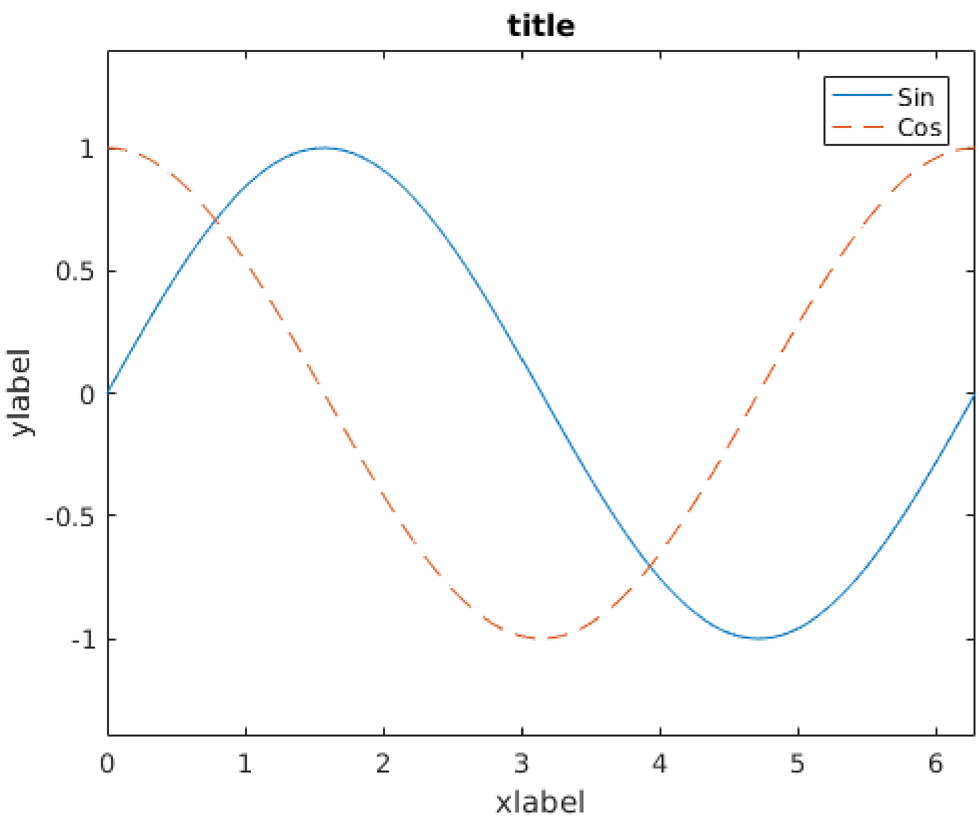
```
% ---------------------------- shortProblems.m start -------------------------------------------
%scalar variables

%basic sclar variables
a = 10
b = 2.5 * 10.^25
c = 4 + 1i*3        % i means complex number i
d = exp(1i*2*pi/3)  % pi is the famous number pi, 3.14...  etc.

%vector variables

aVec = [ 3.14 16 19 26 ]        %an array
bVec = [2.71; 8;  28; 182]      %a vector. rows are separeted by semicolon ;
cVec = 5:-.2:-5                 %from  5 to -5, decreasing 2 at each step
dVec = logspace(0,1,101)        %nice little built-in function ; array of 101
elements 1 to 10 spaced logarithmicly
eVec = 'Hello'              % a string

%matrix variables
aMat = ones(9)*2                    %ones returns a matrix full of 1's
bMat = diag([1 2 3 4 5 4 3 2 1])    %did not need zeros
cMat = reshape(1:100,[10,10])       %re shape does what the name implies
dMat = NaN(3,4)                     %matrix of NaN's
eMat = [13  -1 5 ;  -22  10 -87]    %2 to 3 custom matrix
fMat = floor(7*(rand(5,3)))-3 %randi(6,5,3)-3 would make more sense

%4. Scalar equations
x = 1/(1+ exp(-(a-15)/6))           %exp means e over (...)
y = (sqrt(a) + b^(1/21))^pi         % square root
z = log(real((c+d)*(c-d))*sin(a*pi/3))./(c*conj(c)) %log -> natural log , real ->
real part of a complex number
                                        %conj -> conjugate of a
                                        %complex number

%5. Vector equations
xVec = (1/(2*pi*2.5^2)^(1/2))*exp(-cVec.^2./(2*2.5^2))  %
yVec = ((aVec.').^2 + bVec.^2).^(1/2)                   %.' takes transpose
 zVec = log10(1./dVec)

%6 matrix
xMat = aVec*bVec*aMat^2
yMat = bVec*aVec
zMat = det(cMat)*((aMat.*bMat).')       % det takes determinat

%7Common functions
cSum = sum(cMat)                        %sum squizes matrix to an array by adding
up all rows
eMean = mean(eMat,2)                    %means takes means of all rows

eMat = [1,1,1;eMat(2:end,:)]            %replace first row of eMat by
concatenating
                              %one row with rest of the eMat(exculuding first row)
```

```
cSub = cMat(2:9,2:9)                    %take submatrix
lin = 1:20                       %then make every other value in it negative
lin(:,2:2:20) = -(lin(:,2:2:20))

r = rand(1,5)             %random numbers
r(find(r<0.5)) = 0 %here find is unnecessary r(r<0.5)=0 would do the trick
```

% ---------------------------- shortProblems.m end ---

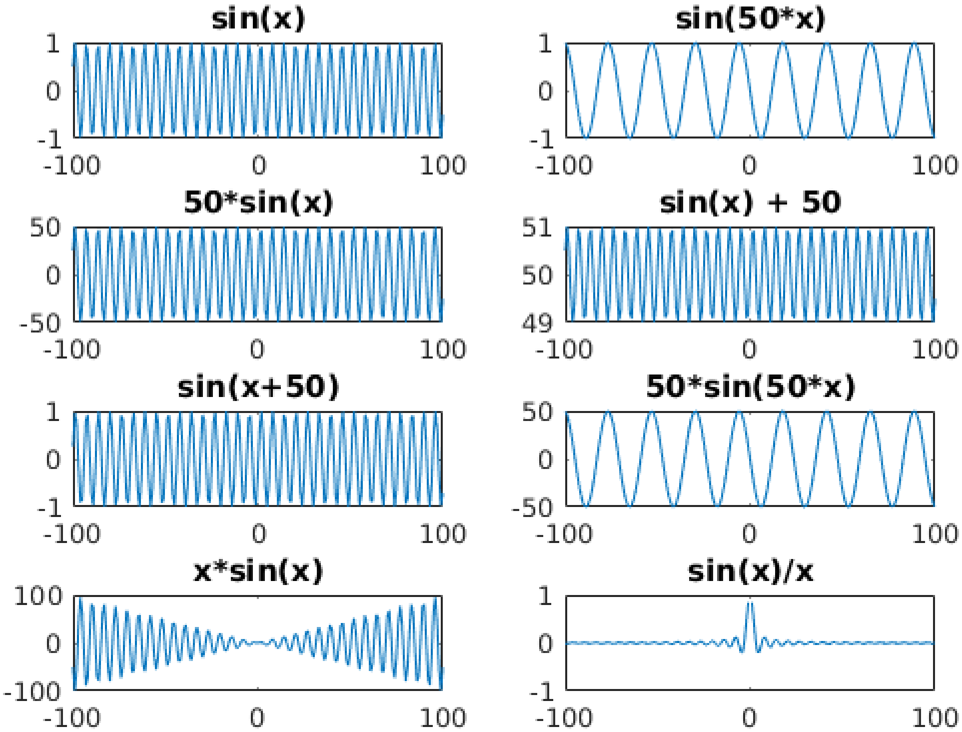
% ---------------------------- twoLinePlot.m start --

```
  t = linspace(0,2*pi,100); %intialize time between 0 and 2pi, 100 pieces

  figure; %instantiate a figure

  plot(t,sin(t)); %plot t sin(t) vs t
  hold on;        %hold it at hand

  plot(t,cos(t),'--');     %plot cos(t) vs t with desired assets
  xlabel('xlabel');        %label x axis
  ylabel('ylabel');        %label y axis
  title('title');          %title
  legend('Sin','Cos')      %the legend

  xlim([0 2*pi]);          %adjsut x and y axis's so that it look nicer
  ylim([-1.4 1.4]);
```



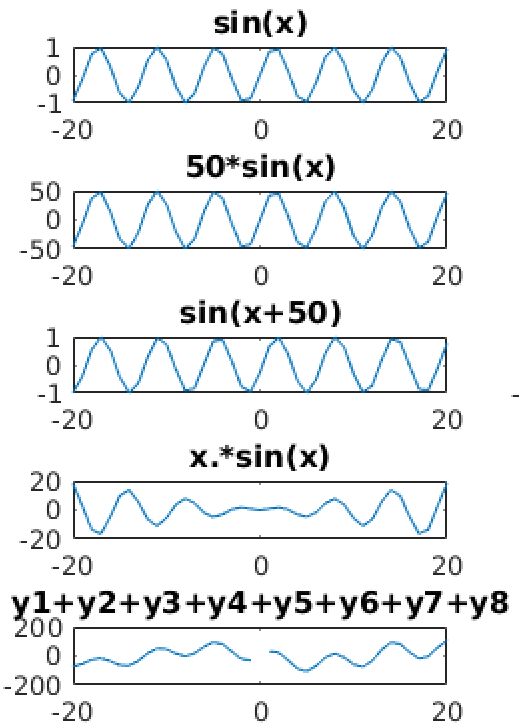
```
% ---------------------------- twoLinePlot.m end -------------------------------------------


% ---------------------------- calculateGrades.m start -------------------------------------

load('classGrades'); %load the .mat file

namesAndGrades(1:5,:)  %print a sample portion two verify whats inside

grades = namesAndGrades(:,2:end);  %chop off first column
meanGrades = mean(grades)           %take means of grades of each
assignment (columns)
meanGrades = nanmean(grades)        % since NaNs cause problem use
nanmean instead
meanMatrix = ones(15,1)*meanGrades
curvedGrades = (grades*3.5)./meanMatrix;

%i would do something like  this instead --> curvedGrades =
(grades(:,:)*3.5)./meanGrades(1,:)

nanmean(curvedGrades)           % since NaNs cause problem use nanmean
instead

curvedGrades(find(curvedGrades>5)) = 5; %find is not necessary here. make
all entries bigger than 5 , 5

totalGrade = nanmean(curvedGrades.').'; %trnspose, nanmean, transpose

ceiledtotalGrade =ceil(totalGrade) ;    %ceil the doubles

letters = ['F' 'D' 'C' 'B' 'A'];     %the array of letters

letterGrades = letters(ceiledtotalGrade);   %nice functional programming trick

disp(['Grades: ',letterGrades]) %display



% ---------------------------- calculateGrades.m end ----------------------------------------



% ---------------------------- signalAndNoise.m start ----------------------------------------

%10 4x2 subplot
x = (-100:100);  %-100 to 1000 201 numbers
%x = linspace(-100,100,20) would be neater
y1 = sin(x);
y2=sin(50*x);
y3=50*sin(x);
y4= sin(x) + 50;
```

```
y5= sin(x+50);
y6= 50*sin(50*x);
y7= x.*sin(x);
y8=sin(x)./x;

figure
subplot(4,2,1);  %first elemnet of 4 to 2 grid (upper left)
plot(x,y1)
title('sin(x)');          %title

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


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

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

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

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

subplot(4,2,7);
plot(x,y7)
title('x*sin(x)');

subplot(4,2,8);      %last element of 4 to 2 grid (lower right)
plot(x,y8)
title('sin(x)/x');
```




```
%11 5x2 subplot
x = (-20:20);
y1 = sin(x);
y2=sin(50*x);
y3=50*sin(x);
y4= sin(x+50);
y5= sin(x+50);
y6= 50*sin(50*x);
y7= x.*sin(x);
y8=sin(x)./x;

y9= y1+y2+y3+y4+y5+y6+y7+y8;

figure
subplot(5,2,1);
plot(x,y1)
title('sin(x)');

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

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

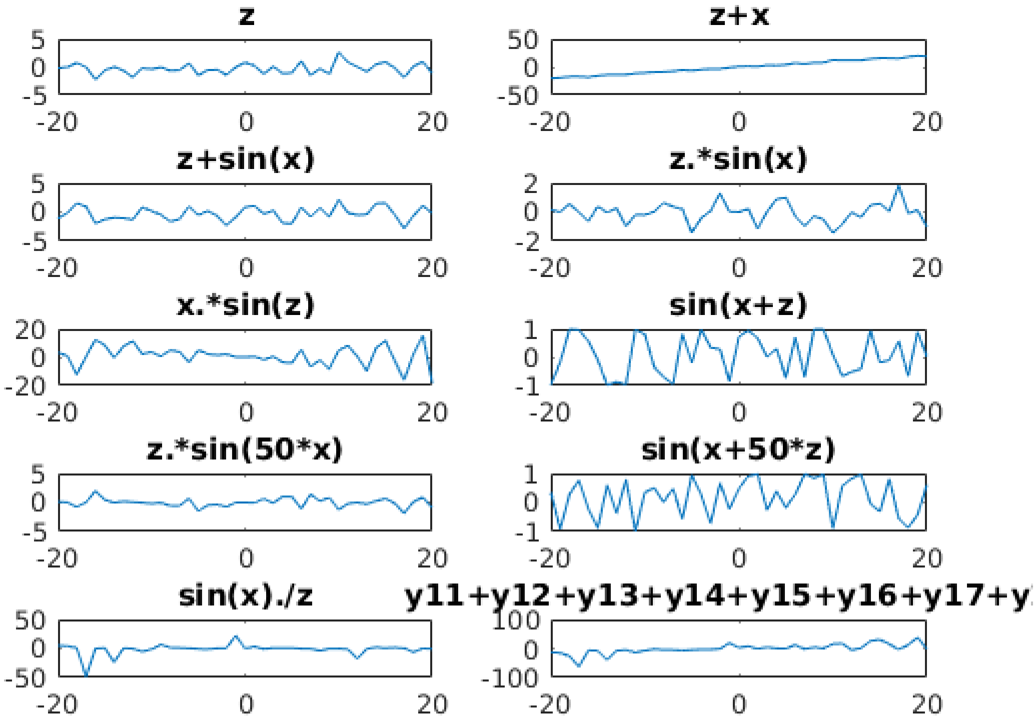
```
subplot(5,2,4);
plot(x,y4)
title('sin(x+50)');

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

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

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

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

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




```
%12
x = (-20:20);
z = randn(1,41); %array of 41 normal dist. random numbers each between 0
and 1
```

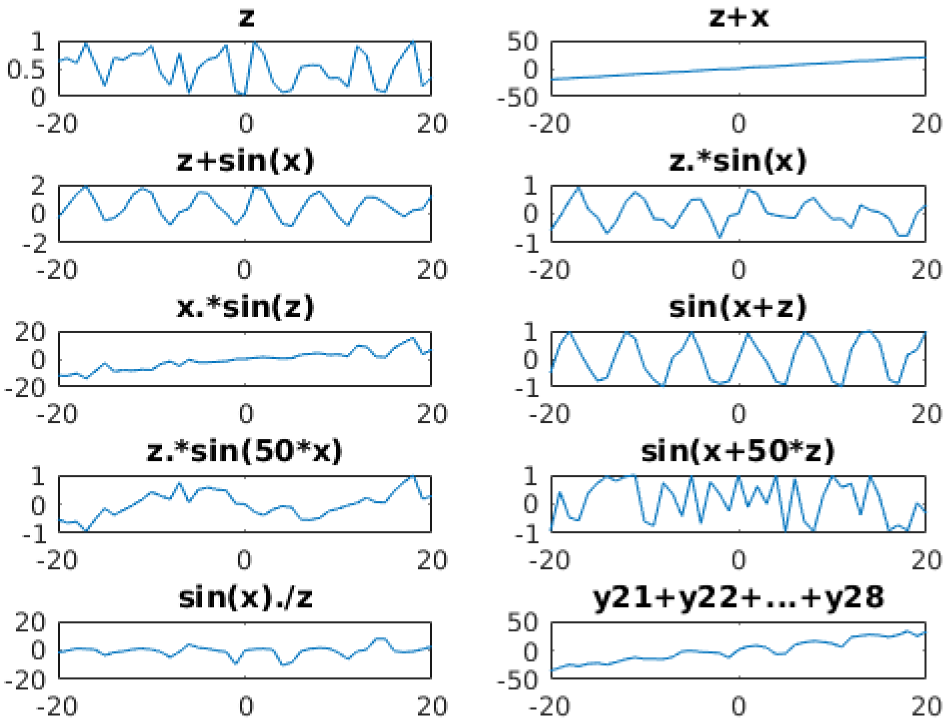
```
y10= z;
y11 = z+x;
y12= z+sin(x);
y13= z.*sin(x);
y14=x.*sin(z);
y15= sin(x+z),
y16= z.*sin(50*x);
y17=sin(x+50*z);
y18=sin(x)./z;
y19= y11+y12+y13+y14+y15+y16+y17+y18;


figure
subplot(5,2,1);
plot(x,y10)
title('z');

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

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


subplot(5,2,4);
plot(x,y13)
title('z.*sin(x)');

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

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

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

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

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

subplot(5,2,10);
```

```
plot(x,y19)
title('y11+y12+y13+y14+y15+y16+y17+y18');
```

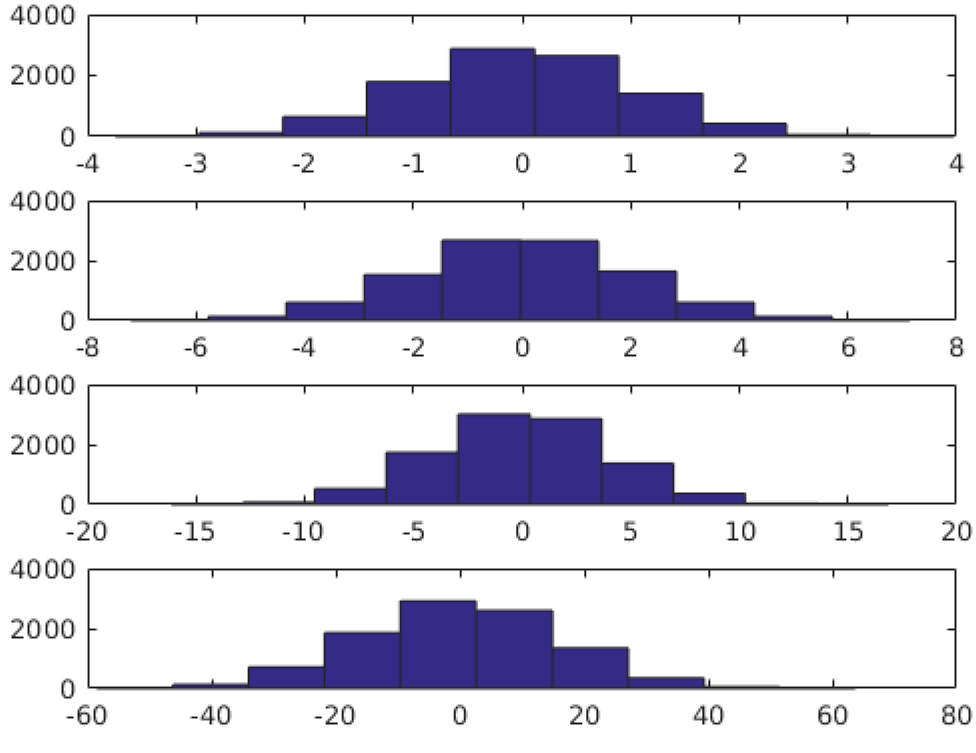



```
%13

z = rand(1,41); %array of 41 uniform dist. random numbers each between 0
and 1

y20= z;
y21 = z+x;
y22= z+sin(x);
y23= z.*sin(x);
y24=x.*sin(z);
y25= sin(x+z),
y26= z.*sin(50*x);
y27=sin(x+50*z);
y28=sin(x)./z;
y29= y21+y22+y23+y24+y25+y26+y27+y28;


figure
subplot(5,2,1);
plot(x,y20)
title('z');

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

subplot(5,2,3);
plot(x,y22)
```

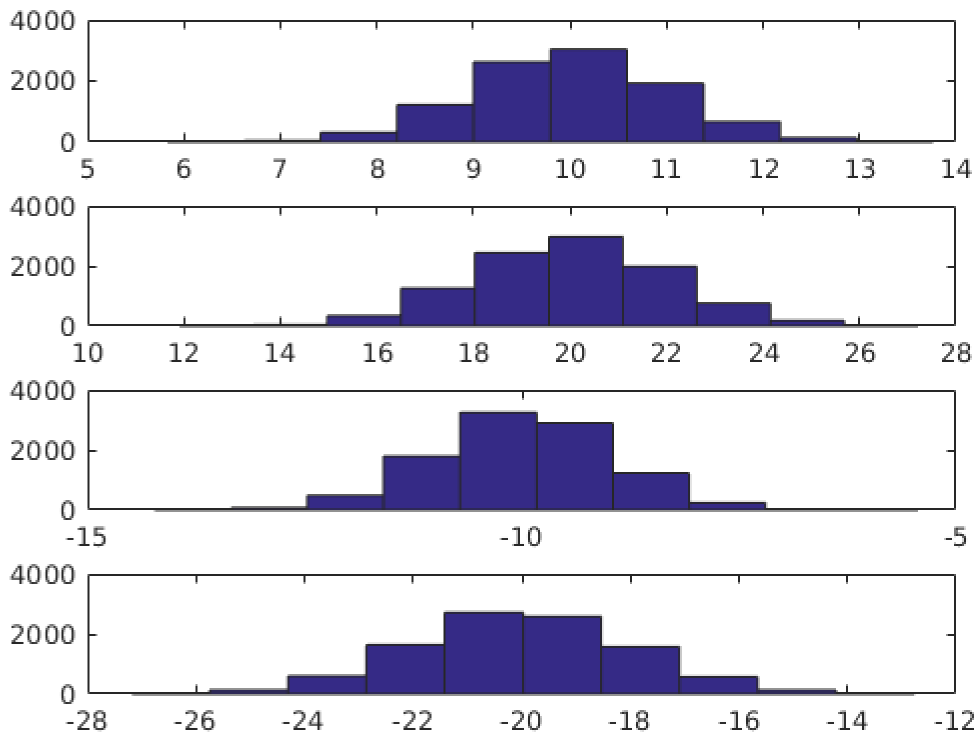
```
title('z+sin(x)');

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

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

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

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

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

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

subplot(5,2,10);
plot(x,y29)
title('y21+y22+...+y28');
```



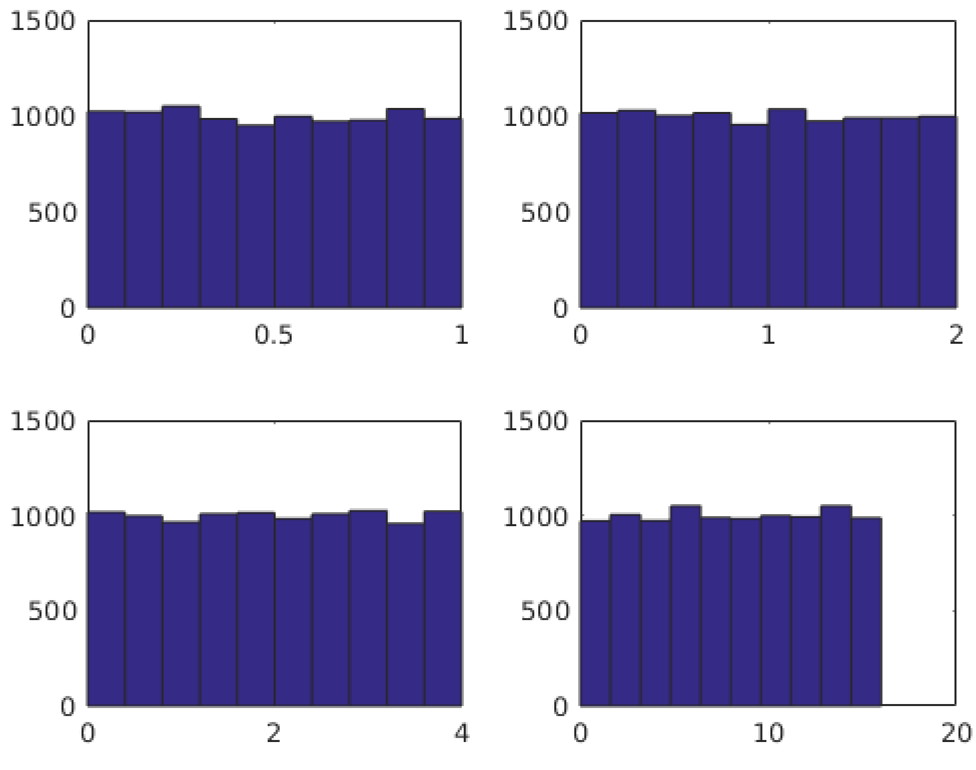
```
%14
mean = 0;
std = 1;

% all vectors of 10000 normal dist. random numbers
%with all mean 0

r1 = std.*randn(10000,1) + mean;    %with standard dev. of 1

std = 2;
r2 = std.*randn(10000,1) + mean;    %with standard dev. of 2

std=4;
r3 = std.*randn(10000,1) + mean;    %with standard dev. of 4

std=16;
r4 = std.*randn(10000,1) + mean;    %with standard dev. of 16



figure                    %init. figure

ax1 = subplot(4,1,1);              %quite similar to plot
hist(ax1,r1)

ax2 = subplot(4,1,2);
hist(ax2,r2)

ax3 = subplot(4,1,3);
hist(ax3,r3)

ax4 = subplot(4,1,4);
hist(ax4,r4)
```



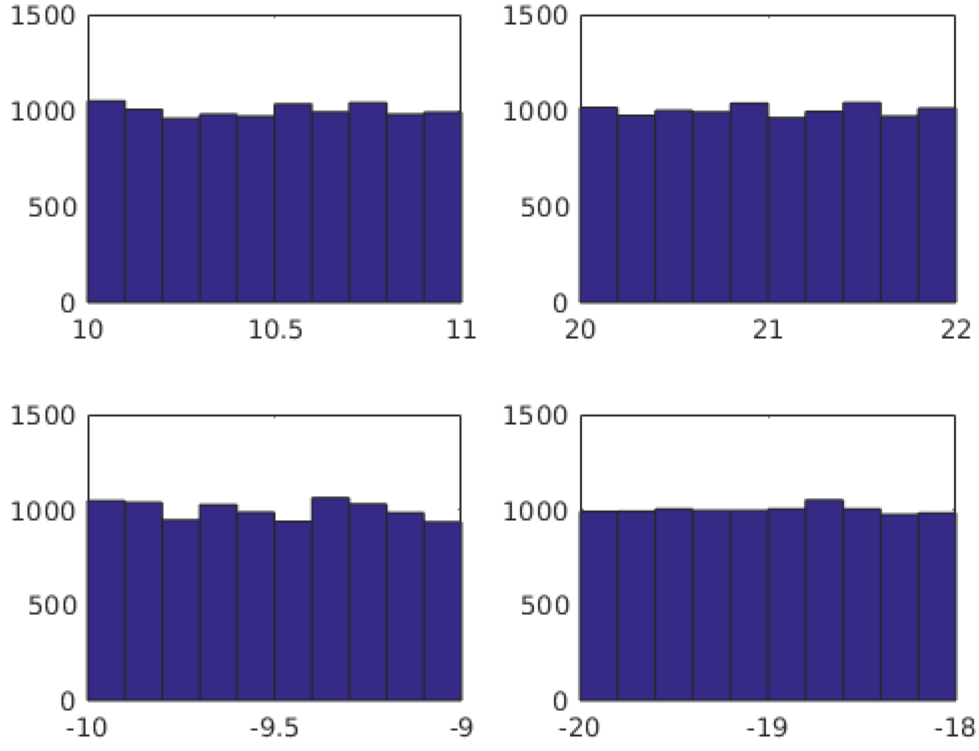
```
%15
mean = 10;
std = 1;
r6 = std.*randn(10000,1) + mean;

mean = 20;
std = 2;
r7 = std.*randn(10000,1) + mean;

mean = -10;
std=1;
r8= std.*randn(10000,1) + mean;

mean = -20;
std=2;
r9 = std.*randn(10000,1) + mean;

figure

ax6 = subplot(4,1,1);
hist(ax6,r6)

ax7 = subplot(4,1,2);
hist(ax7,r7)

ax8 = subplot(4,1,3);
hist(ax8,r8)

ax9 = subplot(4,1,4);
hist(ax9,r9)
```



```
%16

mean = 0;
variance = 1;

r11 = sqrt(variance).*rand(10000,1) + mean;

variance = 4;
r21 = sqrt(variance).*rand(10000,1) + mean;

variance=16;
r31 = sqrt(variance).*rand(10000,1) + mean;

variance=256;
r41 = sqrt(variance).*rand(10000,1) + mean;

figure

ax1 = subplot(2,2,1);
hist(ax1,r11)

ax2 = subplot(2,2,2);
hist(ax2,r21)

ax3 = subplot(2,2,3);
hist(ax3,r31)

ax4 = subplot(2,2,4);
hist(ax4,r41)
```



```
%17

mean = 10;
variance = 1;
r61 = sqrt(variance).*rand(10000,1) + mean;

mean = 20;
variance = 4;
r71 = sqrt(variance).*rand(10000,1) + mean;

mean = -10;
variance=1;
r81= sqrt(variance).*rand(10000,1) + mean;

mean = -20;
variance=4;
r91 = sqrt(variance).*rand(10000,1) + mean;

figure

ax6 = subplot(2,2,1);
hist(ax6,r61)

ax7 = subplot(2,2,2);
hist(ax7,r71)

ax8 = subplot(2,2,3);
hist(ax8,r81)

ax9 = subplot(2,2,4);
hist(ax9,r91)
```



```
% –---------------------------- .signalAndNoise.m end -------------------------------------------


%18
%i have learned how to simulate sinx cosx xsinx etc functions
%also different distributions with desired means and standart
%deviations
%i have learned how to use plots,
% how to put various plots on the same figure
%i have learned hist(), i also learned that it should not be used
% and histogram should be used instead.



%19

% -
%
%
% - matlab is much more efficient in terms of both performance
%   and ease of coding when it comes to intensive math operations
%   and plotting, graphing etc.
%   also working with equations is easy, possibly signal manipulations are too
%Note : I began to like functional programming which I was not quite fond of.
```