# CMPE 362

# INTRODUCTION TO SIGNAL PROCESSING FOR COMPUTER ENGINEERING

**SPRING 2019**

**HW1**

**Getting started to MATLAB**

**Gurkan Demir**

**2015400177**

- **Problem-1**

  *In this problem, I am expected to create a vector from -100 to 100 and create given 8 functions. And plot those functions with x. Since I wanted to draw more smooth plot, I incremented values of vector by 0.1. Below you can find code of this problem. You can reach its figure at the end of problem4.*

```
% Create vector from -100 to 100
x = -100:0.1:100;

% Set the name of figure for problem 1
f1 = figure('Name', 'Problem-1', 'NumberTitle', 'off');
figure(f1);

% Create functions and draw subplots with their titles.
subplot(4, 2, 1);
y1 = sin(x);
figure(1);
plot(x, y1);
title('y1 = sin(x)');

subplot(4, 2, 2);
y2 = sin(50.*x);
plot(x, y2);
title('y2 = sin(50x)');

subplot(4, 2, 3);
y3 = 50.*sin(x);
plot(x, y3);
title('y3 = 50sin(x)');

subplot(4, 2, 4);
y4 = sin(x)+50;
plot(x, y4);
title('y4 = sin(x)+50');

subplot(4, 2, 5);
y5 = sin(x+50);
plot(x, y5);
title('y5 = sin(x+50)');

subplot(4, 2, 6);
y6 = 50.*sin(50.*x);
plot(x, y6);
title('y6 = 50sin(50x)');

subplot(4, 2, 7);
y7 = x .* sin(x);
plot(x, y7);
```

```
title('y7 = xsin(x)');

subplot(4, 2, 8);
y8 = sin(x) ./ x;
plot(x, y8);
title('y8 = sin(x)/x');
```

- **Problem-2**

  *In this problem, I am expected to create a vector from -20 to 20 and create given 9 functions. And plot those functions with x. Below you can find code of this problem. You can reach its figure at the end of problem4.*

```
% Create a vector from -20 to 20
x = -20:20;

% Set name of figure for problem 2
f2 = figure('Name', 'Problem-2', 'NumberTitle', 'off');
figure(f2);

% Create functions and draw subplots with their titles
subplot(5, 2, 1);
y1 = sin(x);
plot(x, y1);
title('y1 = sin(x)');

subplot(5, 2, 2);
y2 = sin(50.*x);
plot(x, y2);
title('y2 = sin(50x)');

subplot(5, 2, 3);
y3 = 50.*sin(x);
plot(x, y3);
title('y3 = 50sin(x)');

subplot(5, 2, 4);
y4 = sin(x)+50;
plot(x, y4);
title('y4 = sin(x)+50');

subplot(5, 2, 5);
y5 = sin(x+50);
plot(x, y5);
title('y5 = sin(x+50)');

subplot(5, 2, 6);
y6 = 50.*sin(50.*x);
plot(x, y6);
title('y6 = 50sin(50x)');
```

```
subplot(5, 2, 7);
y7 = x .* sin(x);
plot(x, y7);
title('y7 = xsin(x)');

subplot(5, 2, 8);
y8 = sin(x) ./ x;
plot(x, y8);
title('y8 = sin(x)/x');

subplot(5, 2, 9);
y9 = y1+y2+y3+y4+y5+y6+y7+y8;
plot(x, y9);
title('y9 = y1+y2+y3+y4+y5+y6+y7+y8');
```

- ## Problem-3

  *In this problem, I am expected to generate 41 gaussion distributed random numbers and create given 10 functions. And plot those functions with x. Below you can find code of this problem. You can reach its figure at the end of problem4.*

```
% Generate 41 random number from gaussion distributed random numbers
z = randn(1,41);

% Set name of figure for problem 3
f3 = figure('Name', 'Problem-3', 'NumberTitle', 'off');
figure(f3);

% Create functions and draw subplots with their titles
subplot(5, 2, 1);
y10 = z;
plot(x, y10);
title('y10 = z');

subplot(5, 2, 2);
y11 = z+x;
plot(x, y11);
title('y11 = z+x');

subplot(5, 2, 3);
y12 = z+sin(x);
plot(x, y12);
title('y12 = z+sin(x)');

subplot(5, 2, 4);
y13 = z.*sin(x);
```

```
plot(x, y13);
title('y13 = zsin(x)');

subplot(5, 2, 5);
y14 = x.*sin(z);
plot(x, y14);
title('y14 = xsin(z)');

subplot(5, 2, 6);
y15 = sin(x+z);
plot(x, y15);
title('y15 = sin(x+z)');

subplot(5, 2, 7);
y16 = z .* sin(50.*x);
plot(x, y16);
title('y16 = zsin(50x)');

subplot(5, 2, 8);
y17 = sin(x+50.*z);
plot(x, y17);
title('y17 = sin(x+50z)');

subplot(5, 2, 9);
y18 = sin(x)./z;
plot(x, y18);
title('y18 = sin(x)/z');

subplot(5, 2, 10);
y19 = y11+y12+y13+y14+y15+y16+y17+y18;
plot(x, y19);
title('y19 = y11+y12+y13+y14+y15+y16+y17+y18');
```

- ## Problem-4

  *In this problem, I am expected to generate 41 uniformly distributed random number and create given 10 functions. And plot those functions with x. Below you can find code of this problem. You can reach its figure at the end of this problem.*

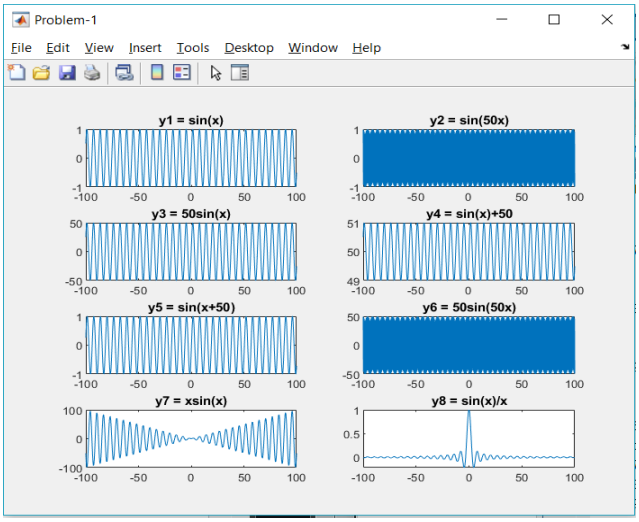
```
% Generate 41 random number from uniformly distributed random numbers
z = rand(1,41);

% Set the name of figure for problem 4
f4 = figure('Name', 'Problem-4', 'NumberTitle', 'off');
figure(f4);

% Create functions and draw subplots with their titles
subplot(5, 2, 1);
```

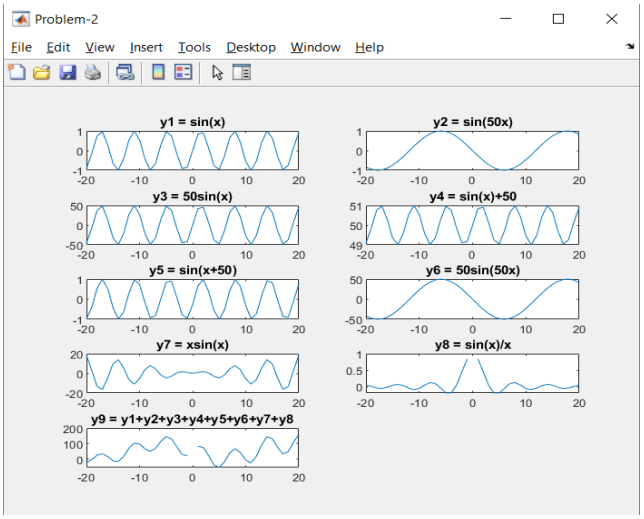
```
y20 = z;
plot(x, y20);
title('y20 = z');

subplot(5, 2, 2);
y21 = z+x;
plot(x, y21);
title('y21 = x+z');

subplot(5, 2, 3);
y22 = z+sin(x);
plot(x, y22);
title('y22 = z+sin(x)');

subplot(5, 2, 4);
y23 = z .* sin(x);
plot(x, y23);
title('y23 = z*sin(x)');

subplot(5, 2, 5);
y24 = x .* sin(z);
plot(x, y24);
title('y24 = x*sin(z)');

subplot(5, 2, 6);
y25 = sin(x+z);
plot(x, y25);
title('y25 = sin(x+z)');

subplot(5, 2, 7);
y26 = z .* sin(50.*x);
plot(x, y26);
title('y26 = z*sin(50x)');

subplot(5, 2, 8);
y27 = sin(x+50.*z);
plot(x, y27);
title('y27 = sin(x+50z)');

subplot(5, 2, 9);
y28 = sin(x)./z;
plot(x, y28);
title('y28 = sin(x)/z');

subplot(5, 2, 10);
y29 = y21+y22+y23+y24+y25+y26+y27+y28;
plot(x, y29);
title('y29 = y21+y22+y23+y24+y25+y26+y27+y28');
```

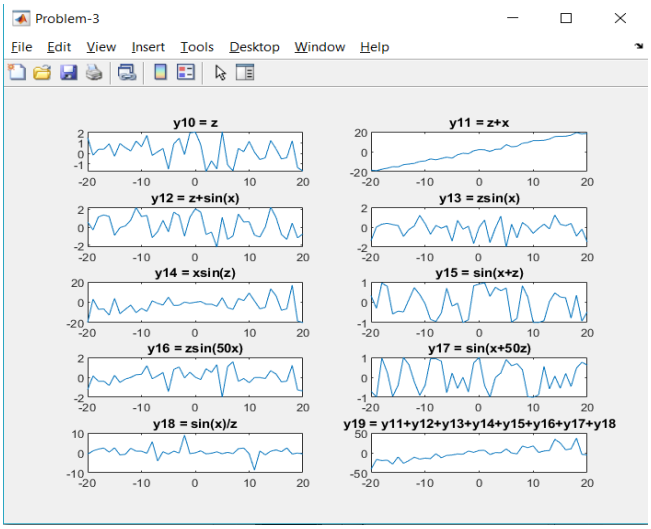
In the first problem, since I wanted to plot better and more smooth graph, I incremented values of vector by 0.1. In this context, I got more smooth graph. According to figure for problem1 below, I can conclude that if we increase the frequency we will see more waves. Main difference between y1 and y2 is sourced from frequency difference. Also, if we increase the amplitude, we will see an increase in the max value and decrease in the min value. Main difference between y1 and y3 is sourced from amplitude difference.

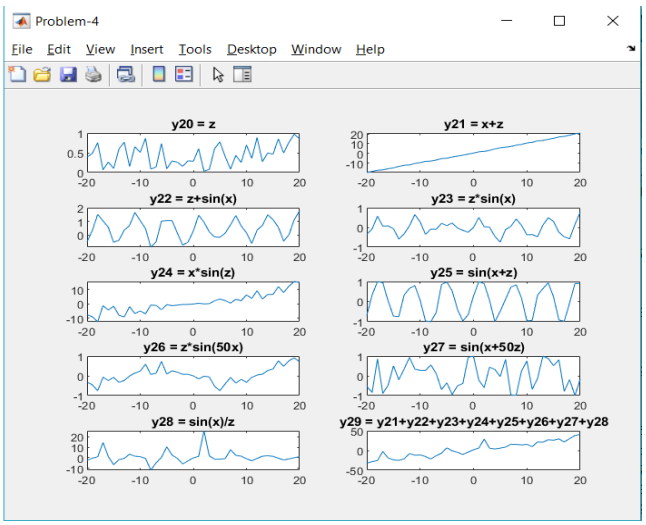
However in other problems(2-3-4), I incremented values of vector X by 1, since I am expected to generate a vector named Z which has 41 numbers and total number of elements of X must be 41. In this context, I got less smooth graph in problem2 than problem1, since total number of elements is really small. In other words, main difference between figure1 and figure 2 is sourced from number of elements of vector X.

In the third problem, I am expected to generate 41 zero mean, unit variance Gaussian distributed random numbers. However in problem4, I am expected to generate 41 uniformly distributed random numbers. Main difference between problem3 and problem4 is sourced from different distribution.



- **Problem-5**

  *In this problem, I am expected to generate 10000 gaussian distributed random var. According to those vector, I plot a histogram of 4 variables. All of them have mean 0, and have variance 1, 4, 16, and 256 respectively. Below you can find code of this problem. You can reach its figure at the end of problem8.*

```
% Set the name of figure for problem 5
f5 = figure('Name', 'Problem-5', 'NumberTitle', 'off');
figure(f5);

% Generate 10000 gaussian random var
z = randn(10000, 1);

% With mean 0, and variance 1
r1 = z;

% With mean 0, and variance 4
r2 = 2.*z;

% With mean 0, and variance 16
r3 = 4.*z;

% With mean 0, and variance 256
r4 = 16.*z;

% Draw histograms
subplot(2, 2, 1);
hist(r1);
title('r1');

subplot(2, 2, 2);
hist(r2);
title('r2');

subplot(2, 2, 3);
hist(r3);
title('r3');

subplot(2, 2, 4);
hist(r4);
title('r4');
```

- **Problem-6**

  *In this problem, I am expected to generate 10000 gaussian distributed random var. According to those vector, I plot a histogram of 4 variables. I generated variables have mean 10, 20, -10, -20, and have variance 1, 4, 1,*

*and 4 respectively. Below you can find code of this problem. You can reach its figure at the end of problem8.*

```
% Set name of figure for problem 6
f6 = figure('Name', 'Problem-6', 'NumberTitle', 'off');
figure(f6);

% Generate 10000 gaussian random var
z = randn(10000, 1);

% With mean 10, and variance 1
r6 = z+10;

% With mean 20, and variance 4
r7 = 2.*z+20;

% With mean -10, and variance 1
r8 = z-10;

% With mean -20, and variance 4
r9 = 2.*z-20;

% Draw histograms
subplot(2, 2, 1);
hist(r6);
title('r6');

subplot(2, 2, 2);
hist(r7);
title('r7');

subplot(2, 2, 3);
hist(r8);
title('r8');

subplot(2, 2, 4);
hist(r9);
title('r9');
```

- ## Problem-7

  *In this problem, I am expected to generate 10000 uniformly distributed random var. According to those vector, I plot a histogram of 4 variables. All of them have mean 0, and have variance 1, 4, 16, and 256 respectively. Below you can find code of this problem. You can reach its figure at the end of problem8.*

```
% Set name of figure for problem 7
f7 = figure('Name', 'Problem-7', 'NumberTitle', 'off');
figure(f7);

% Generate 10000 uniformly distributed random var
z = rand(10000, 1);

% For uniform dist, mean = (a+b)/2
% Variance = (b-a)^2 / 12
% ==> (a)+(b-a)(rand)

% With mean 0, and variance 1
r11 = (-sqrt(3))+ (2*sqrt(3)).*z;

% With mean 0, and variance 4
r21 = (-2*sqrt(3))+ (4*sqrt(3)).*z;

% With mean 0, and variance 16
r31 = (-4*sqrt(3))+ (8*sqrt(3)).*z;

% With mean 0, and variance 256
r41 = (-16*sqrt(3))+ (32*sqrt(3)).*z;

% Draw histograms
subplot(2, 2, 1);
hist(r11);
title('r11');

subplot(2, 2, 2);
hist(r21);
title('r21');

subplot(2, 2, 3);
hist(r31);
title('r31');

subplot(2, 2, 4);
hist(r41);
title('r41');
```

- ## Problem-8

  *In this problem, I am expected to generate 10000 uniformly distributed random var. According to those vector, I plot a histogram of 4 variables. I generated variables have mean 10, 20, -10, -20, and have variance 1, 4, 1, and 4 respectively. Below you can find code of this problem. You can reach its figure at the end of this problem.*

```
% Set name of figure for problem 8
f8 = figure('Name', 'Problem-8', 'NumberTitle', 'off');
```

```
figure(f8);

% Generate 10000 uniformly distributed random var
z = rand(10000, 1);

% For uniform dist, mean = (a+b)/2
% Variance = (b-a)^2 / 12
% ==> (a)+(b-a)(rand)

% With mean 10, and variance 1
r61 = (10-sqrt(3))+ (2*sqrt(3)).*z;

% With mean 20, and variance 4
r71 = (20-2*sqrt(3))+ (4*sqrt(3)).*z;

% With mean -10, and variance 1
r81 = (-10-sqrt(3))+ (2*sqrt(3)).*z;

% With mean -20, and variance 4
r91 = (-20-2*sqrt(3))+ (4*sqrt(3)).*z;

% Draw histograms
subplot(2, 2, 1);
hist(r61);
title('r61');

subplot(2, 2, 2);
hist(r71);
title('r71');

subplot(2, 2, 3);
hist(r81);
title('r81');

subplot(2, 2, 4);
hist(r91);
title('r91');
```

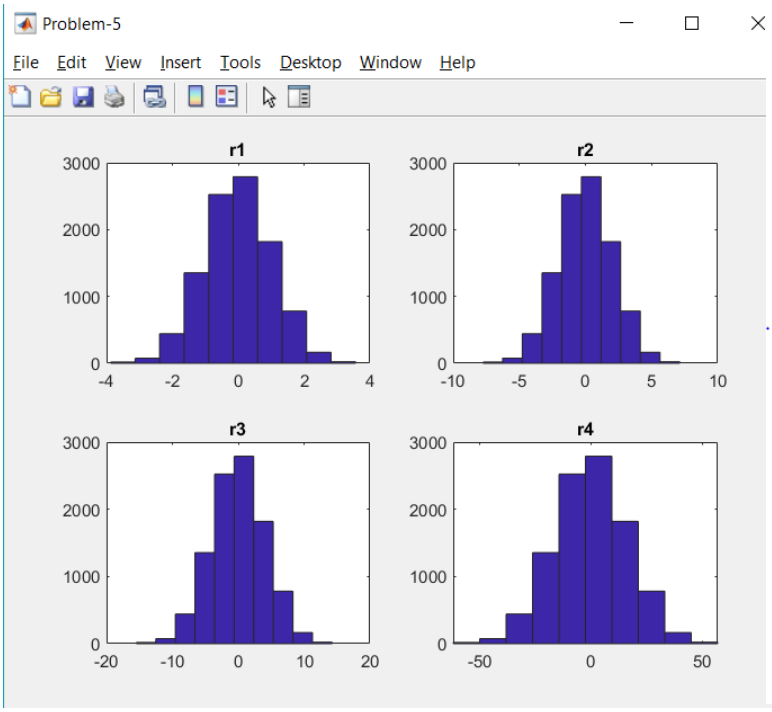
In those 4 problems(5-6-7-8), I am expected to generate 10000 random numbers and plot their histograms using hist function. But when I searched for hist on Web, most of individuals has given an advice about using histogram instead of hist. First I plotted all histograms in the same figure without using subplot. Then I decided to plot them in subplots in the same figure so I used hist for this function.

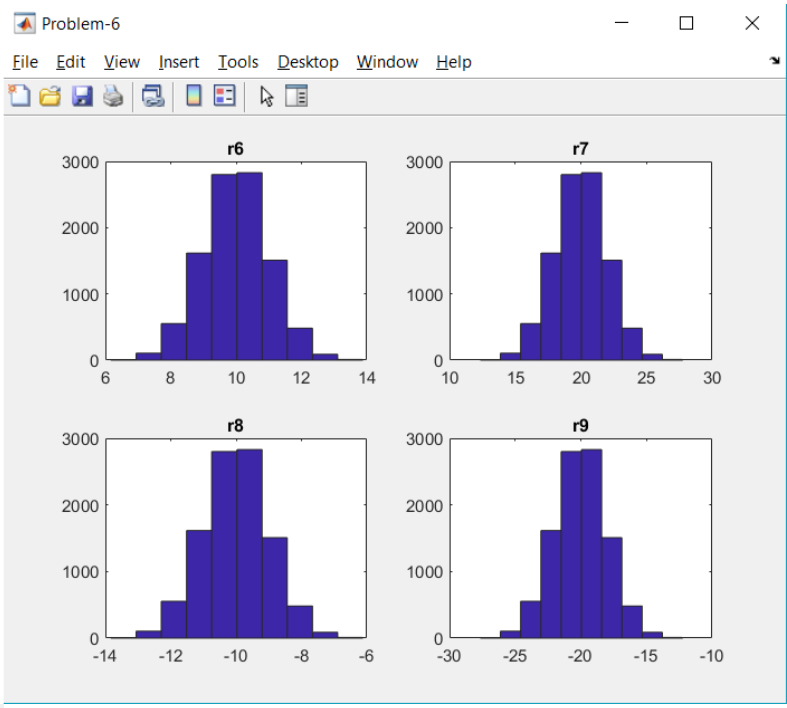
We all know that for Gaussian distribution, if we multiply random numbers with given standart deviation and add with given mean, we got expected function. In other words, function = (mean)+(std)*rand

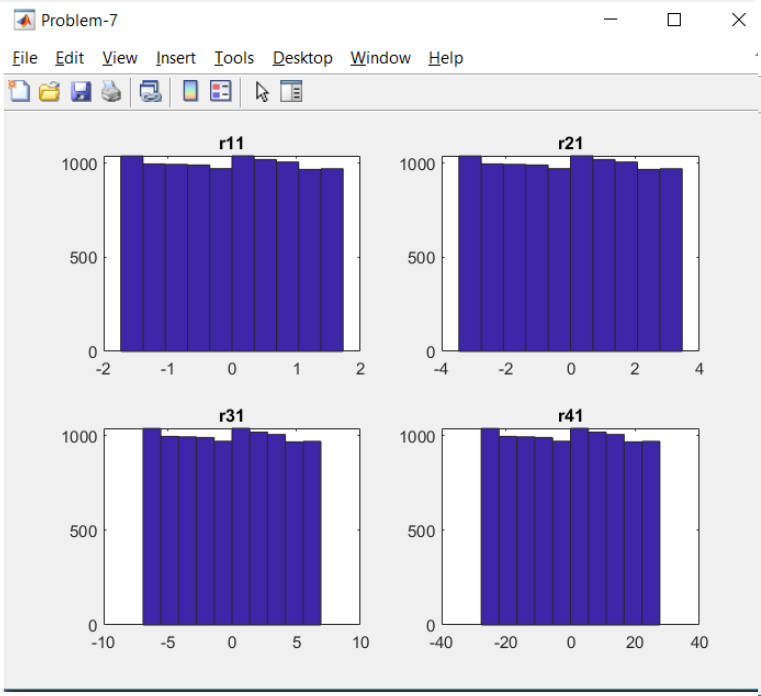
However, for uniformly distribution, things are little complex. It was one of the most challenging part in this project in my opinion. I used my experience about uniform

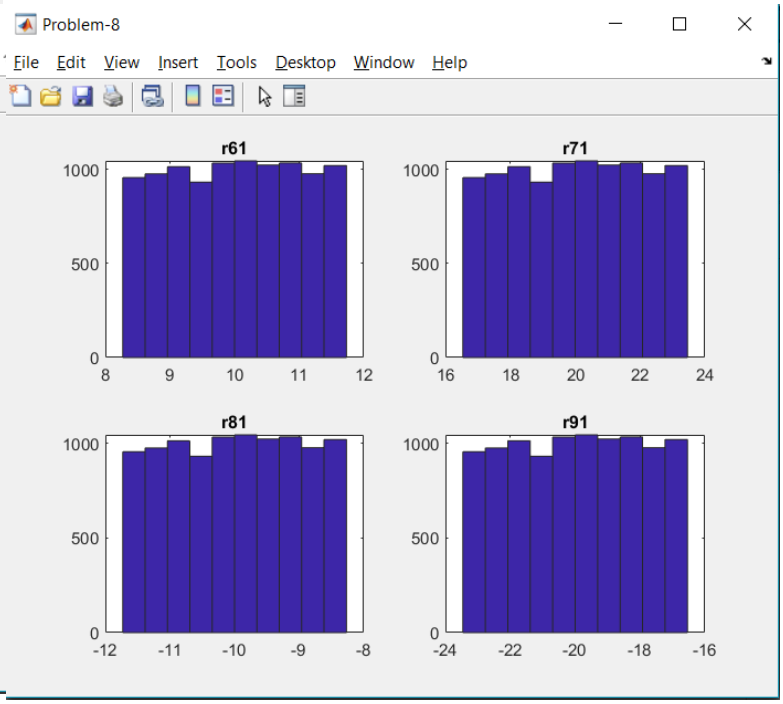
distributions that I gained from CMPE343 course. For a given mean and variance we have to find variables named a and b. For uniformly distributed numbers, mean = (a+b)/2, std = (b-a)^2 / 12. As a result our function = a + (b-a)*rand

Main difference between first two histograms(problem5-6) and last two histograms(problem7-8) is sourced from different distribution.



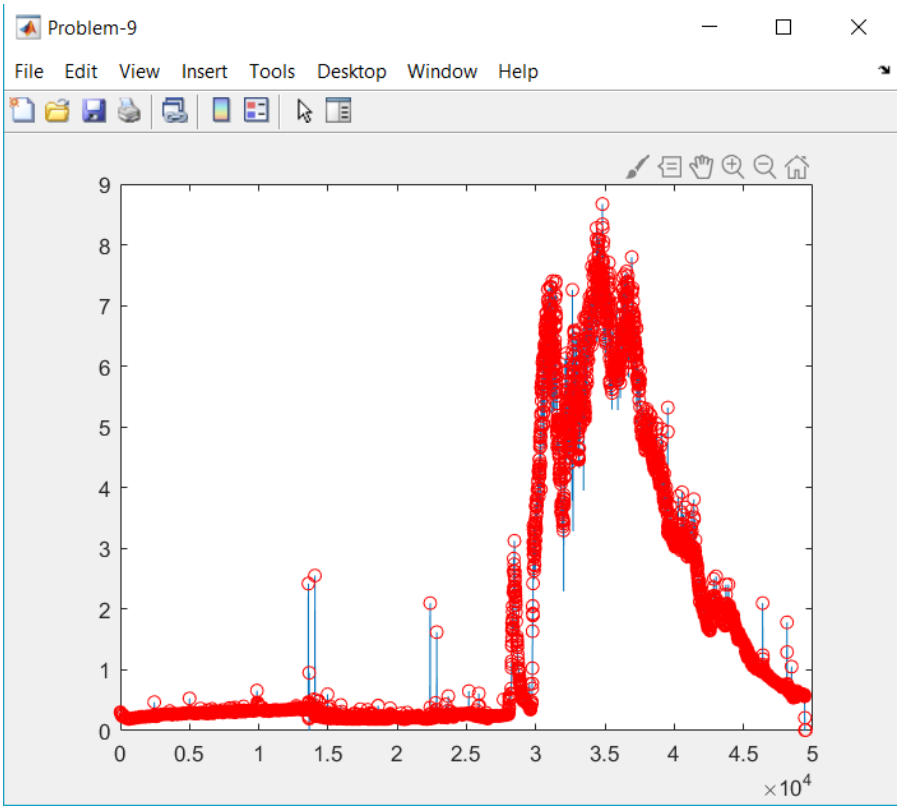



- **Problem-9**

  *In this problem, I am expected to read file named 'exampleSignal.csv' and according to values in the file, find peaks and their locations. Then plot the graph with peaks. For a better plot, I skipped first 3 elements in the file. Below you can find code of this problem. You can reach its figure at the end of this problem.*

```
% Set name of figure for problem 9
f9 = figure('Name', 'Problem-9',
'NumberTitle', 'off');
figure(f9);

% Open file named
exampleSignal.csv which is in the
same directory.
fileID =
fopen('exampleSignal.csv', 'r');

%Read file
A = fscanf(fileID, '%f');
A = A(4:length(A));

%Find peaks and their locations
[peaks, locs] = findpeaks(A);

% Plot values and peaks
plot(A);
hold on
plot(locs, peaks, 'ro');
```




According to my observations, function findPeaks does not miss any peak locations when it is called with default parameters. However there are some parameters for this function such as MaxPeakWidth, MinPeakDistance etc. If we call this function with those parameters algorithm filters some peak locations according to parameter values. In this context, we have a chance to filter peak locations according to their widht, height, distance etc.

- **Problem-10**

  *In this problem, I am expected to read image named 'lena.png' and turn it into gray format. According to those values, I am expected to find mean and standart deviation of values. Moreover, finding maximum value and its location are expected to be found as well as minimum value. Below you can find code of this problem. You can reach its result at the end of this problem.*

```
% Read image named lena.png which is in the same directory.
A = imread('./lena.png');

% Turn it into gray format
gray = rgb2gray(A);

% Find matrix's mean and std
mean = mean2(gray)
std = std2(gray)

% Find maximum value of matrix and its location
maximum = max(max(gray))
[max_x, max_y]=find(gray==maximum)

% Find minimum value of matrix and its location
minimum = min(min(gray))
[min_x, min_y]=find(gray==minimum)
```

**Results of problem10:**

- mean = 124.0425
- std = 47.8556
- max = 245
- max_x, max_y = 274, 396
- min = 25
- min_x, min_y = 72, 4

# Conclusion

Due to the fact that it was the first time that I implemented some functions in MATLAB, it was sometimes really difficult. Since I had not known the basics of this programming language, I searched many contents on mathworks.com via Web such as plotting more than one subplot in the same figure, reading file, finding max value etc. But after some time, I get used to implement problems in MATLAB and it become really easy to implement.

The most interesting that I found in MATLAB is if we do not put semicolon to the end of statement, system prints result to the console and workspace; otherwise it prints it only to workspace.

Moreover, in my opinion, plotting in MATLAB is more efficient and easiser than other programming languages like python. Another fact that I loved in MATLAB is help command. When I command ‘help _function_’, it documents that function and clearly explains how to use it. Sometimes it it really challenging for us to find usage of a library for other programming languages. Of course, all languages have some pros and cons, as a programmer we have to choose appropriate language for a given problem.