabetical order, and uses a FOR loop to create a table that calculates the mean for each quadrant, the maximum possible mean is also found. In part four the file secrets is read into the script, and a variable containing the blue layer image data is defined. ***** In part five creates a cell array that is split by strtok, and put into a while loop that continues to split up the vector into tokens and appends them to the cell array votes. In part six four variable are created that contain information about the strings (total, unique). A FOR loop is used compare two strings two determine if they contain the same city, a second for loop is used to create a table of the tallies. In part seven a new table is created using a FOR loop that is sorted numerically.

Skills Used:

- Using while and for loops
- Cell arrays, string concatenation, character data stored numerically
- End keyword used as an index
- Use of eval, char, strtok, max, size, sort, unique, imread, imshow, subplot, floor, sprintf or num2str

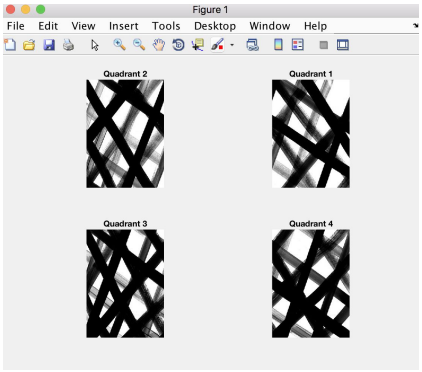
```
% Bella Barbera
%Secrets
%November 5 2017
%% Housekeeping
clc
clearvars
close all

%% Part One
blackwhite = imread('black_white.bmp'); %reads image data into
variable: blackwhite

[x,y] = size(blackwhite); %size determines both dimensions of matrix
blackwhite and stores them in variables: x and y

p1 = floor(x/2); % p1 is a variable that stores the midpoint of the x
dimensions. Floor rounds to minus infinity.
p2 = floor(y/2); % p2 is a variable that stores the midpoint of the y
dimension. Floor rounds to minus infinity.

Q1 = blackwhite(1:p1,p2+1:end); %Q1 is a variable that holds one
quarter of the image data
Q2 = blackwhite(1:p1,1:p2); %Q2 is a variable that holds one quarter
of the image data
```

```
Q3 = blackwhite(p1+1:end,1:p2); %Q3 is a variable that holds one
quarter of the image data
Q4 = blackwhite(p1+1:end,p2+1:end); %Q4 is a variable that holds one
quarter of the image data

%% Part Two
qstr = {'Q2','Q1','Q3','Q4'}; % qstr is a variable that is a cell
array of the strings of the quadrants 1-4

l = length(qstr); %l is the variable for the length of the cell array

for k = 1:l % k is the loop iterator variable. 1:L means 1 through
the L which is through each quadrant (a total of 4)
    subplot(2,2,k); %subplot(m,n,p) breaks the figure window into and
m-by-n (in this case 2 by 2) matrix of small axes, selects the pth
(in this case k) axes for the current plot
    qvar = ['imshow(' qstr{k} ')']; % qvar is a variable that is a
string. Imshow displays qstr{k} which indexes into each string
showing each quadrant.
    eval(qvar) % executes qvar in text
    num2str(k); % ***converts numbers to characters
    title(['Quadrant ' num2str(qstr{k}(2))]) %Creates title above
each quadrant of what quadrant number it is by extracting the second
character of each Quadrant in qstr (ex: 'Q1' =1 'Q2' =2)
end

%% Part Three
b = sort(qstr); %sort updates the cell array into alphabetical
listing
e=zeros(1,l); % ***using the length variable, a vector of zeros is
created the same length of the string, which will store the mean of
its corresponding quadrant

fprintf('%45s\n\n','Mean Signal Intensity by Quadrant') % Title for
table and headers
fprintf('%20s%20s\n','Quad','Mean',...
                     '------','-----')
for k = 1:l %creating table body (1:L)
    mstr = ['mean(' b{k} '(:))']; %defining a string (mstr) that
represents the quadrants mean calculation that uses k to index into
your cell array (b)
    e(k) = eval(mstr); %executes the matlab expression (mstr) in text
    fprintf('%19s%20.2f\n',b{k},e(k)) %using k to index into cell
array and means vector to produce table
end

s = max(blackwhite(:)); %finds maximum possible numerical value (255)
the out of all
```

```
[V,W] = max(e); %finds maximum quadrant and quadrant mean

fprintf('Quadrant %d has the largest mean signal intensity: %0.0f out
of %d.\n',W,V,s) %W is max quadrant, V is max quadrant mean, s (255)

%% Part Four
secrts = imread('secret.bmp'); %reading data of secrets into variable
secrets
blue = secrts(:,:,3); %variable containing the blue layer image data
for quadrant 1 (RGB = :,:3) blue
Q1B = blue(1:p1,p2+1:end); %variable containing blue layer image data
of Q1

str=[]; %setting empty vector as string
ind=1; %setting index
ch = 'A'; %

while ch ~='E' %ends when the 'E' is reached which is at the end of
the data
   ch = char(Q1B(ind)); %Since Q1B is a number it needs to be
converted to a number so it can be tacked onto the vector of
characters
   ind = ind + 1;
   str = [str ch];
End

%% Part Five
votes = cell(1); %creating cell array
[votes{1},rest] = strtok(str); %splits vector of characters into a
cell array of meaningful string values; [TOKEN,REMAIN] = strtok(STR)

while ~isempty(rest) %while loop that continues to parse the 'rest'
into tokens (strings) ~(while rest is not empty keep going)
    [votes{1+end},rest] = strtok(rest); %appending each string to the
cell array votes
end

votes = votes(1:end-1); %updating votes to include only elements 1
through end -1

%% Part Six
citycell = unique(votes); %**creates cell array with no repetitions
in sorted order
totalstrings = length(votes); %variable for total number of strings
uniquestrings = length(citycell); %variable for the number of unique
strings
```

```
g = zeros(1,uniquestrings); %g is a vector of zeros whose length
matches the number of unique strings
for c = 1:totalstrings % loop that is bound by the total number of
strings
    current = votes{c}; %index into the votes cell array
    temp = strcmp(current,citycell); %strcmp compares (1,2) and
returns a logical value 1 if they are the same and 0 if they are not
    g = g + temp; %updating tally vector by adding its current value
to temp
end

fprintf('%-40s\n\n','Unique Values and Tallies') %title for table
fprintf('%-20s%20s\n','Unique','Tallies',... %headers for table
                      '------','-----')
for c = 1:uniquestrings %for loop to create table of the unique
values and their tallies
    fprintf('%-20s%20d\n',citycell{c},g(c)) %index into the cell
array of unique values and the corresponding vector of tallies
end

%% Part Seven
[M,N] = sort(g); %creates the variables M -a numerically sorted tally
vector, and N the order in which these sorted values appear in the
original vector

sorted = citycell(N); %create new cell array of unique strings
indexing into original cell array using N

fprintf('%-40s\n\n','Sorted Table') %table headers
fprintf('%-20s%20s\n','Cities','Number of Votes',... %table headers
                      '------','-----')
for c = 1:uniquestrings %creates table of values in new sorted order
    fprintf('%-20s%20d\n',sorted{c},M(c)) %puts values in increasing
order of number of votes
End
```

IMAGES AND TABLES (HW7_8)



Black_white.bmp
Part Two: Subplot of Image Data in Quadrant

Command Window

```
            Mean Signal Intensity by Quadrant

    Quad                Mean
    ------              -----
      Q1               105.88
      Q2                93.90
      Q3                68.96
      Q4                84.74
Quadrant 1 has the largest mean signal intensity: 106 out of 255.
```

Part Three: Mean Signal Intensity by Quadrant Table

Unique Values and Tallies

Unique	Tallies
Baltimore	1115
Boston	1120
Chicago	1103
Cleveland	1106
Houston	1124
LA	1060
NY	1101
Seattle	1168
Toronto	1103
Sorted Table	

Cities	Number of Votes
LA	1060
NY	1101
Chicago	1103
Toronto	1103
Cleveland	1106
Baltimore	1115
Boston	1120
Houston	1124
Seattle	1168

Workspace

Name	Value
ans	*4x4 double*
b	*1x4 cell*
blackwhite	*766x550 uint8*
blue	*766x550 uint8*
c	9
ch	'E'
citycell	*1x9 cell*
current	'Boston'
e	[105.8760,93.8961,68.9595,84.7395]
g	[1115,1120,1103,1106,1124,1060,1101,11...
ind	72522
k	4
l	4
M	[1060,1101,1103,1103,1106,1115,1120,11...
mstr	'mean(Q4(:))'
N	[6,7,3,9,4,1,2,5,8]
p1	383
p2	275
Q1	*383x275 uint8*
Q1B	*383x275 uint8*
Q2	*383x275 uint8*
Q3	*383x275 uint8*
Q4	*383x275 uint8*
qstr	*1x4 cell*
qvar	'imshow(Q4)'
rest	''
s	255
secrts	*766x550x3 uint8*
sorted	*1x9 cell*
str	'Chicago Houston Boston Toronto Baltimore...
temp	*1x9 logical*
totalstrings	10000
uniquestrings	9
V	105.8760
votes	*1x10000 cell*
W	1
x	766
y	550

Part Six: Table of Unique Values and Tallies
Part Seven: Table of Sorted Unique Values and Tallies

Homework 9

Overview: Imports data from two data files, pumps and deaths, and use them to recreate the point map and tally the fatalities according to the nearest pump. In part one deaths and pumps are loaded into the script and a while loop is used to create two variable that contain two columns of doubles. In part two a figure is created that plots the deaths and pumps. In part three three variables are created for the number of pumps, deaths, and a vector of zeros, a for loop is then used to calculated distances between the pumps and fatalities. In part four the minimum distance between from each fatality to a pump is calculated and a FOR loop is used to step through each fatality, the tallies are then recorded and sorted in ascending order. In part five a figure is created that plots all the pumps (as a number 1-13) and fatalities (as a number of the pump they are closest to). In part six all the pumps and deaths are re-plotted, deaths as circles whos size are based on the number of fatalities closest to the pump. In part seven a file is created that stores two tables each that contain the pumps and the number of cases closest to them, one of them in order of pump number and the other in descending number of cases.

Skills Used:

- WHILE and FOR loops
- Vectorizing plots and distance calculations
- End keyword used as an index
- [~] used to indicate “don’t care”

- Use of figure, clf, plot, text, zeros, cell2mat, find, strdouble, sort, min, axis, grid, xlabel, ylabel, title, legend, fopen, fclose, feof, fgetl, textscan, strtok, input, fprintf

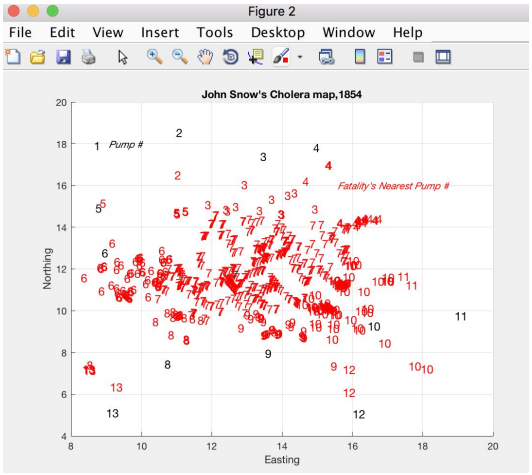
```
% Bella Barbera
% November 8th, 2017
% Johnsnow

clc
clearvars
close all

%% Part One

fid = fopen('pumps.txt') % Opening pumps.txt into a variable fid
pumps = textscan(fid,'%f,%f','headerlines',1); %Creating a variable
and opening pumps (fid) in the proper format
fclose(fid) %close file
pumps = cell2mat(pumps) %converts pumps to a two-column matrix of
doubles

fid = fopen('deaths.txt') %Opening deaths into variable fid
deaths = []; %create empty vector to initialize deaths

while ~feof(fid) %read into each data line until the end of the file
is reached
    line = fgetl(fid); %creates variable 'line' that returns the next
file line
     [x,y] = strtok(line,','); %strtok assigns x and y variables that
token the selected line
    x = str2double(x); %converts each variable from string to double
    y = str2double(y); %converts each variable from string to double
    deaths(end+1,:) = [x,y]; %append a 1x2 vector comprised of x and
y to the next row of the deaths matrix
    fprintf('%s\n',line) %new line
end
fclose(fid) %close file

%% Part Two

figure(1) %clear figure
clf

plot(deaths(:,1),deaths(:,2),'rx') %plotting x and y (all of column
1/2 of deaths data)
hold on %add to plot whats below
plot(pumps,'ko') %plot all pumps data
```

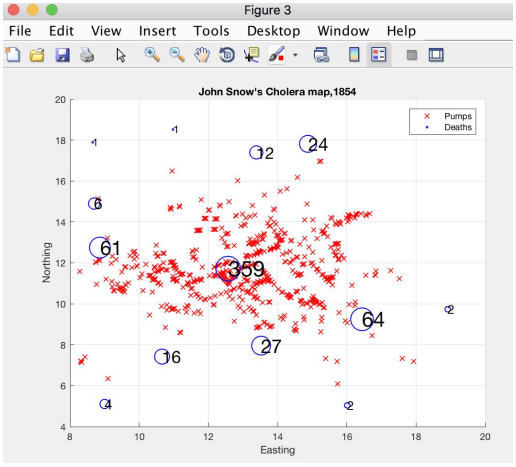
```
title('John Snow''s Cholera Map, 1854') %creating titles/setting axes
axis([0 24 0 20])
xlabel('Easting')
ylabel('Northing')
grid ON
legend(' Deaths',' Pumps')

%% Part Three
numberpumps = length(pumps); %variable for the number of pumps
numberdeaths = size(deaths,1); %variable for the number of deaths

z = zeros(numberdeaths,numberpumps); %creating a vector of zeros with
as many rows as deaths, and as many columns as pumps

for k = 1:numberpumps % FOR loop to compute distance from each pump
to each fatality (indexing thru one pump at a time)
    dx = pumps(k,1) - deaths(:,1); %one pump (x) to all fatalities
    dy = pumps(k,2) - deaths(:,2); %one pump (y) to all fatalities
    z(:,k) = sqrt((dx.^2)+(dy.^2)); %pythagorean formula for
calculating distance from each pump to all fatalities
end

%% Part Four

[~,closest_pump] = min(z,[],2); %identifying the smallest distance
between each pump and fatality (each fatality to pump)
pumptally = zeros(numberpumps,1); %create a vector of zeros for the
number of pump tallies that is the length of the number of pumps

for k = 1:numberdeaths %FOR loop stepping thru closest pumps thru #
of deaths
    set_pump = closest_pump(k); %step through closest_pump each of
whose entries represents which pump is closest to that row's fatality
    pumptally(set_pump) = pumptally(set_pump) + 1; %keeping tally of
which pump has the most fatalities closest to it
end

[sorted_tally,order] = sort(pumptally); % sort tallies in ascending
order

%% Part Five

figure(2),clf %clear figure and create second figure

hold on %add to plot
grid on %turn on grid lines
axis([8 20 4 20]) %set axes
```

```
% plotting pumps
for k = 1:numberpumps %For loop through all the pumps
    text(pumps(k,1),pumps(k,2),num2str(k),'color','k','fontsize',12)
%adding text to data points for each pump (points = numbers)
end

%plotting deaths
for k = 1:numberdeaths %for loop through all deaths
text(deaths(k,1),deaths(k,2),num2str(closest_pump(k)),'color','r','fo
ntsize',12) %also create text data points but also index into
closest_pump for the annotation value
end

%labels
title('John Snow''s Cholera map,1854')
xlabel('Easting')
ylabel('Northing')

text(15.5,16,'\it Fatality''s Nearest Pump #','color',
'r','linewidth',10)
text(9,18,'\it Pump #', 'color','k','linewidth',10),

%% Part Six

figure(3),clf %create third figure, clear previous
hold on %add to graph
grid on %grid on
axis([8 20 4 20]) %set axes
%plotting deaths
plot(deaths(:,1),deaths(:,2), 'rx','linewidth',1)

%plotting pumps
for k = 1:numberpumps
    sz = find(sorted_tally == pumptally(k)); %returns nonzero entries
of sorted_tally (whether or not they are equal)

plot(pumps(k,1),pumps(k,2),'bo','markersize',sz(1)*2,'linewidth',1)
%plot blue circles (pumps) with their number of tallies

text(pumps(k,1),pumps(k,2),num2str(pumptally(k)),'color','k','fontsiz
e',sz(1)+10,'linewidth',2) %display the pumps point as a number that
is the number of tallies
end


%labels
title('John Snow''s Cholera map,1854')
xlabel('Easting')
```

```
ylabel('Northing')
legend('Pumps','Deaths')

%% Part Seven

prompt = 'Insert name: '; %prompts user to insert name
name = input(prompt,'s'); %^
datestr = date; %displays date

%display date without dashes
datestr(datestr=='-') = [];

fname = ['report_',datestr,'.txt']; %names the file with report #,
date, and as a .txt file
fid = fopen(fname,'w'); %opens file with data below

%fprintf statements for tallies and proximal fatalities
fprintf(fid,'Analysis of John Snow''s 1854 Cholera Data\n');
fprintf(fid,'Produced by %s on %s\n\n',name, datestr);
fprintf(fid,'%50s','Tallies: Pumps and Their Proximal Fatalities');
fprintf(fid,'\n');
fprintf(fid,'%10s%15s%15s%15s\n','Pump #','Easting','Northing','Num
Cases');
fprintf(fid,'%10s%15s%15s%15s\n','------','-------','---------','----
-----');
for k = 1:numberpumps %data for tally table
fprintf(fid,'%8d%18.5f%15.5f%12d\n',k,pumps(k,1),pumps(k,2),pumptally
(k));
end
%fprintf statements for sorted tallies and proximal fatalities (by
pump #)
fprintf(fid,'\n');
fprintf(fid,'%55s','Sorted Tallies: Pumps and Their Proximal
Fatalities');
fprintf(fid,'\n');
fprintf(fid,'%10s%15s%15s%15s\n','Pump #','Easting','Northing','Num
Cases');
fprintf(fid,'%10s%15s%15s%15s\n','------','-------','---------','----
-----');

for k = numberpumps:-1:1 %data for sorted tally table (by number of
cases)
fprintf(fid,'%8d%18.5f%15.5f%12d\n',order(k),pumps(order(k),1),pumps(
order(k),2),sorted_tally(k));
end
```

TABLES AND IMAGES (HW 9)





Editor – /Users/bellabarbera/Desktop/Desktop – Bella's MacBook Pro/CS

johnsnow.m | report_13Nov2017.txt | report_12Dec2017.txt

```
Analysis of John Snow's 1854 Cholera Data
Produced by Bella on 12Dec2017

      Tallies: Pumps and Their Proximal Fatalities
    Pump #        Easting       Northing      Num Cases
    ------        -------       --------      ---------
         1         8.65120       17.89160              1
         2        10.98478       18.51785              1
         3        13.37819       17.39454             12
         4        14.87983       17.80992             24
         5         8.69477       14.90547              6
         6         8.86442       12.75354             61
         7        12.57136       11.72717            359
         8        10.66097        7.42865             16
         9        13.52146        7.95825             27
        10        16.43489        9.25213             64
        11        18.91439        9.73782              2
        12        16.00511        5.04684              2
        13         8.99944        5.10102              4

    Sorted Tallies: Pumps and Their Proximal Fatalities
    Pump #        Easting       Northing      Num Cases
    ------        -------       --------      ---------
         7        12.57136       11.72717            359
        10        16.43489        9.25213             64
         6         8.86442       12.75354             61
         9        13.52146        7.95825             27
         4        14.87983       17.80992             24
         8        10.66097        7.42865             16
         3        13.37819       17.39454             12
         5         8.69477       14.90547              6
        13         8.99944        5.10102              4
        12        16.00511        5.04684              2
        11        18.91439        9.73782              2
         2        10.98478       18.51785              1
         1         8.65120       17.89160              1
```

Homework 14

Overview: In part one of homework 14 the raw data, means, and standard deviation are displayed in a table. In a conclusion statement the target diameter is displayed, which line diameters run closer to the target, and which line diameters are less variable are displayed. In part two (handedness) handtime.dat data is imported and used to create a graph that shows left versus right handedness (percentage measured) based on time. Two lines of best fits for both left and right handed people are also calculated and displayed. In part three two rosters are read into the script and used to create two functions that create a table of member vs. tension. The polynomials are then solved and plotted.

Skills: polyfit, Polyval, polyder, Diff, ezplot, solve, struct

```
% HW 14 Part 1
% November 15th,2017
% Bella Barbera

clc;
clearvars;
close all
%% Part 1

% Storing Raw Measurements
prod_AB = [15.94 15.98 15.94 16.16 15.86 15.86 15.90 15.88
           15.96 15.94 16.02 16.10 15.92 16.00 15.96 16.02]; %creates
a variable that stores the measurements in a 2x8 matrix

% Mean and Standard Deviation
MEAN = mean(prod_AB,2); %computes both means of the above data
STD = std(prod_AB,0,2); %returns the standard deviation
lines = {'A' , 'B'}; %sotres production line in a 1x2 cell array

% Name, Date and Table of Raw Data
fprintf('QA Sampling results for Lines A and B\n')
name = 'Bella Barbera';
fprintf('%10s\n%10s\n\n',name,date)
fprintf('%10s\n','/*Raw Data*/')

for k = 1:length(lines) %For loop using k (iterator variable) to
index into the cell array
    fprintf('%10.2f',prod_AB(k,:)) %print all eight measurements
    fprintf('\n')
end

% Summary of Statistics
fprintf('\n%10s\n','/*Summary Stats*/') % Titles/display
fprintf('%20s%10s\n','A','B')
fprintf('%22s%10s\n','-----','-----')
fprintf('%s%13.2f%10.2f', 'Means (mm)', MEAN(1), MEAN(2))
fprintf('\n%10s%13.2f%10.2f', 'SD(mm)', STD(1), STD(2))

% Target
TARGET = 16;
delta = abs(MEAN-TARGET); %compute delta (variable)

closer = find(delta == min(delta)); %identify which line's mean is
closer to target
precision = find(delta == min(delta)); %identify which standard
deviation is smaller

fprintf('\n\n/*Conclusion*/') %display title
```

```
fprintf('\nTarget Diameter = %.2f mm', TARGET) %display target number

fprintf('\nOn average, Line %c diameters run closer to target. \nLine
%c diameters are less variable.',lines{closer},lines{precision})
%display which line is closer to target/are less variable

                                % Handedness
% November 15th, 2017
% Bella Barbera

clc;
clearvars;
close all
%% Part 1 - Acquire and parse the data

fid = fopen('handtime.dat','r'); % Loading the file into the script
handtime = textscan(fid,'%18c%d%d%f'); %assigning contents of file to
the target variable (handtime)

% Assignment statements for columns of handtime.dat
period = handtime{1};
periodnum = handtime{2};
handedness = handtime{3};
count = handtime{4};

%% Part 2 - Aggregate the tallies

matrix = [count(1:2:length(count)) count(2:2:length(count))]; %
matrix that contains original count values
matrix = [matrix(:,1) matrix(:,2) sum(matrix')']; % matrix of right,
left and sum
percent_right = matrix(:,1)./matrix(:,3).*100; % calculate right hand
percentage
percent_left = matrix(:,2)./matrix(:,3).*100;  %calculate left hand
percentage

%% Part 3 - Plot the Data

figure(1) %create figure
hold on %add to figure

period = period(1:2:end,:); %extract the period names from cell array
producing a matrix of characters
period = cellstr(period); % use cellstr to turn matrix back into cell
array of strings
labels = period; %labels = new matrix removing duplicate period names
x = [1:length(period)];
x = x';
```

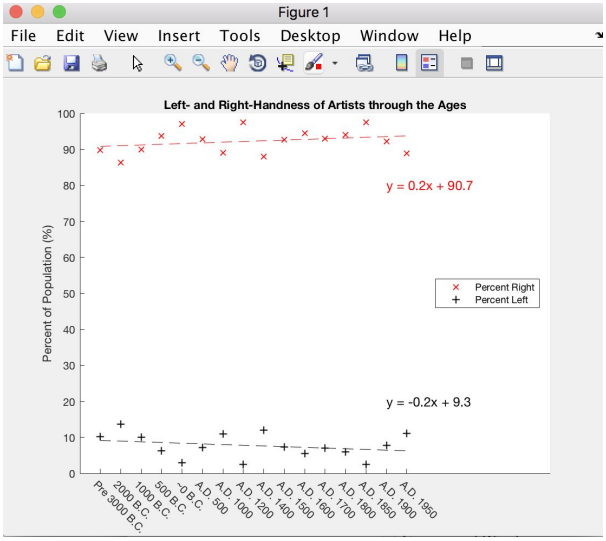
```
xlim([0 23])

% Plotting
plot(percent_right,'rx','linewidth',1) %plotting percent right
plot(percent_left,'k+','linewidth',1) %plotting percent left

% Coeffecients of line of best fit for right hand
coeff_right = polyfit(x,percent_right,1);
best_r = polyval(coeff_right,x);
plot(x,best_r,'r--')

% Coeffcients of line of best fit for left hand
coeff_left = polyfit(x,percent_left,1);
best_l = polyval(coeff_left,x);
plot(x,best_l,'k--')

set(gca,'xtick',x,'xticklabel',labels,'xticklabelrotation',-45) %sets
value of the specified property for the graphics

% Titles
title('Left- and Right-Handness of Artists through the Ages')
ylabel('Percent of Population (%)')
legend('Percent Right','Percent Left','location','east')

if coeff_right(2) >= 0
    sign = '+';
else
    sign = '-';
end

% Equations and plotting
equation_right = sprintf('y = %.1fx %c %.1f', coeff_right(1), sign,
coeff_right(2));
text(15,80, equation_right, 'color','red','fontsize',12)

if coeff_left(2) >= 0
    sign = '+';
else
    sign = '-';
end

equation_left = sprintf('y = %.1fx %c %.1f', coeff_left(1), sign,
coeff_left(2));
text(15,20, equation_left, 'color','black','fontsize',12)

                    % HW 14 Part 3
% November 15th,2017
```

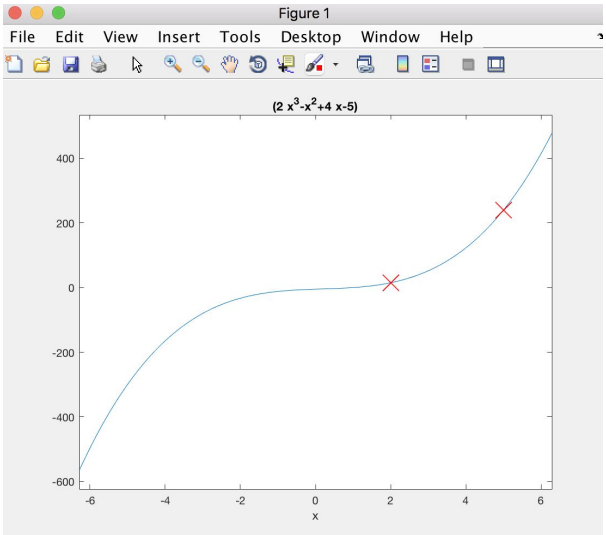
```
% Bella Barbera

clc;
clearvars;
close all

%% Section 1 -  Set Operations
% Importing the data from each roster file
A = importdata('rosterA.xlsx');
B = importdata('rosterB.xlsx');

% Indexing for columns
classA = A(2:end,1);
classB = B(2:end,1);
degreesA = A(2:end,2);
degreesB = B(2:end,2);
majorsA = A(2:end,3);
majorsB = B(2:end,3);

% Set functions
classAandB = intersect(classA,classB) %intersect returns the values
common to the two vectors with no repititions
degreesAnotB = setdiff(degreesA,degreesB) %returns the values in A
that are not in B with no repititions
majorsAorB = union(majorsA,majorsB) %reutrns the combined values of
the two vectors with no repititions
classnotboth = setxor(classA,classB) %returns the values that are not
in the intersection of A and B with no repititions

%% Section 2 - Solve Systems of Linear Equations

% Assignment statements for f1 and f2
f1 = 1000;
f2 = 5000;

% Coefficient a defined
coeff_a = [.5 1 0 0 0 0 0
    .866 0 0 0 0 0 0
    -.5 0 .5 1 0 0 0
    .866 0 .866 0 0 0 0
    0  -1 -.5 0 .5 1 0
    0 0 .866 0 .866 0 0
    0 0 0 -1 -.5 0 .5];

% Constant b defined
const_b = [f1 -.433*f1-.5*f2 -f1 0 0 f2 0]';

tension = coeff_a\const_b; %computing the tensions vector x
```

```
fprintf('%8s%15s\n','Member', 'Tension(N)',... %titles
                    '------', '----------')
fprintf('%5d%17.2f\n',[1:length(tension);tension']) %first row =
integers from 1 to the lengh of x and second row = values of x

%% Section 3 - Attaway Chap 14, Exercise 35 solve and subs

% Assigning syms to  be solved
syms x y z

% Setting equations to be solved
r = solve(2*x + 2*y + z == 2,y + 2*z == 1,x + y + 3*z == 3);

% Displaying all three roots (answers)
disp([r.x,r.y,r.z])

% Anonymous function
f1 = @(x,y,z) (2*x + 2*y + z);

% Checking the answers (that the result does =2)
check = subs(f1,r);

% Coefficients
A = [2 2 1; 0 1 2; 1 1 3];

% Matrix of solutions (matrix multiplication)
b = [2 1 3]';
x = A\b

%% Section 4 - Polynomials and Calculus
clc

% Define anonymous function
f = @(x) (2*x.^3 - x.^2 + 4*x -5);

% Graph of f
ezplot(f) %calling ezplot with f as only input argument
hold on %add to graph

% Marking the points
plot([2,5],[f(2),f(5)],'rx','markersize',20,'linewidth',1)

% Coefficient vectors of f(x)
P = [2 -1 4 -5];

% Value of degree with polyval
```

```
f(5) == polyval(P,5) %polyval returns the value of a polynomial
evaluated at 5

% Derivative
derivative = polyder(P) %differentiates polynomial

% Solutions
solution = roots(P) %finds polynomial roots

% x values from 2 to 5
x = 2:5;
y = f(x)

% Finding area using quad
area1 = quad(f,2,5) %numerically evaluate the integral from 2,5

%Finding area using traps
area2 = trapz(y) %computes approximation of y via trapezoidal method
with unit spacing

syms x f
f = x^3 + 2*x^2 - 4*x + 3 %defining f
derivative2 = diff(f) %difference and approximate derivative
area = int(f,2,5) % computes definite integral between 2 and 5
```

IMAGES AND TABLES (HW14)






```
Command Window
QA Sampling results for Lines A and B
Bella Barbera
13-Dec-2017

/*Raw Data*/
     15.94     15.98     15.94     16.16     15.86     15.86     15.90     15.88
     15.96     15.94     16.02     16.10     15.92     16.00     15.96     16.02

/*Summary Stats*/
                   A         B
               -----     -----
Means (mm)      15.94     15.99
    SD(mm)       0.10      0.06

/*Conclusion*/
Target Diameter = 16.00 mm
On average, Line B diameters run closer to target.
Line B diameters are less variable.>>
```

HOMEWORK 12

Overview: In homework 12 part one two equations are created one of degree 3 and 4. These are then plotted on a 2x1 subplot with a line of best fit. In part two pumps and deaths are loaded into the script and used to create a meshgrid with a colorbar showing the pumps and their number as well as the deaths. In part three four figures are created that contain graphs of sqrt x, x^2, lnx, y=e^x. In part four a 3D plot, pie chart, and bar graph are created based off of data from turbine.dat. In **part two** and AVI file is created that creates a video animation.

Skills:

- Polyfit, polyval, cell2mat, meshgrid, set, colorbar, feval, bar, pie

```
% HW 12
%November 15th
%Bella Barbera

clc; clearvars; close all

%% Section 1: Subplots & Best Fit Curves

figure(1),clf %create figure, clear figure
hold on %add to plot


%raw data
time = 0:3:24; %time range (0 to 24 seconds)
flow = [800 980 1090 1520 1920 1670 1440 1380 1300]; %flow in cubic
feet of water (data)

%plotting the data (dots)
coeff3 = polyfit(time,flow,3); %finds coefficients
bestfit3 = polyval(coeff3,time); %y values of line of best fit
subplot(1,2,1) %creating plot of the two graphs by into an m-by-n
matrix of small axes, selects the p-th axes for the current plot
(m,n,p)
plot(time,flow,'ko') %plotting the data
equation1 = sprintf('Mystical River Flow Rate\n Q(t) = %.1ft^3 +
%.1ft^2 + %.1ft + %.1f', coeff3(1), coeff3(2), coeff3(3), coeff3(4));
%displaying the equation

%best fit line 3rd power (dashed line)
hold on %add to graph
plot(time,bestfit3, 'b--','linewidth',1) %plotting dashed line
axis([0 25 600 2000]) %setting axes
xlim([-1,25]) %sets limit on x-axis

%axis labels
xlabel('Time (hr)')
ylabel('Flow rate (m^3/s)')
```

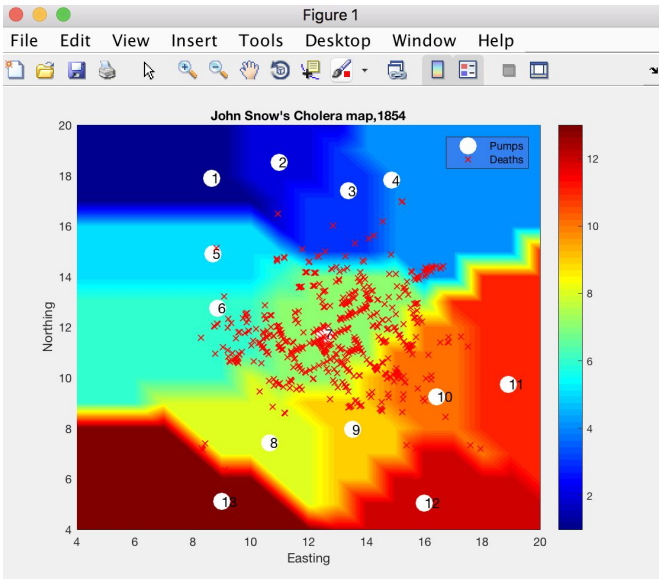
```
title(equation1)
legend('Measured','Best fit')

%best fit line 4th power (dashed line and dots)
coeff4 = polyfit(time,flow,4); %finds coefficients of equation
bestfit4 = polyval(coeff4,time); %y values of line of best fit
subplot(1,2,2) %plot two plots
plot(time,flow,'ko') %plot data (dots)
axis([0 25 600 2000]) %set axes
xlim([-1,25]) %sets limits on x-axis
hold on %add to graph
plot(time,bestfit4,'b--','linewidth',1) %plot dashed line
equation2 = sprintf('Mystical River Flow Rate\n Q(t) = %.1ft^4 +
%.1ft^3 + %.1ft^2 + %.1ft + %.1f', coeff4(1), coeff4(2), coeff4(3),
coeff4(4),coeff4(5)); %displaying equation

%axis labels
xlabel('Time (hr)')
ylabel('Flow rate (m^3/s)')
title(equation2)
legend('Measured','Best fit')

%% Section 2: Voronoi diagram (Thiessen Polygons): Cholera Dataset

pumpsfid = fopen('pumps.txt','r'); %opening pumps
pumps = textscan(pumpsfid,'%f , %f','Headerlines',1); %loading in
pumps
fclose(pumpsfid); %closing pumps file
pumps = cell2mat(pumps); %converting each cell array into a matrix of
doubles

deathsfid = fopen('deaths.txt','r'); %opening deaths
deaths = textscan(deathsfid,'%f , %f','Headerlines',1); %loading in
deaths
fclose(deathsfid); %closing file deaths
deaths = cell2mat(deaths); %converting each cell array into a matrix
of doubles

closestpump = zeros(20,20); %defining closest pump as a 20x20 matrix
of zeros

for rows = 1:20 %finding the pump closest to each pixel in the 20x20
square
    for columns = 1:20 %for columns 1-20
    dist = sqrt((columns - pumps(:,1)).^2 + (rows - pumps(:,2)).^2);
%computing the distances from each pixel to all pumps
    [~,closestpump(rows,columns)] = min(dist); %assigning the index
of the closest pump
```

```
    end
end

figure(1),clf %creating figure 2

%plotting the data
X = 1:20;
Y = 1:20;
[X,Y] = meshgrid(X,Y); %produces rectangular grid
colormap(jet) %colors plot
pcolor(X,Y,closestpump) %creates a checkerboard matrix plot of
closest pump
shading interp %controls the color shading by setting it to
interpolated
hold on %add to figure

grid on %grid on
for k = 1:length(pumps) %plotting all pumps
    plot1 =
plot(pumps(k,1),pumps(k,2),'w.','markersize',50,'linewidth',1);
%plotting pumps as white circles

text(pumps(k,1),pumps(k,2),num2str(k),'fontsize',12,'color','k','line
width',7)%plotting numbered pumps
end
plot2 = plot(deaths(:,1),deaths(:,2), 'rx','linewidth',1); %plotting
deaths as x's
axis([4 20 4 20]) %setting axes

%labels
title('John Snow''s Cholera map,1854')
xlabel('Easting')
ylabel('Northing')
leg = legend([plot1 plot2], 'Pumps', 'Deaths');
set(leg, 'color',[51 128 240]/255) %set object properties
colorbar %display colorbar

%% Section 3: Log Plots

x = .05:.1:10; %assign x, a vector of values from 0.05 to 10
sqr = @(x)(x.^2); %define sqr as a function
functions = {@sqrt,sqr,@log,@exp}; %define functions a cell array of
function handles
graphs = {@plot,@semilogx,@semilogy,@loglog}; %cell array of
functions for each subplot
title1 = {'\surdx','x^2','lnx','e^x'}; %assign a cell array for the
title strings
```

```
for k = 1:length(functions) % for length of all of functions
    figure %create figure
    for j = 1:length(functions) %for length of all functions
        subplot(2,2,j) %make new 2x2 subplot, then assign current
graph position j
        y = feval(functions{k},x); %execute the functions by
substituting in x
        feval(graphs{j},x,y) %executes function which is to graph x
and y
        if j == 1 %put a title on the first subplot in every figure
            title(['y = ' title1{k}])
        end
    end
end

%% Section 4: Graphing Turbine Data

clc; clearvars; %clear workspace and command window

load 'turbine.dat' %load turbine data

%assigning x y z for the graph
x = turbine(:,1);
y = turbine(:,2);
z = turbine(:,3);

%plotting the data
subplot(2,2,1)
plot3(x,y,z,'ko')
grid on

%labels
title('Wind-Generated Electricity Production')
xlabel('Blade Diameter (ft)')
ylabel('Wind Velocity (mph)')
zlabel('Electricity (kwh/yr)')

% Plotting the Pie Chart
mask1 = x == 5; %assign mask, will be used to index when the diameter
is 5
mask2 = x == 10;
diameter5 = sum(z(mask1)); %index into output using mask to parse
data by blade size
diameter10 = sum(z(mask2));
diameters = [diameter5 diameter10]; %actually create the pieplot

%plotting the data
subplot(2,2,2) %plotting the pieplot in the second subplot place
```

```
pie(diameters)
title('Percentage of Electricity Produced by Blade Size') %titles
legend('5ft Blade', '10ft Blade', 'location','southwest') %legends

%Plotting the bar graph
mask3 = y == 5;
mask4 = y == 10;
mask5 = y == 15;
mask6 = y == 20;
windspeed5 = z(mask3);
windspeed10 = z(mask4);
windspeed15 = z(mask5);
windspeed20 = z(mask6);

subplot(2,2,3) %putting bar chart in third subplot place
bar([windspeed5 windspeed10 windspeed15 windspeed20]') %creating bar
chart
ylabel('Kilowatt-hours per year') %labels
xlabel('Windspeed')
title('Wind-Generated Electricity Production')
set(gca,'xticklabel',{'5mph','10mph','15mph','20mph'})
legend('5ft Blade','10ft Blade','location','northwest')

%% Bella Barbera
% November 27th 2017
%AVI file

%%

%creating function ani2avi so we can place two helper functions in
the same file
function ani2avi

%transformation matrices
refx = [-1 0; 0 1]; %x coordinates
refy = [1 0; 0 -1]; %y coordinates
ref0 = refx*refy; %used to reflect coordinates through the origin

%defining anonymous functions
scale = @(sx,sy)([sx 0; 0 sy]); %define function scale which scales
up or down
shearx = @(k)([1 k;0 1]); %define the functions shearx/y which shear
the plot in the y or x direction (slanting)
sheary = @(k)([1 0;k 1]);
rotccw = @(t)([cos(t) sin(t); -sin(t) cos(t)]); %rotates
counterclockwise about the origin
```

```
%loading mypts
load mypts1.mat

pts = [X;Y]; %define points as the combination of x and y from
mypts.mat

%saving as animate.avi
vidObj = VideoWriter('animate.avi');
vidObj.FrameRate = 1; %defining frame rate of the video
open(vidObj); %opening file for writing

%updating pts
makefig1 %creating figure to display points on
helper(pts,vidObj); %plot original points/add to video
pts = scale(.4,.6)*pts;
helper(pts,vidObj);%
pts = shearx(3)*pts;
helper(pts,vidObj);
pts = sheary(-1)*pts; %reflect over y-axis, plot, add to video
helper(pts,vidObj);
pts = rotccw(35)*pts; %rotate all points 35 degrees, plot, add to
video
helper(pts,vidObj);
pts = scale(2,1)*pts;
helper(pts,vidObj);
pts = sheary(3)*pts;
helper(pts,vidObj);
plotme(pts);
close(vidObj);
end

%creating plotme and helper functions
function h = plotme(pts)
h = plot(pts(1,:),pts(2,:),'bp','markersize',30); %plot x,y
coordinates given in points/ assign plot handle
end
```




```
function helper(pts,obj)
h = plotme(pts);
currFrame = getframe;
writeVideo(obj,currFrame);
End
```

TABLES AND IMAGES

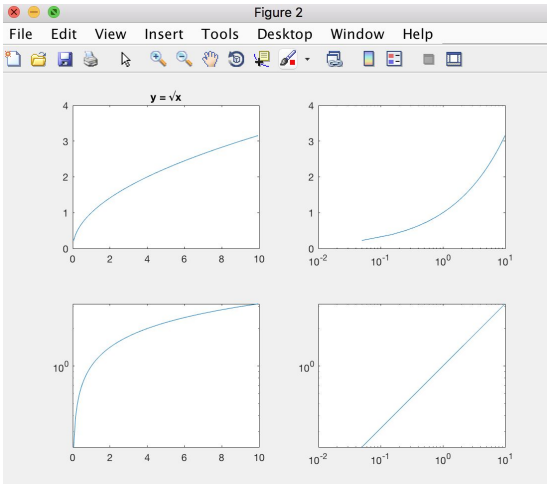


Fig 4