CVIP Laboratory

UNIVERSITY OF LOUISVILLE

Hands-on Generating Random Variables

Shireen Elhabian

Aly Farag

October 2007

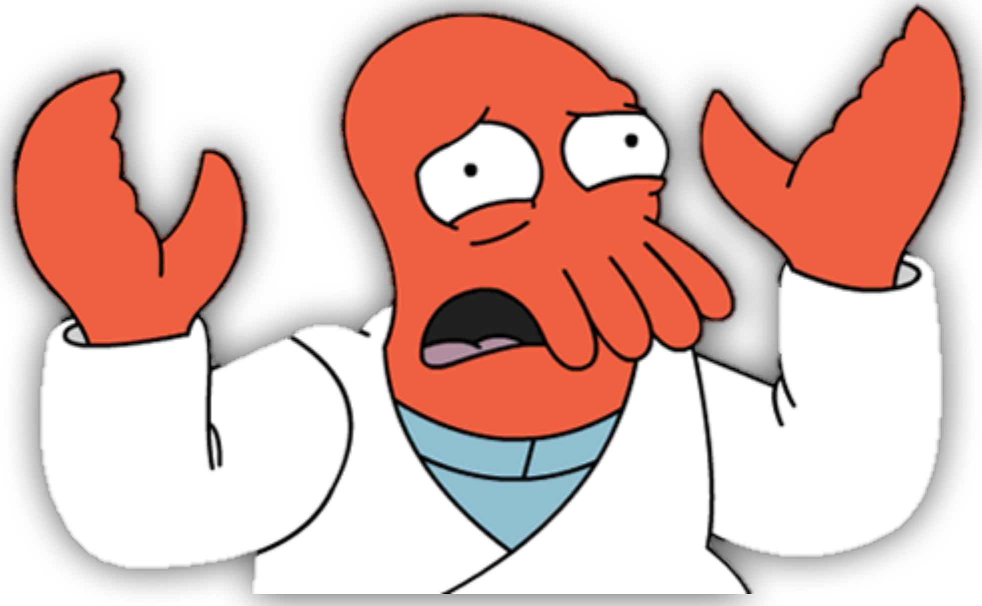
Generate random variables from known probability distributions.

WHAT

WHY ?!

- The heart of Monte Carlo simulation for statistical inference.
- Generate synthetic data to test our algorithms, such as data fitting and classification.
- Generating data encryption keys.
- Simulating and modeling complex phenomena.
- Selecting random samples from larger data sets.

HOW ?!



We will learn algorithms and get them into action !!!

Agenda

- $U(0,1)$
- $N(0,1)$
- $N(\mu,\sigma)$
- $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$

Random

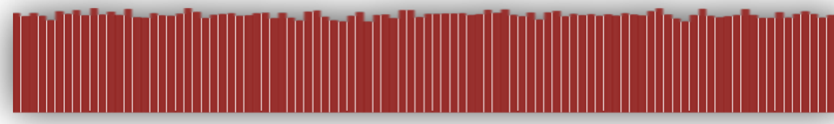
- Truly Random
  - Exhibiting true randomness
- Pseudorandom
  - Appearance of randomness but having a specific repeatable pattern
- Quasi-random
  - Having a set of non-random numbers in a randomized order

Generating U(0,1) Random Variables

- They are usually the building block for generating other random variables.
- We will look at:
  - Properties that a random number generator should possess
  - Linear Congruential Generators (LCGs)
  - Use Matlab to generate U(0,1) variates.

Properties of a U(0,1) Generator

- Numbers should appear to be ~ U(0,1) and independent.



- Generator should be fast and not require too much storage.
- Should be able to reproduce a iven set of numbers for comparison purposes.

Linear Congruential Generators (LCGs)

- Numbers are generated according to:

$$Z_i = (aZ_{i-1} + c) \bmod m$$

  - where m, a, c and Z_0 are non-negative numbers.
  - Z_0 is the seed.
  - m is the modulus.

- Z_i is a sequence of integer values ranging from 0 to $m - 1$, starting from the seed point Z_0.

- To generate pseudo-random numbers, $U_1, \dots, U_n$, we set $U_i = Z_i/m$. Thus $U_i \in (0,1)\ \forall i$.

Example 1

- $Z_0 = 1, m = 16, a = 11, c = 0$
$$Z_i = (11\, Z_{i-1}) \bmod 16$$

- Now iterate to determine the $Z_i{}'$s
$$Z_0 = 1$$
$$Z_1 = (11) \bmod 16 = 11$$
$$Z_2 = (121) \bmod 16 = 9$$
$$Z_3 = (99) \bmod 16 = 3$$
$$Z_4 = (33) \bmod 16 = 1$$
$$\ldots$$

- What is wrong with this?
  - The Z_i's are not that random.
  - They can only take on a finite number of values.
  - The period of the generator can be very poor.

How to Guarantee a Full Period?!!

Theorem

The linear congruential generator has full period if and only if the following three conditions holds:

1. If 4 divides m, then 4 divides $a - 1$
2. The only positive integer that exactly divides both m and c is 1, i.e. m and c are relatively prime, such that $\gcd(m, c) = 1$
3. If q is a prime number that divides m, then it divides $a - 1$.

Example 2

- $Z_0 = 1, m = 16, a = 13, c = 13$

$$Z_i = (13\, Z_{i-1} + 13) \bmod 16$$

- Now iterate to determine the Z_i 's

$$Z_0 = 1$$
$$Z_1 = (26) \bmod 16 = 10$$
$$Z_2 = (143) \bmod 16 = 15$$
$$Z_3 = (248) \bmod 16 = 0$$
$$Z_4 = (13) \bmod 16 = 13$$
$$\ldots$$

- Check to see that this LCG has full period:
  - Are the conditions of the theorem satisfied?
  - They can only take on a finite number of values.
  - Does it matter what integer we use for Z_0?

The linear congruential generator has full period if and only if the following three conditions holds:

1. If 4 divides m, then 4 divides $a - 1$
2. The only positive integer that exactly divides both m and c is 1, i.e. m and c are relatively prime, such that $\gcd(m, c) = 1$
3. If q is a prime number that divides m, then it divides $a - 1$.

Seeds

- In your experimentation, you can generate 100 streams each with a different seed point in order to:
  - avoid duplicate streams of random numbers,
  - get a general idea of the behavior of the random number generator on your workstation.
- Seeds can be obtained from:
  - A. M. Law and W. D. Kelton, *Simulation Modeling & Analysis,* 2nd Edition, McGraw-Hill, NewYork, 1991.

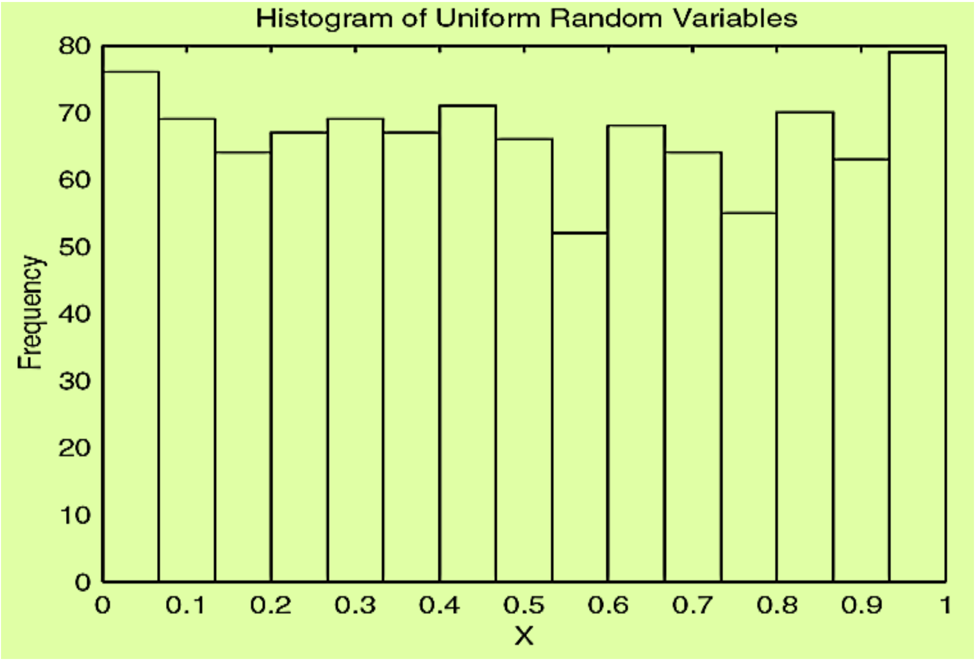
```
Seeds = [ 1, ...
 1973272912, 281629770,  20006270,1280689831,2096730329,1933576050,...
  913566091, 246780520,1363774876, 604901985,1511192140,1259851944,...
  824064364, 150493284, 242708531,  75253171,1964472944,1202299975,...
  233217322,1911216000, 726370533, 403498145, 993232223,1103205531,...
  762430696,1922803170,1385516923,  76271663, 413682397, 726466604,...
  336157058,1432650381,1120463904, 595778810, 877722890,1046574445,...
   68911991,2088367019, 748545416, 622401386,2122378830, 640690903,...
 1774806513,2132545692,2079249579,  78130110, 852776735,1187867272,...
 1351423507,1645973084,1997049139, 922510944,2045512870, 898585771,...
  243649545,1004818771, 773686062, 403188473, 372279877,1901633463,...
  498067494,2087759558, 493157915, 597104727,1530940798,1814496276,...
  536444882,1663153658, 855503735,  67784357,1432404475, 619691088,...
  119025595, 880802310, 176192644,1116780070, 277854671,1366580350,...
 1142483975,2026948561,1053920743, 786262391,1792203830,1494667770,...
 1923011392,1433700034,1244184613,1147297105, 539712780,1545929719,...
  190641742,1645390429, 264907697, 620389253,1502074852, 927711160,...
  364849192,2049576050, 638580085, 547070247 ];
```

In Matlab ☺

```
% Obtain a vector of uniform random variables in (0,1).
x = rand(1,1000);
% Do a histogram to plot.
% First get the height of the bars.
[N,X] = hist(x,15);
% Use the bar function to plot.
bar(X,N,1,'w')
title('Histogram of Uniform Random Variables')
xlabel('X')
ylabel('Frequency')
```

Notes:

- The function *rand* with no arguments returns a single instance of the random variable *U*.
- To get an array *mxn* of uniform variates, you can use the syntax **rand(m,n).**
- If you use **rand(n)**, then you get an *nxn* matrix.



In Matlab ☺

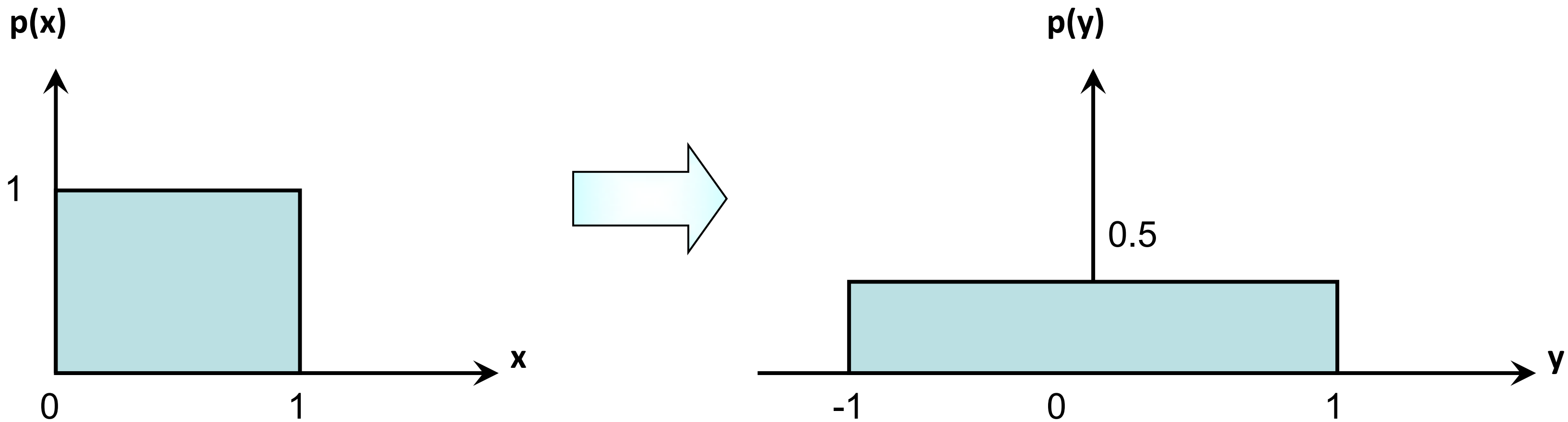
- The seed or the state of the generator is reset to the default when Matlab starts up, so the same sequencyes of random variables are generated whenever you start Matlab.
- If you call the function using **rand('state',0),** then MATLAB resets the generator to the initial state.
- If you want to specify another state, then use the syntax **rand('state',j)** to set the generator to the *j*-th state.
- You can obtain the current state using **S = rand('state'),** where **S** is a 35 element vector. To reset the state to this one, use **rand('state',S)**.

```
% Generate 3 random samples of size 5.
x = zeros(3,5);    % Allocate the memory.
for i = 1:3
   rand('state',i) % set the state
   x(i,:) = rand(1,5);
end
```

0.9528	0.7041	0.9539	0.5982	0.8407
0.8752	0.3179	0.2732	0.6765	0.0712
0.5162	0.2252	0.1837	0.2163	0.4272

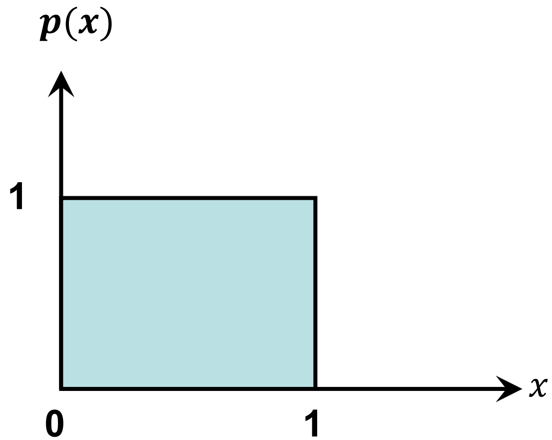
Inverse Transform Method

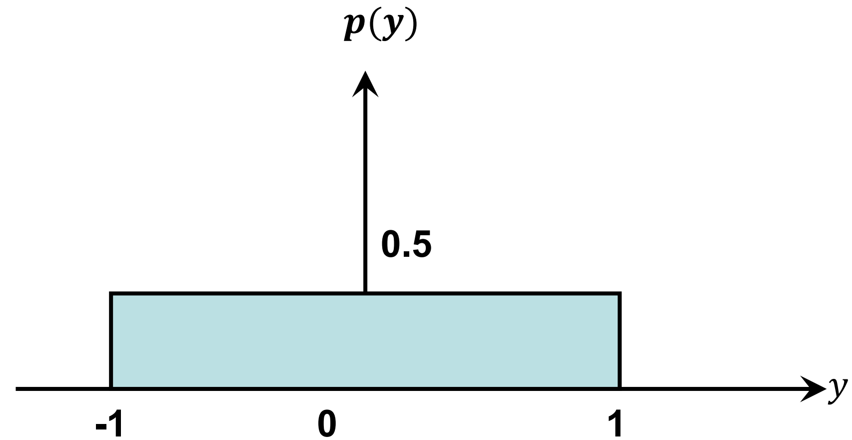
- This method converts a known distribution with known parameters to another distribution with different parameters.
- Example: Generate $Y \sim U(-1,1)$ from $X \sim U(0,1)$.



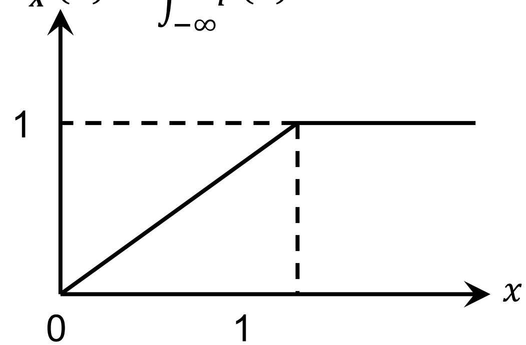
$$x \sim U(0,1) \longrightarrow \boxed{g(.)} \longrightarrow y \sim U(-1,1)$$

Inverse Transform Method

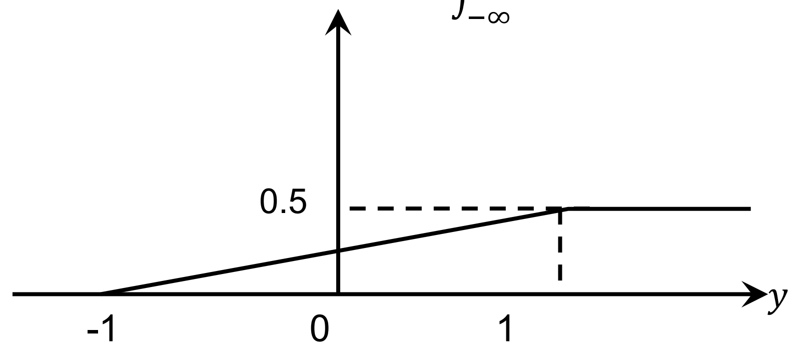




$$F_X(x) = \int_{-\infty}^{x} p(x)dx$$



$$F_Y(y) = \int_{-\infty}^{y} p(y)dy$$

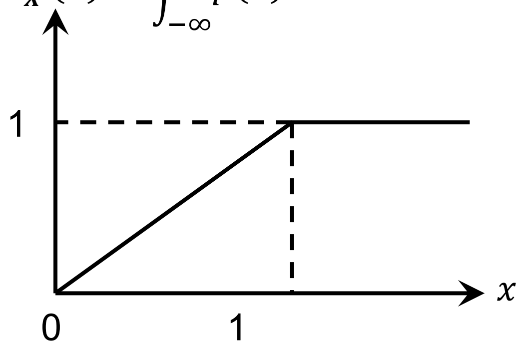


$$F_X(x) = \begin{cases} 0 & x \le 0 \\ x & 0 < x < 1 \\ 1 & x \ge 1 \end{cases}$$

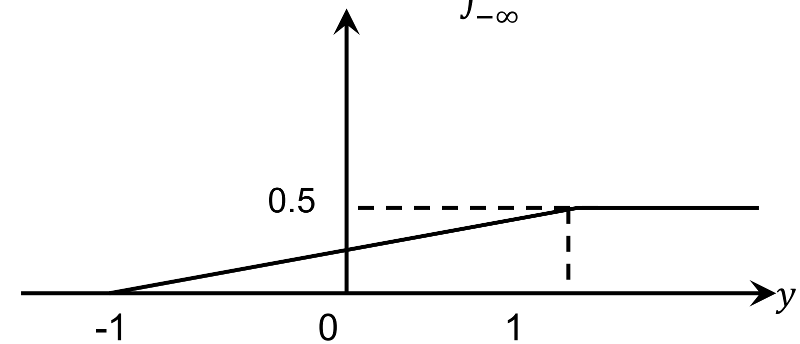
$$F_Y(y) = \begin{cases} 0 & y \le -1 \\ \dfrac{y+1}{2} & -1 < y < 1 \\ 1 & y \ge 1 \end{cases}$$

Inverse Transform Method

$F_X(x) = \int_{-\infty}^{x} p(x)dx$



$F_Y(y) = \int_{-\infty}^{y} p(y)dy$



$$F_X(x) = \begin{cases} 0 & x \le 0 \\ x & 0 < x < 1 \\ 1 & x \ge 1 \end{cases}$$

$$F_Y(y) = \begin{cases} 0 & y \le -1 \\ \dfrac{y+1}{2} & -1 < y < 1 \\ 1 & y \ge 1 \end{cases}$$

$$F_y(Y) = P\left(Y \le y\right) = P\left(g(x) \le y\right) = P\left(x \le g^{-1}(y)\right) = F_x\left(g^{-1}(y)\right)$$

$$F_y(Y) = F_x\left(g^{-1}(y)\right) = x$$

$$x = F_Y(y) = \begin{cases} 0 & y \le -1 \\ \dfrac{y+1}{2} & -1 < y < 1 \\ 1 & y \ge 1 \end{cases}$$

$$\boxed{y = F^{-1}{}_Y(x) = \begin{cases} 0 & x \le 0 \\ -1 + 2x & 0 < x < 1 \\ 1 & x \ge 1 \end{cases}}$$

```
% Generate a random variable following U(0,1)
Ui = rand(1,100000);

% Uniform distribution between - 1 and + 1 (use the inverse
% transform method)
Zi = -1 + 2 .* Ui;
```

Box-Muller Approach – Standard Normal $\boldsymbol{U(0,1) \rightarrow N(0,1)}$

- If U_1 and U_2 are independent random variates from U(0,1) – generated before.
- Then

$$Z_1 = \sqrt{-2\ln U_1}\cos 2\pi U_2$$

and

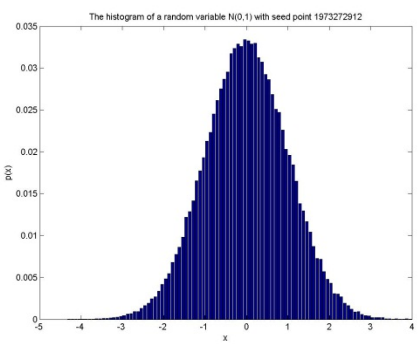
$$Z_2 = \sqrt{-2\ln U_1}\sin 2\pi U_2$$

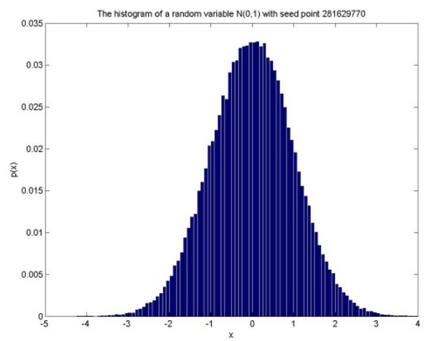
are ~ $N(0,1)$ and independent.

In Matlab ☺

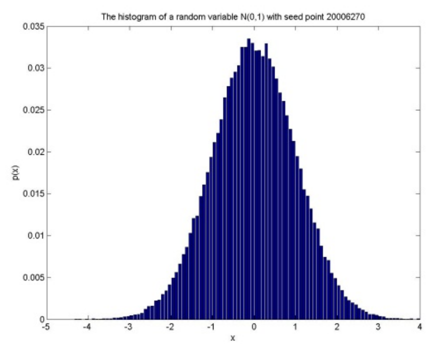
```
for i = 1 : length(Seeds)
    % setting the current seed
    rand('seed', Seeds(i));

    % Generate a random variable following U(0,1)
    Ui = rand(1,200000);

    % Extracting from the generated uniform random variable two
    % independent uniform random variables
    u1 = Ui(1:2:end);
    u2 = Ui(2:2:end);

    % Using u1 and u2, we will use Box-Muller method to generate the
    % random variable following standard normal
    Zi = sqrt((-2).*log(u1)) .* (cos(2*pi.*u2));

    if(i==1)
        [hi , bins_i] = hist(Zi,100);
        hi = hi./ 100000;
        bins = bins_i;
        H = hi;
    else
        [hi , bins_i] = hist(Zi,bins);
        hi = hi./ 100000;
        H = H + hi;
    end
end
```

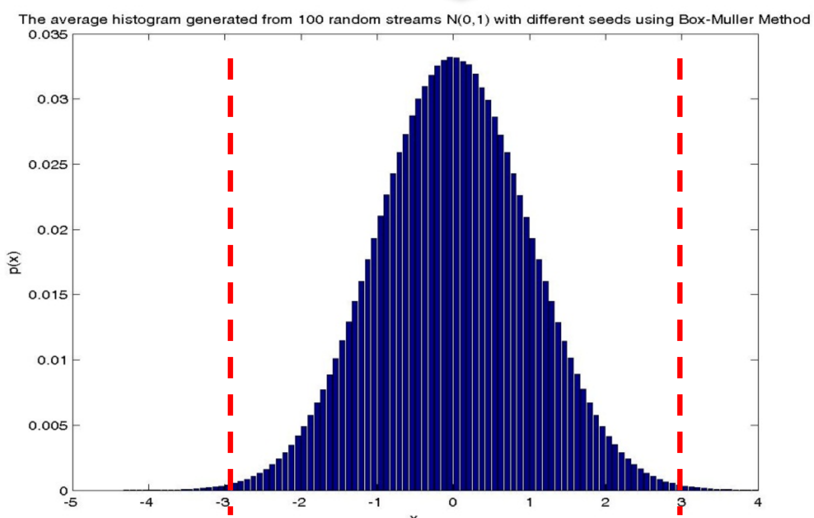
In Matlab ☺





•••





As shown when using more streams to obtain the histogram, the resultant becomes closer to the ideal standard normal where about 98% of the area under curve (pdf) lies in the interval [-3,3], centered at the zero mean.

Univariate Normal : $X \sim N(\mu, \sigma)$

- If U_1 and U_2 are independent random variates from U(0,1) – generated before.
- Then

$$Z = \sqrt{-2\ln U_1}\cos 2\pi U_2 \quad \sim \; N(0,1)$$

- Therefore,

$$X = \sigma Z + \mu \sim N(\mu, \sigma)$$

Multivariate Normal: $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

- Start with a d −dimensional vector of standard normal $N(0,1)$.
- These can be transofrmed to the desried distribution using

$$\mathbf{x} = \mathbf{R}^T \mathbf{z} + \boldsymbol{\mu}$$

$d \times d$ matrix
$\mathbf{R}^T \mathbf{R} = \boldsymbol{\Sigma}$

$d \times 1$

$d \times 1 \sim N(0,1)$

In Matlab ☺ $M = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 4 & 4 \\ 4 & 9 \end{bmatrix}$

```
% Generate a random variable following uniform(0,1)
U = rand(2,200000);

% Extracting from the generated uniform random variable two
% independent uniform random variables
u1 = U(:,1:2:end);
u2 = U(:,2:2:end);

% Using u1 and u2, we will use Box-Muller method to generate the
% random variable to follow standard normal
X = sqrt((-2).*log(u1)) .* (cos(2*pi.*u2));

% Now ... Manipulating the generated variable N(0,1) to follow
% certain mean and variance other than the standard normal

% First we will change its variance
% Getting the eigen vectors and values of the covariance matrix
% D is the eigen values matrix and V is the eigen vectors matrix
[V,D] = eig(CovM);
Y = zeros(X);

for j = 1 : size(X,2)
    Y(:,j) = V * sqrt(D) * X(:,j);
end

% Changing its mean
Ym = Y + repmat(Mu,1,size(Y,2));

beforeHist = histogram2(X(1,:),X(2,:));
beforeHist = beforeHist ./100000;

afterHist = histogram2(Ym(1,:),Ym(2,:));
afterHist = afterHist ./100000;
```

```
Mu = [1; 2];
CovM = [4 4; 4 9];
```

```
MuEstimated = mean(Ym')'
CovEstimated = cov(Ym')
```

MuEstimated =

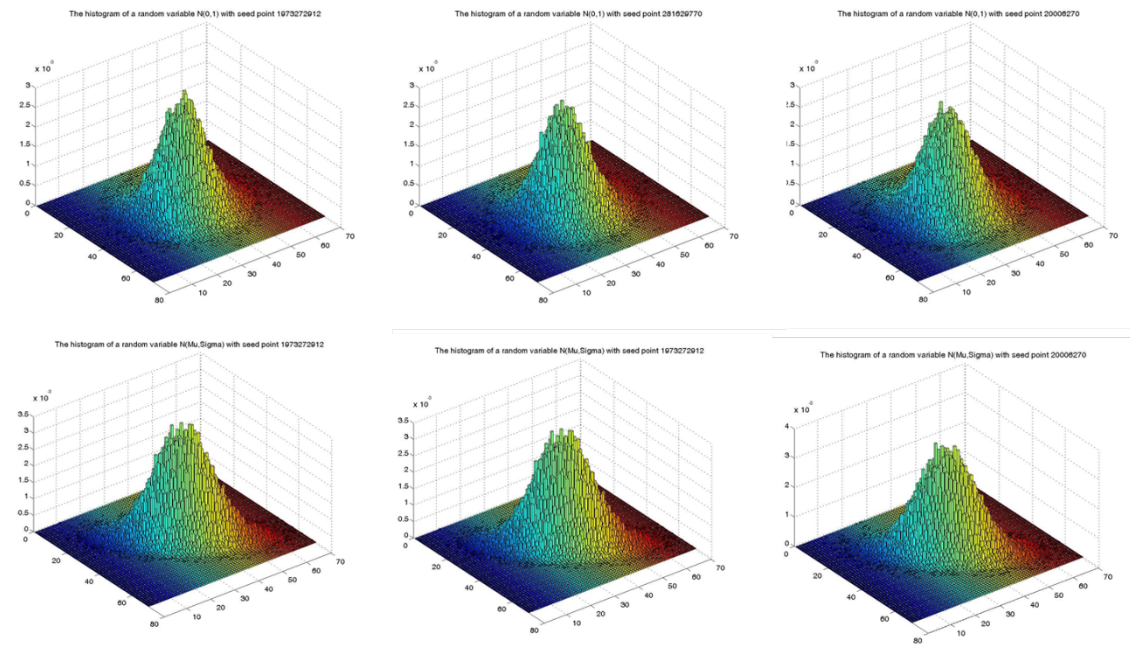
0.9919

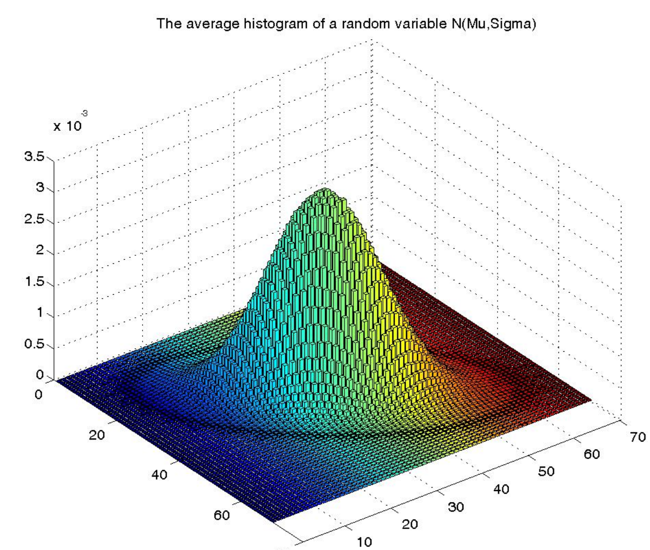
1.9868

CovEstimated =

4.0219	4.0308
4.0308	9.0534

In Matlab ☺ $M = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 4 & 4 \\ 4 & 9 \end{bmatrix}$





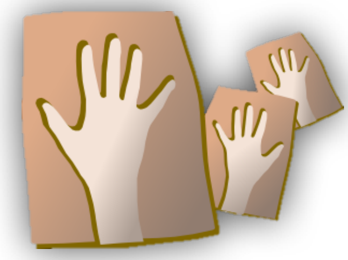
Probability Density Function

- The multi-normal Gaussian PDF can be computed using the following equation:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

where d is the dimension of the input vector **x.**

In Matlab ☺



```
% first we will generate the 2D matrices of the two features in order
% to visualize the probability as a surface due to memory contraints we
% will extract from the features representative samples in order to
% evaluate the probability
[N,xx] = hist(Ym(1,:), 100);
[N,yy] = hist(Ym(2,:), 100);
[x,y] = meshgrid(xx,yy);

% this will hold the probability values
z = zeros(size(x));
zEst = zeros(size(x));

for i = 1 : size(x,1)
    for j = 1 : size(x,2)
        z(i,j) = gausspdf([x(i,j) y(i,j)]', Mu, CovM);
        zEst(i,j) = gausspdf([x(i,j) y(i,j)]', MuEstimated, ...
                                                CovEstimated);
    end
end
```

```
function px = gausspdf (X,Mu,CovM)

N = length(X);
detCovM = det(CovM);
A = ((2*pi)^(N/2)) * sqrt(detCovM) ;
B = (X-Mu)' * inv(CovM) * (X-Mu);
px = real((1/A) * exp((-1/2)*B));
```

```
Mu = [1; 2];
CovM = [4 4; 4 9];
```

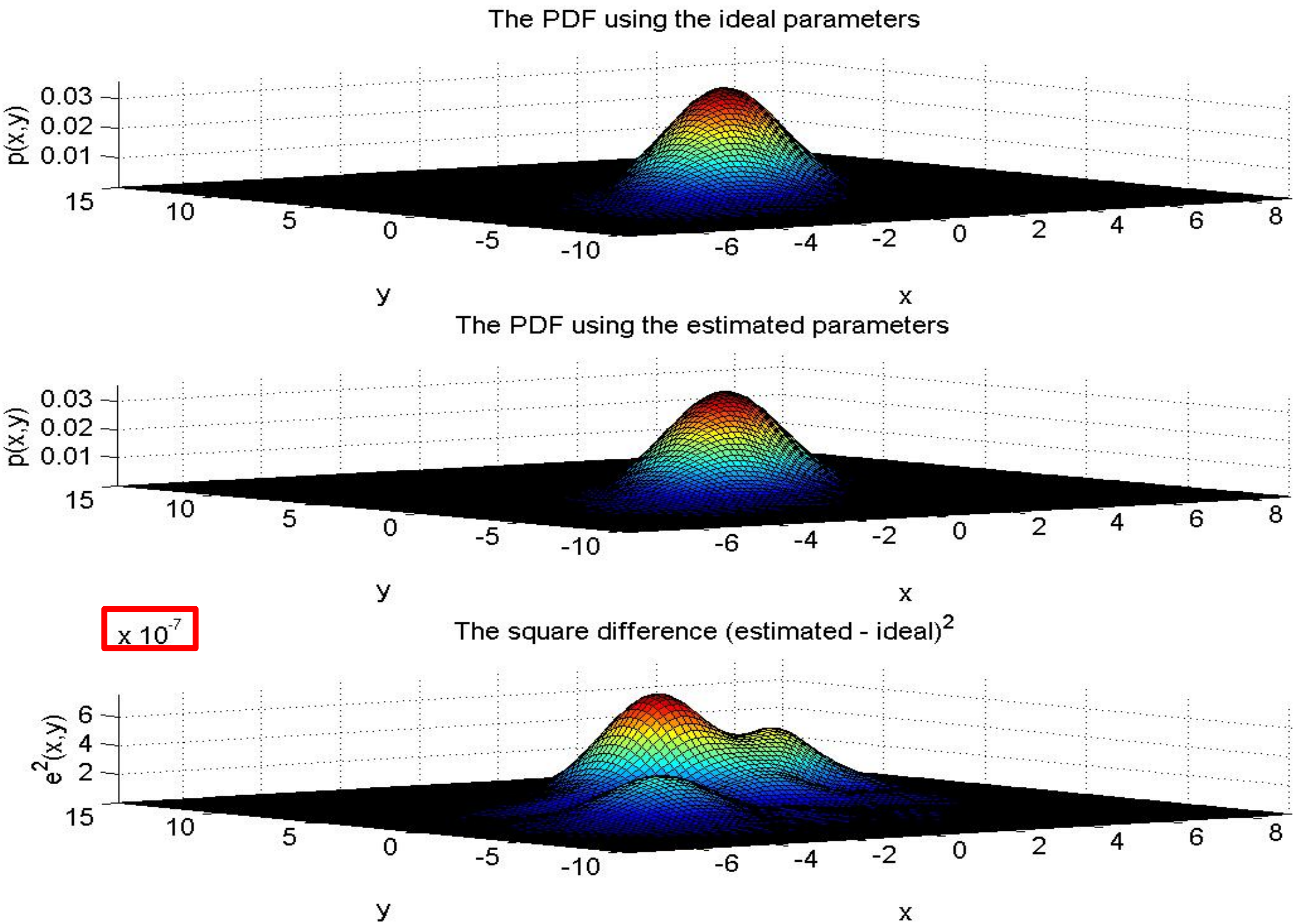
```
MuEstimated = mean(Ym')'
CovEstimated = cov(Ym')
```

```
MuEstimated =

    0.9919
    1.9868

CovEstimated =

    4.0219    4.0308
    4.0308    9.0534
```

In Matlab ☺

```
hfig = figure;
subplot(3,1,1);
surfc(x,y,z);
hold on
axis([min(min(x)) max(max(x)) min(min(y)) max(max(y)) min(min(z)) max(max(z))]);
title('The PDF using the ideal parameters');
xlabel('x');
ylabel('y');
zlabel('p(x,y)');
hold off

subplot(3,1,2);
surfc(x,y,zEst);
hold on
axis([min(min(x)) max(max(x)) min(min(y)) max(max(y)) min(min(zEst))
max(max(zEst))]);
title('The PDF using the estimated parameters');
xlabel('x');
ylabel('y');
zlabel('p(x,y)');
hold off

subplot(3,1,3);
error =(abs(zEst-z))^2;
surfc(x,y,error);
hold on
axis([min(min(x)) max(max(x)) min(min(y)) max(max(y)) min(min(error))
max(max(error))]);
title('The square difference (estimated - ideal)^2');
xlabel('x');
ylabel('y');
```

Results



The PDF using the ideal parameters
p(x,y)
0.03
0.02
0.01
15
10
5
0
-5
-10
-6
-4
-2
0
2
4
6
8
y
x
The PDF using the estimated parameters
p(x,y)
0.03
0.02
0.01
15
10
5
0
-5
-10
-6
-4
-2
0
2
4
6
8
y
x
x 10^-7
The square difference (estimated - ideal)^2
e^2(x,y)
6
4
2
15
10
5
0
-5
-10
-6
-4
-2
0
2
4
6
8
y
x

In Matlab ☺

$$M = \begin{bmatrix} 5 \\ -5 \\ 6 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 5 & 2 & -1 \\ 2 & 5 & 0 \\ -1 & 0 & 4 \end{bmatrix}$$

```
% Generate a random variable following uniform(0,1)
U = rand(3,200000);

% Extracting from the generated uniform random variable two
% independent uniform random variables
u1 = U(:,1:2:end);
u2 = U(:,2:2:end);

% Using u1 and u2, we will use Box-Muller method to generate the
% random variable to follow standard normal
X = sqrt((-2).*log(u1)) .* (cos(2*pi.*u2));

% Now ... Manipulating the generated variable N(0,1) to follow
% certain mean and variance other than the standard normal

% First we will change its variance
% Getting the eigen vectors and values of the covariance matrix
% D is the eigen values matrix and V is the eigen vectors matrix
[V,D] = eig(CovM);
Y = zeros(X);

for j = 1 : size(X,2)
    Y(:,j) = V * sqrt(D) * X(:,j);
end

% Changing its mean
Ym = Y + repmat(Mu,1,size(Y,2));
```

```
Mu = [5;-5;6];
CovM = [5 2 -1;2 5 0;-1 0 4];
```

```
MuEstimated = mean(Ym')'
CovEstimated = cov(Ym')
```

MuEstimated =

5.0134
-4.9836
5.9992

CovEstimated =

4.9838	2.0075	-0.9930
2.0075	4.9884	0.0114
-0.9930	0.0114	3.9861

In Matlab ☺

```matlab
% this will hold the probability values
[N,xx] = hist(Ym(1,:), 50);
[N,yy] = hist(Ym(2,:), 50);
[N,zz] = hist(Ym(3,:), 50);

[x,y,z] = meshgrid(xx,yy,zz);

% this will hold the probability values
w = zeros(size(x));
wEst = zeros(size(x));

for i = 1 : size(x,1)
    for j = 1 : size(x,2)
        for k = 1 : size(x,3)
            w(i,j,k) = gausspdf([x(i,j,k) y(i,j,k) z(i,j,k)]', Mu, CovM);
            wEst(i,j,k) = gausspdf([x(i,j,k) y(i,j,k) z(i,j,k)]', ...
                                    MuEstimated, CovEstimated);
        end
    end
end
```

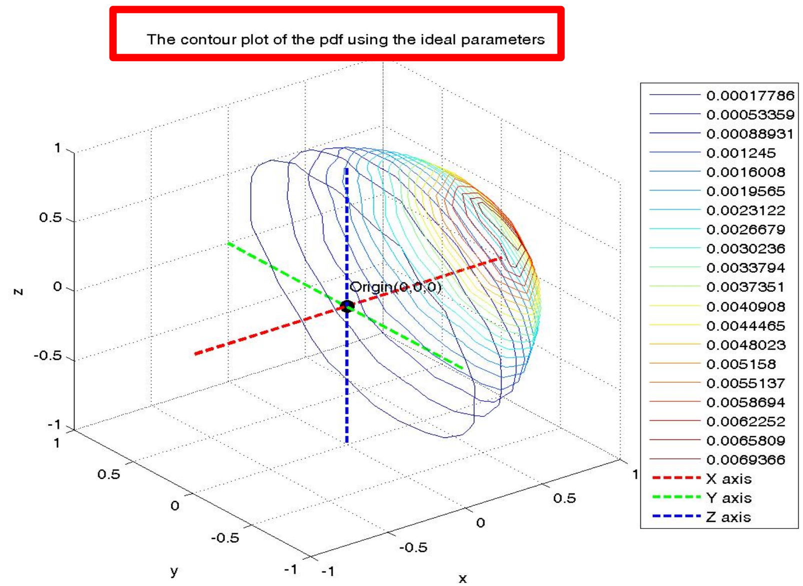
```matlab
Mu = [5;-5;6];
CovM = [5 2 -1;2 5 0;-1 0 4];
```

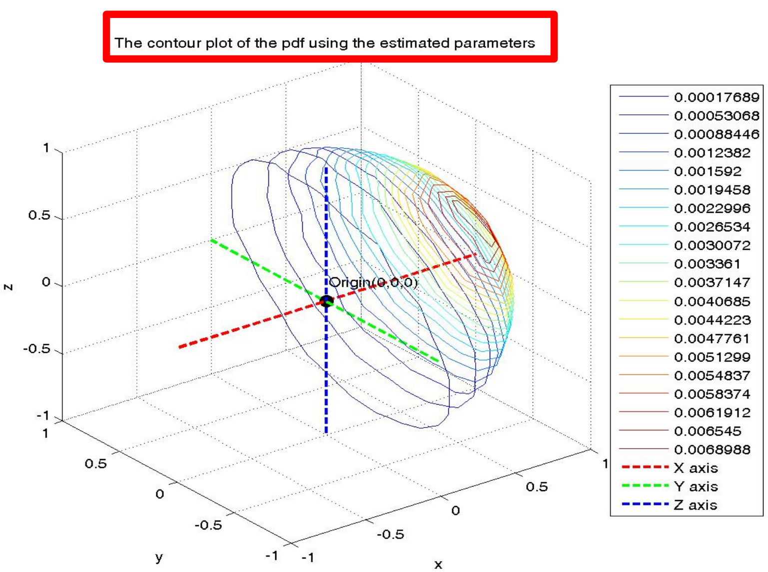
```matlab
MuEstimated = mean(Ym')'
CovEstimated = cov(Ym')
```

```
MuEstimated =

    5.0134
   -4.9836
    5.9992

CovEstimated =

    4.9838    2.0075   -0.9930
    2.0075    4.9884    0.0114
   -0.9930    0.0114    3.9861
```

In Matlab ☺

```
hfig = figure;

[N,ww] = hist(w(:),20);
[xi,yi,zi] = sphere;            % Plane to contour
contourslice(x,y,z,w,xi,yi,zi,20,'cubic');
hold on;

% plotting axes
a = -1:0.5:1;
ze = zeros(1,length(a));

plot3(a,ze,ze,'r--','LineWidth',2);
hold on
plot3(ze,a,ze,'g--','LineWidth',2);
hold on
plot3(ze,ze,a,'b--','LineWidth',2);
hold on
plot3(0,0,0,'ko','LineWidth',4);
hold on
text(0,0,0,'Origin(0,0,0)');
hold on

title('The contour plot of the pdf using the ideal parameters');
xlabel('x');
ylabel('y');
zlabel('z');
grid on;
view(3);
hold off
```





Thank You

Questions