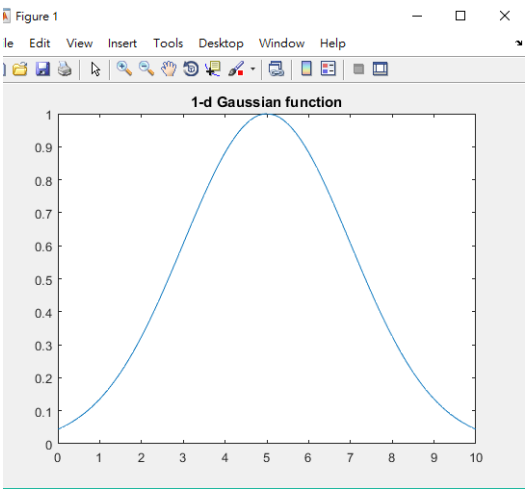
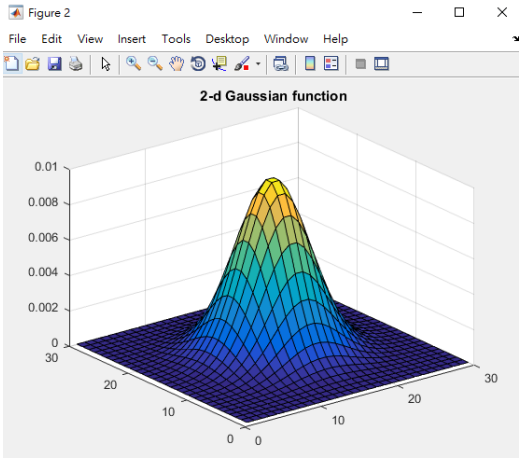
%1

```
x = -2:0.1:8;
y = gaussmf(x,[2 4]);
figure(1);
plot(x,y);
title('1-d Gaussian function');
```



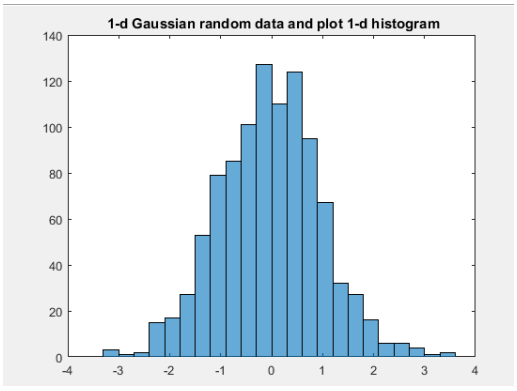
%2

```
h = fspecial('gaussian', [30 30], 4);
figure(2);
surf(h);
title(' 2-d Gaussian function');
```



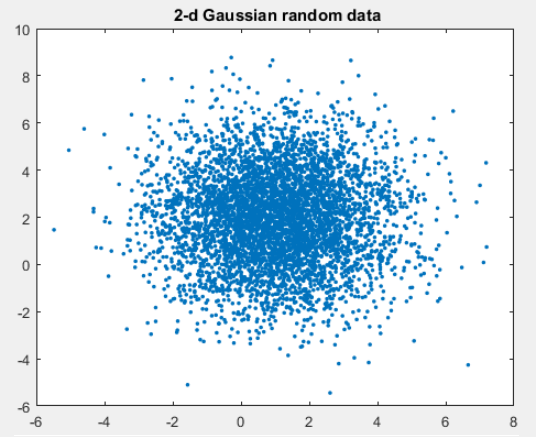
%3

```
rng default; % For reproducibility
r = normrnd(0,1,1000,1);
figure(3);
histogram(r);
title(' 1-d Gaussian random data and plot 1-d histogram');
```



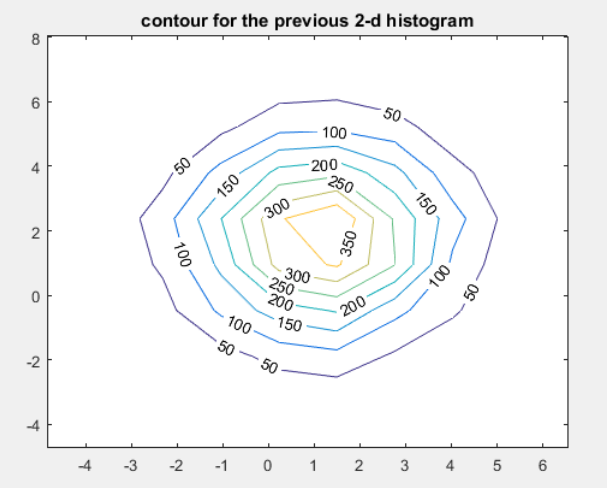
%4

```
mu = [1 2];
sigma = [3 0; 0 4];
rng default    % For reproducibility
R = mvnrnd(mu,sigma,5000);
figure(4);
plot(R(:,1),R(:,2),'.','MarkerSize',8);
title(' 2-d Gaussian random data');
figure (5);
hist3(R);
title('    2-d histogram for 2-d Gaussian random data');
```



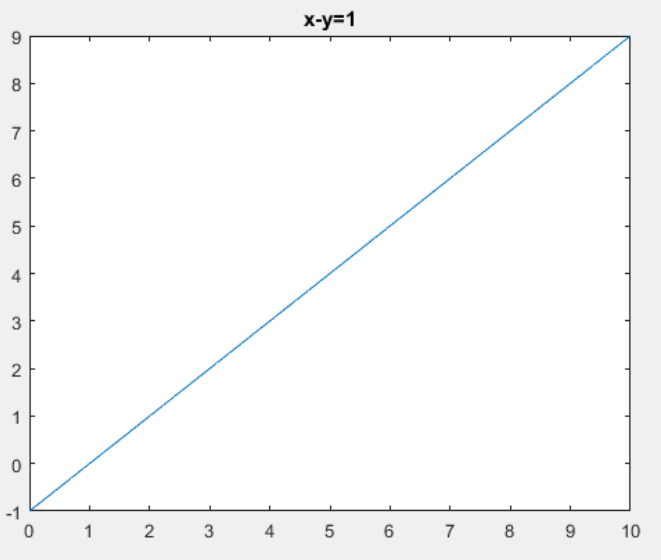
% 5

```
figure(6);
  [n,c] = hist3(R, [10 10]);
  contour(c{1}, c{2}, n);
  [a,handle] = contour(c{1}, c{2}, n);
  clabel(a, handle);
title(' contour for the previous 2-d histogram ');
```



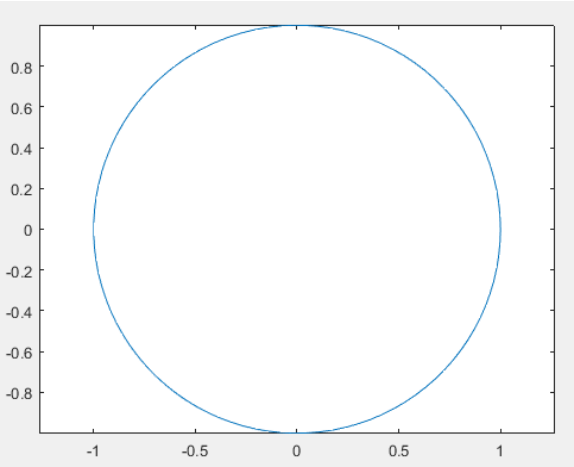
% 6

```
x = linspace(0, 10);
y = x - 1;
figure(7);
plot(x, y);
title('x-y=1');
```



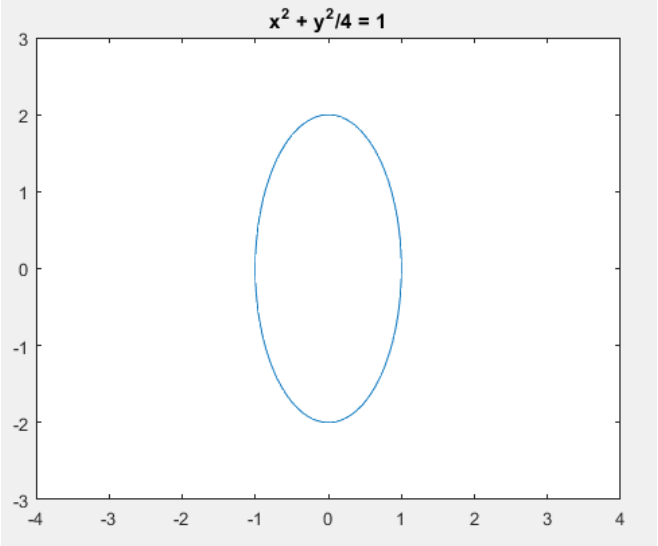
% 7

```
figure(8);
title('x^2 + y^2 =1');
r = 1;
xc = 0;
yc = 0;
theta = linspace(0,2*pi);
x = r*cos(theta) + xc;
y = r*sin(theta) + yc;
plot(x,y);
axis equal;
```



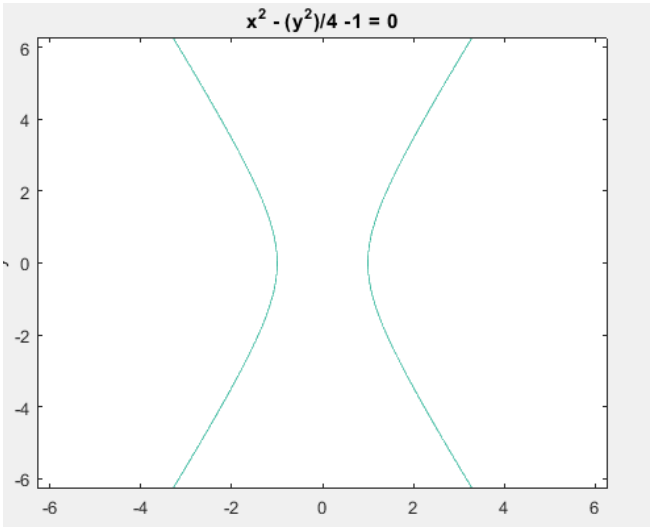
% 8

```
a=1; % horizontal radius
b=2; % vertical radius
x0=0; % x0,y0 ellipse centre coordinates
y0=0;
t=linspace(0,2*pi);
x=x0+a*cos(t);
y=y0+b*sin(t);
figure(9);
plot(x,y)
axis([-4 4 -3 3]);
title('x^2 + y^2/4 = 1');
```



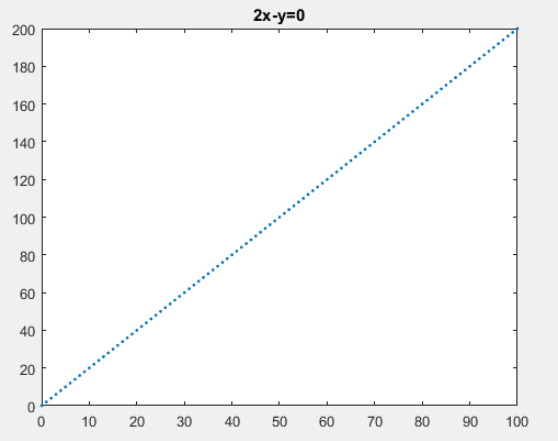
% 9

```
figure(10);
x = [-5:0.1:-1];
x = [x 1:0.1:5];
y1 = sqrt(x.^2-1) * 2;
y2 = - sqrt(x.^2-1) * 2;
plot(x,y1,'b',x,y2,'b');
ezplot('x.^2 - (y.^2)/4 -1');
```



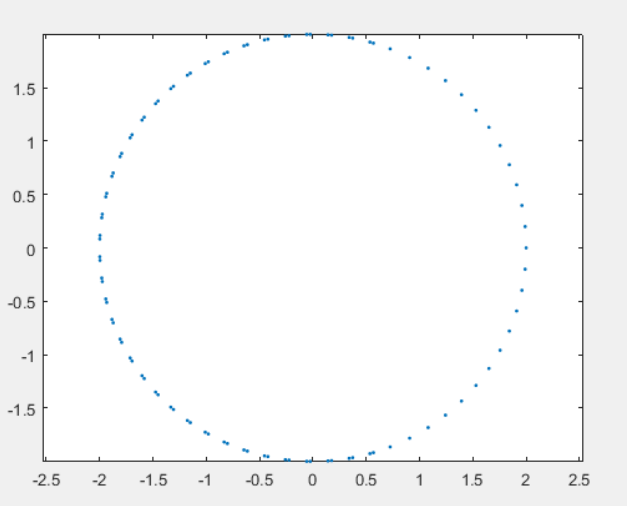
% 10

```
figure(11);
x = (0:1:100);
y = 2*x;
plot(x,y,'.');
title('2x-y=0');
```



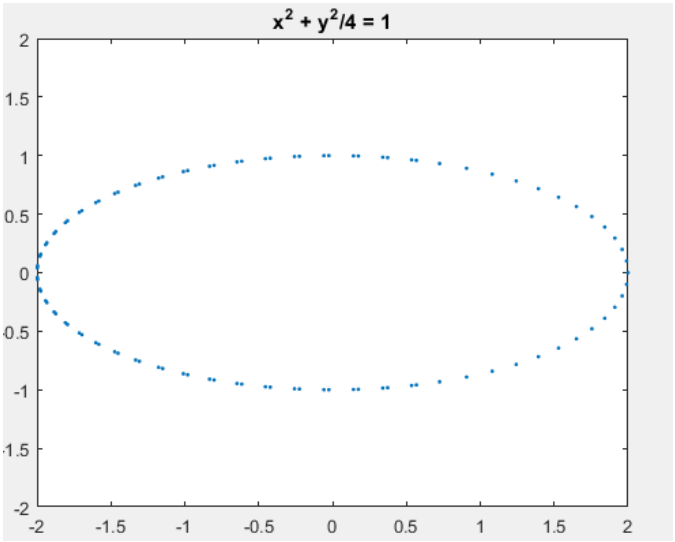
% 11

```
figure(12);
title('x^2 + y^2 =4');
r = 2;
xc = 0;
yc = 0;
theta = (-5:0.1:5);
x = r*cos(theta) + xc;
y = r*sin(theta) + yc;
plot(x,y,'.');
axis equal;
```



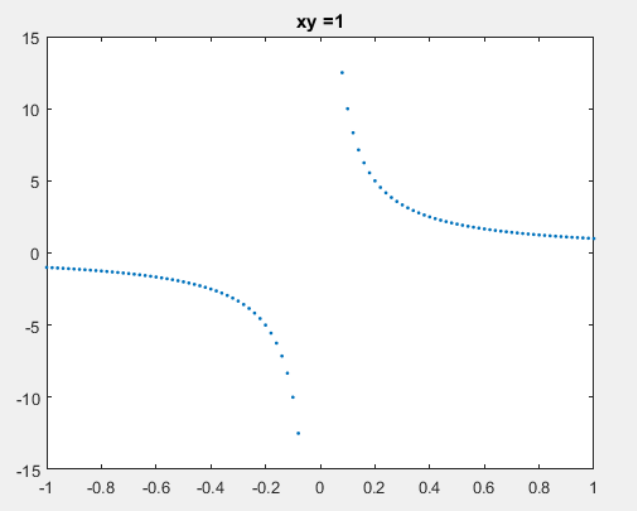
% 12

```
a=2; % horizontal radius
b=1; % vertical radius
x0=0; % x0,y0 ellipse centre coordinates
y0=0;
t=(-5:0.1:5);
x=x0+a*cos(t);
y=y0+b*sin(t);
figure(13);
plot(x,y,'.')
axis([-2 2 -2 2]);
title('x^2 + y^2/4 = 1');
```



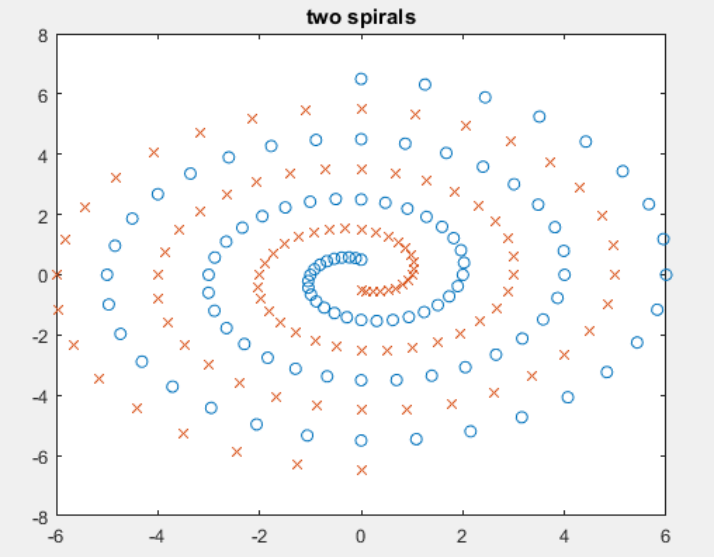
% 13

```
figure(14);
x = (-1:0.02:1);
y = 1./x;
plot(x,y,'.');
axis([-1 1 -15 15]);
title('xy =1');
```



% 14

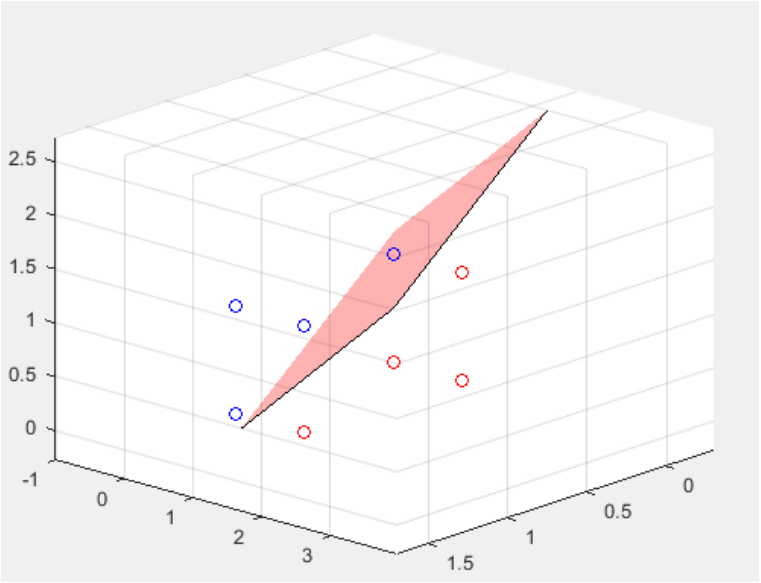
```
for i = 0 : 1 : 96
    r = 6.5 * (104 - i) / 104;
    theta = pi * i /16 ;
    temp1 = r * sin(theta);
    temp2 = r * cos(theta);
    temp3 = -r * sin(theta);
    temp4 = -r * cos(theta);
    if i == 0
        x1 = [temp1];
        y1 = [temp2];
        x2 = [temp3];
        y2 = [temp4];
    end
    x1 = [ x1 temp1];
    y1 = [ y1 temp2];
    x2 = [ x2 temp3];
    y2 = [ y2 temp4];

end
figure(15);
plot(x1,y1,'o',x2,y2,'x');
title('two spirals');
```



% 15

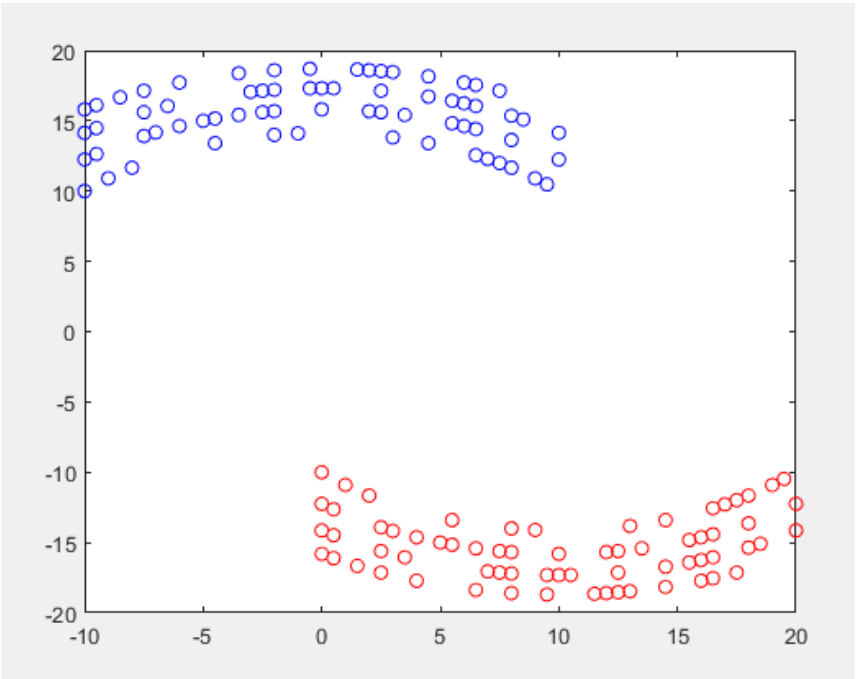
```
x1 = [0 0 1 0];
y1 = [1 0 1 1];
z1 = [1 1 1 0];
x2 = [0 1 1 1];
```

```
  y2 = [0 0 0 1];
  z2 = [0 0 1 0];
  figure(16);
  scatter3(x1,y1,z1,'b');
  hold on;
  scatter3(x2,y2,z2,'r');
  hold on;
  pointA = [1,1,0.5];
pointB = [2.5,0,3];
pointC = [3.5,2,2];
  normal = cross(pointA-pointB, pointA-pointC);
x = [pointA(1) pointB(1) pointC(1)];
y = [pointA(2) pointB(2) pointC(2)];
z = [pointA(3) pointB(3) pointC(3)];
A = normal(1); B = normal(2); C = normal(3);
D = -dot(normal,pointA);
zLim = [min(z) max(z)];
yLim = [min(y) max(y)];
[Y,Z] = meshgrid(yLim,zLim);
X = (C * Z + B * Y + D)/ (-A);
reOrder = [1 2    4 3];
patch(X(reOrder),Y(reOrder),Z(reOrder),'r');
grid on;
alpha(0.3);
```



% 16

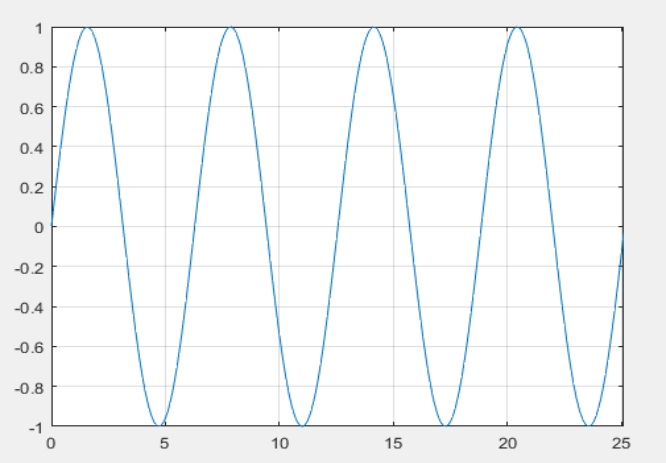
```
moon = 100;
r = randi([1 160],1,80);
x1 = [-10:0.5:10];
x2 = [ 0 : 0.5 : 20];
y1 = sqrt(100 + moon - x1.^2);
y2 = sqrt(150 + moon - x1.^2);
y3 = sqrt(200 + moon - x1.^2);
y4 = sqrt(250 + moon - x1.^2);
temp1 = [y1 y2 y3 y4];
y5 = -sqrt( 100 + moon - (x2 - 10).^2 );
y6 = -sqrt( 150 + moon - (x2 - 10).^2 );
y7 = -sqrt( 200 + moon - (x2 - 10).^2 );
y8 = -sqrt( 250 + moon - (x2 - 10).^2 );
temp2 = [y5 y6 y7 y8];
x1 = [x1 x1 x1 x1];
```

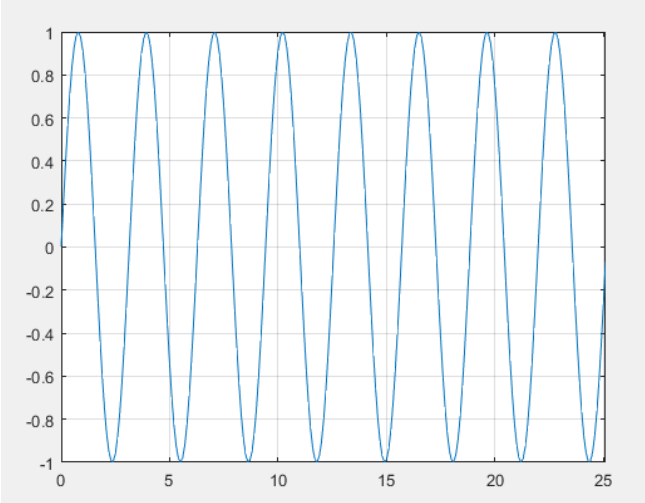


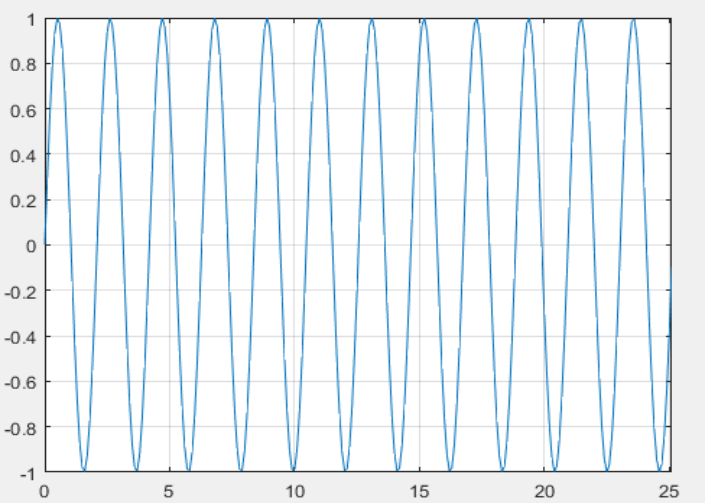
```
x2 = [x2 x2 x2 x2];
pic_x1 = x1(r);
pic_y1 = temp1(r);
pic_x2 = x2(r);
pic_y2 = temp2(r);
figure(17);
plot(pic_x1,pic_y1,'o','MarkerEdgeColor','b');
hold on;
plot(pic_x2,pic_y2,'o','MarkerEdgeColor','r');
hold on;
```

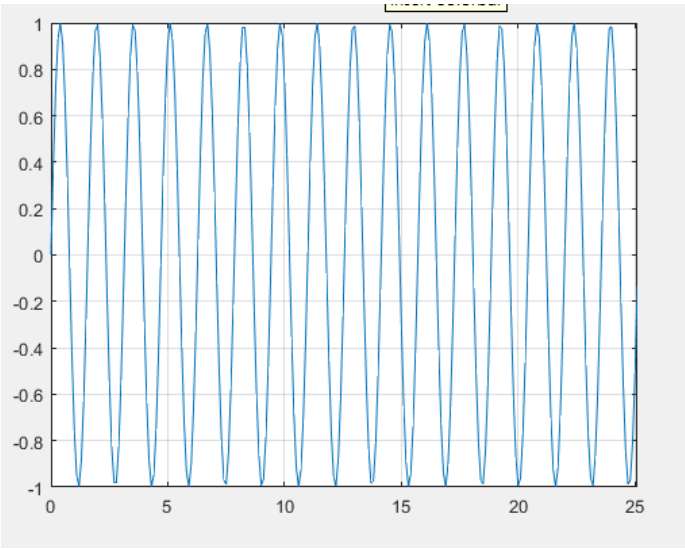
% 17

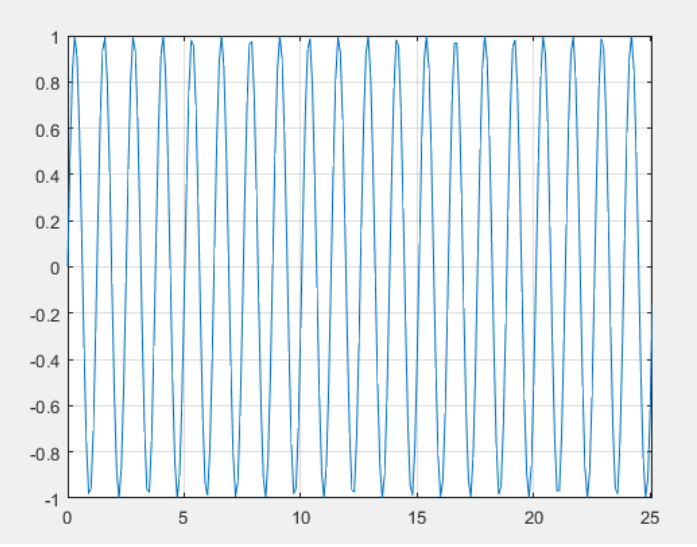
```
figure(18);
x = 0:0.1:8*pi;
for i = 1:250
    plot(x, sin(x*floor(i/50)));
    axis([-inf inf -1 1]);
    grid on
    drawnow
end
```











Q18

分析:

這題我使用了 linear congruential generator 這個方法，他是目前最被廣泛使用且歷史最悠久的一種算法。

基本上只有一個公式:

- $r_{n+1} = a \times r_n + c \pmod{m}$
- r_0 is a seed.
- $r_1, r_2, r_3, \ldots$, are the random numbers.
- a, c, m are constants.

而雖然我們從公式中可以看到，r 的下個值可以被前一個值預測，這個公式還是可以用於簡單的亂數產生(不包括需要高度安全的狀況)，而還有一個重點是 a, c ,m 應該選何值，怎麼選比較好，關於這點也有很多種作法，我選擇其中一個 microsoft formula 實作，最大的數會到 32767。而如果係數選得好的話，產生的亂數會接近 uniform 的理想結果。

程式:

```
a = 214013;
seed = 0;
c = 2531011;
m=2147483648;
for i = 1:100
    seed =mod( (a *seed + c) ,m);
    ans = bitshift(seed, -16);
    disp(ans./32767)
end
```

(a)部分結果-產生0~1的數字

0.5098
0.4657
0.4865
0.0838
0.8354
0.8811
0.5495
0.5088
0.0962
0.3607

(b)如果要設其他範圍，只要對 a 的結果處理即可，例如想要產生 1~4，就將 a 的結果乘以三再加 1，以此類推

參考:

http://edisonx.pixnet.net/blog/post/69653913-%5Brand%5D-linear-congruential-generator-(%E7%B7%9A%E6%80%A7%E5%90%8C%E9%A4%98%E6%B3%95,lcg)
https://zh.wikipedia.org/wiki/%E7%B7%9A%E6%80%A7%E5%90%8C%E9%A4%98%E6%96%B9%E6%B3%95
https://rosettacode.org/wiki/Linear_congruential_generator

Q19

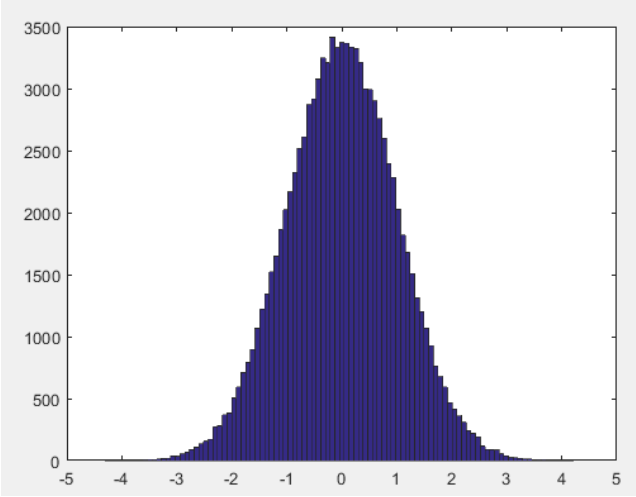
分析:

Using Box–Muller transform to generate Gaussian random number, Box-Muller sampling is based on representing the joint distribution of two independent standard Normal random Cartesian variables X and Y The joint distribution p(x,y) (which is circular-symmetric) is: $p(x,y) = p(x)p(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}} = \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}} = \left(\frac{1}{2\pi}\right)\left(e^{\frac{-r^2}{2}}\right)$ **which is the product of two density functions, an exponential distribution over squared radii** $r^2 \sim Exp(\frac{1}{2})$ **and a uniform distribution over angles** $\theta \sim Unif(0, 2\pi)$ **Then we make connection between the exponential distribution and the uniform distribution** $Exp(\lambda) = \frac{-\log(Unif(0,1))}{\lambda}$ **then** $r \sim \sqrt{-2\log(Unif(0,1))}$ **This a way to generate points from the joint Gaussian distribution by sampling from two independent uniform distributions, one for r and another for theta, and transforming them into Cartesian coordinates. And the steps are 1. Draw** $u_1, u_2 \sim Unif(0,1)$ **2. Transform the variables into radius and angle representation** $r = \sqrt{-2\log(u_1)}, \text{and } \theta = 2\pi u_2$ **3. Transform radius and angle into Cartesian coordinates:** $x = r\cos(\theta), y = r\sin(\theta)$ **The x and y are the result of the Gaussian random number.**

程式:

```
u = rand(2,100000);
r = sqrt(-2*log(u(1,:)));
theta = 2*pi*u(2,:);
x = r.*cos(theta);
y = r.*sin(theta);
figure(19);
hist(x,100);
```

結果:



參考:

https://theclevermachine.wordpress.com/2012/09/11/sampling-from-the-normal-distribution-using-the-box-muller-transform/

https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

https://www.alanzucconi.com/2015/09/16/how-to-sample-from-a-gaussian-distribution/

Q20

```
filename = 't10k-images.idx3-ubyte';
fp = fopen(filename,'r');
magic = fread(fp,1, 'int32',0, 'ieee-be');
num_image = fread(fp,1, 'int32',0, 'ieee-be');
num_row = fread(fp,1, 'int32',0, 'ieee-be');
num_col = fread(fp,1, 'int32',0, 'ieee-be');
image = fread(fp, inf, 'unsigned char');
image = reshape(image, num_col, num_row, num_image);
image = permute(image,[2 1 3]);
fclose(fp);
image = reshape(image, size(image, 1) * size(image, 2), size(image, 3));
image = double(image) / 255;
for tmp = 1:150
    if mod(tmp,15) == 0
        img( (ceil(tmp/15) *28 - 27) : ceil(tmp/15) *28, 393:420) = reshape(image(:,tmp),28,28);
    else
        img( (ceil(tmp/15) *28 - 27) : ceil(tmp/15) *28,    (mod(tmp,15)*28-27) : mod(tmp,15)*28) =
reshape(image(:,tmp),28,28);
    end
end
imshow(img);
```