Probability & Linear Algebra Hands-on with Scilab

Santosh Chapaneri

Random Number Generator

- A good random number generator should have the following characteristics:

1. **Randomness**: It should pass statistical tests of randomness.
2. **Long Period**: For obvious reasons.
3. **Efficiency**: This is important since simulations often require millions of random variables.
4. **Repeatability**: It should produce the same sequence of numbers if started in the same state. This allows the repetition and checking of simulations.

RNG - LCG

Linear Congruential Generators The simplest are the linear congruential generators. Starting with the seed x_0, these generate a sequence of integers by

$$x_k = (ax_{k-1} + c) \; mod \; M.$$

where a, c and M are given integers. All the x_k are integers between 0 and $M - 1$. In order to produce floating point numbers these are divided by M to give a floating point number in the interval [0, 1).

RNG - LCG

$$x_k = (ax_{k-1} + c) \; mod \; M$$

```
function [x] = lcg(n, a, c, m, x0)
x = zeros(1,n+1);
x(1) = x0;
for i = 2:n+1
    x(i) = pmodulo(a*x(i-1)+c, m);
end
x = x/m;
endfunction
```

```
-->xx = lcg(100, 1203, 0, 2048, 1);
-->plot(xx);
```

RNG - LCG

- Problem with LCG: Let $a = 1203$, $c = 0$, $m = 2048$, $x_0 = 1$
- For this case, period = 512 ☹
- `-->xx = lcg(511, 1203, 0, 2048, 1);`
- `-->xx(1:10);`

- Now take successive pairs of values as the x and y coordinates of a point in the plane and plot the results.
- `-->x = xx(1:2:511);`
- `-->y = xx(2:2:512);`
- `-->plot2d(x,y,style = -1);`

RNG - Scilab

- RNGs in Scilab are based on number theory and have quite sophisticated implementations.
- One in common use is the **Mersenne twister** which takes a vector of 625 as its seed.

- Two RNGs: **rand** and **grand**.
- Uniform random numbers are the default.

- rand(m, n) - gives a $m \times n$ matrix of random numbers.
- rand(m, n, ’normal’) - gives a $m \times n$ matrix of normally distributed random numbers.

Simulate Uniform Distribution

- Uniform distribution: Let $X \sim \mathrm{Unif}(0, 1)$
- Consider change of variable: $Y = (b-a)X + a$
- Write Scilab code to generate Unif(-5, 5)
- -->x =rand(1,10000);
- -->y = 10*x - 5;
- -->histplot(100,y)

Simulate Gaussian Distribution

- Normal distribution: Let $X \sim N(0, 1)$
- Consider change of variable: $Y = \mu + \sigma X$

- Write Scilab code to generate N(100, 10)

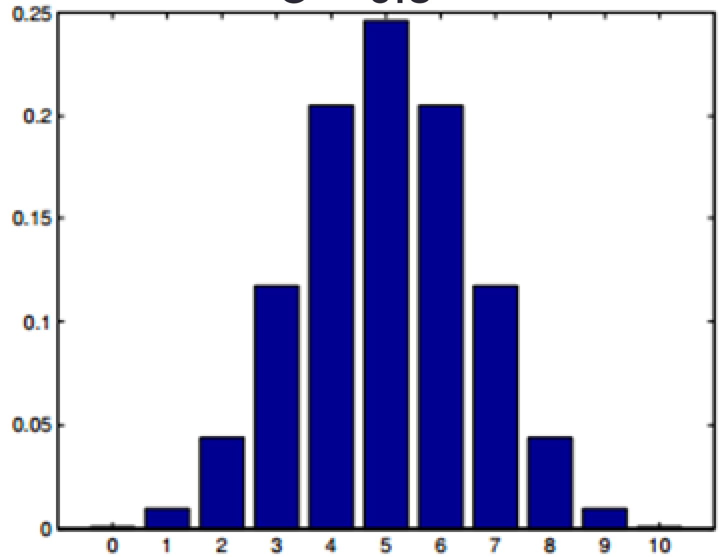
- -->x =rand(1,10000,'normal');
- -->y = 10*x + 100;
- -->histplot(100,y);

Simulate Binomial Distribution

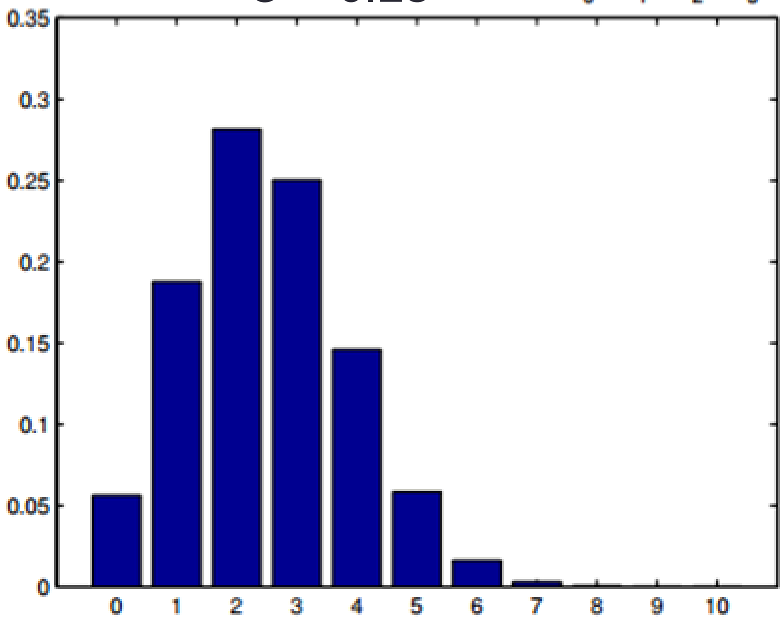
$$\mathrm{Bin}(k \mid n, \theta) =$$

$$\binom{n}{k} \theta^k (1-\theta)^{n-k}$$

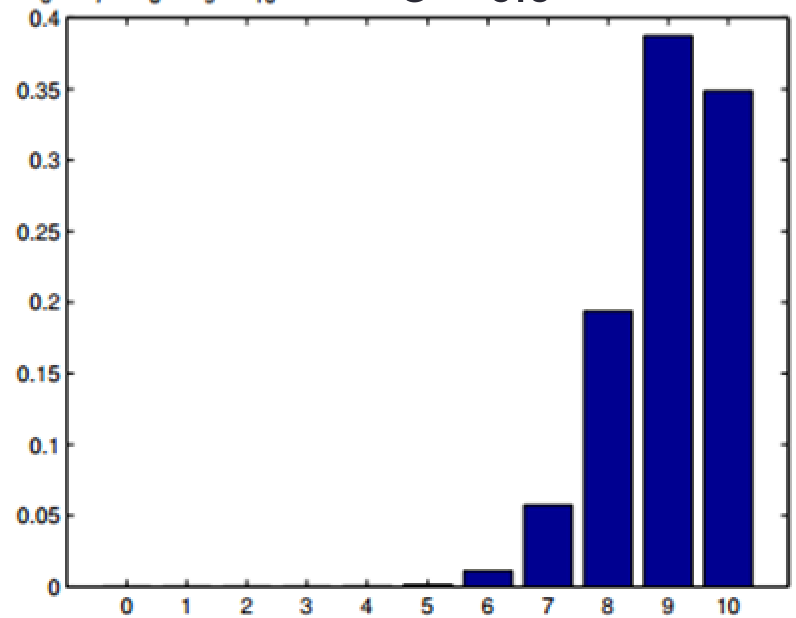
Θ = 0.5



Θ = 0.25



Θ = 0.9



Simulate Binomial Distribution

- $X \sim \text{Bin}(n = 5, p = 0.25)$

```
function [x] = mybinomial(s,n,p)
// simulation binomial
// s = Sample size
// n,p = parameters
x=0*ones(1:s);
y=rand(s,n);
for i=1:s
    for j=1:n
        if y(i,j)< p
            x(i)=x(i)+1;
        elseif y(i,j) >= p
            x(i)=x(i);
        end;
    end;
end;
endfunction
```

-->x = mybinomial(10000,5,0.25);
-->histplot(10,x);

-->x = mybinomial(10000,5,0.5);
-->histplot(10,x);

-->x = mybinomial(10000,5,0.75);
-->histplot(10,x);

-->x = mybinomial(10000,5,0.99);
-->histplot(10,x);

Simulate Exponential Distribution

- Exponential distribution: Let X ~ Exp(λ)

$$f(x) = \begin{cases} 0 & x < 0 \\ \lambda e^{-\lambda x} & x \geq 0 \end{cases}$$

$$F(x) = 1 - e^{-\lambda x}, \ x \geq 0$$

$$y = 1 - e^{-\lambda x}$$

$$x = F^{-1}(y) = -(1/\lambda)\ln(1 - y)$$

- Let X ~ Exp(2)
- `-->y = rand(1,10000); -->x = (-0.5)*log(1-y); -->histplot(100,x);`
- Compare with exact density:
- `-->xx = 0:0.01:5;`
- `-->plot2d(xx, 2*exp(-2*xx))`

Simulate Poisson Distribution

$$f_X(x) = \frac{e^{-\lambda} \cdot \lambda^x}{x!}, \quad x = 0,1,2,\ldots,\infty$$

Since Γ(n+1) = n!, we can use the gamma function to simulate Poisson RV.

```
-->deff('[fX] = fpoisson(x, lambda)','fX = exp(-lambda).*(lambda^x)./gamma(x+1)')
```

Let λ = 8.5 indicate mean value of vehicles per hour visiting service station

Prob. of exactly 10 vehicles visiting in an hour =
$P(X = 10) = e^{(-8.5)}(8.5)^{10}/10! = 0.1104$

Simulate:

```
-->lambda = 8.5;
-->fpoisson(10)
```

Simulate Poisson Distribution

Q: Prob. of 7 vehicles or less visiting in an hour =

$$P(X \le 7) = F_X(7) = \sum_{k=0}^{7} f_X(k) = \sum_{k=0}^{7} \frac{e^{-8.5} \cdot 8.5^k}{k!} = 0.3856$$

Simulate:

-->xx = [0:7]; -->pp = fpoisson(xx); -->prob = sum(pp)

Q: Prob. of 5 vehicles or more visiting in an hour =

$$P(X > 5) = 1 - P(X \le 5) = 1 - \sum_{k=0}^{5} \frac{e^{-8.5} \cdot 8.5^k}{k!} = 1 - 0.1496 = 0.8504$$

Simulate:

-->xx = [0:5]; -->pp = fpoisson(xx); -->prob = 1 - sum(pp)

Calculating Moments of RV

- The moments about origin are: $\mu'_k = \sum_{i=1}^{n} x_i^k \cdot f_X(x_i)$

- The moments about mean are: $\mu_k = \sum_{i=1}^{n} (x_i - \mu_X)^k \cdot f_X(x_i)$

$$\mu_X = \mu'_1 = \Sigma\, x_i \cdot f_X(x_i)$$

x	-2	-1	0	1	2
$f_X(x)$	0.2	0.3	0.1	0.2	0.2

- -->X = [-2:1:2];
- -->fX = [0.2 0.3 0.1 0.2 0.2];
- -->deff('[mu_k_p] = mukp(k)','mu_k_p = sum((X.^k).*fX)');
- -->deff('[mu_k] = muk(k)','mu_k = sum(((X-muX).^k).*fX)')

Calculating Moments of RV

- The 3rd moment about origin is
- -->mukp(3) (-0.8)
- The 4th moment about origin is
- -->mukp(4) (8.4)
- The 0th moment about origin is, by def, 1

$$\mu'_k = \sum_{i=1}^{n} x_i^k \cdot f_X(x_i)$$

- -->mukp(0) (1)

$$\mu'_0 = \Sigma\ x_i^0 \cdot f_X(x_i) = \Sigma\ f_X(x_i) = 1$$

- The 1st moment about origin is (mean value of distribution)
- -->muX = mukp(1) (-0.2)

Calculating Moments of RV

- The **Variance** is Var(X) = μ_2
- -->muk(2)   (2.36)
- Verify variance by definition:

$$Var(X) = \Sigma\ (x_i - \mu_X)^2 \cdot f_X(x_i) = \Sigma\ x_i^2 \cdot f_X(x_i) - \mu_X^2 = \mu'_2 - (\mu'_1)^2$$

- -->mukp(2)-mukp(1).^2

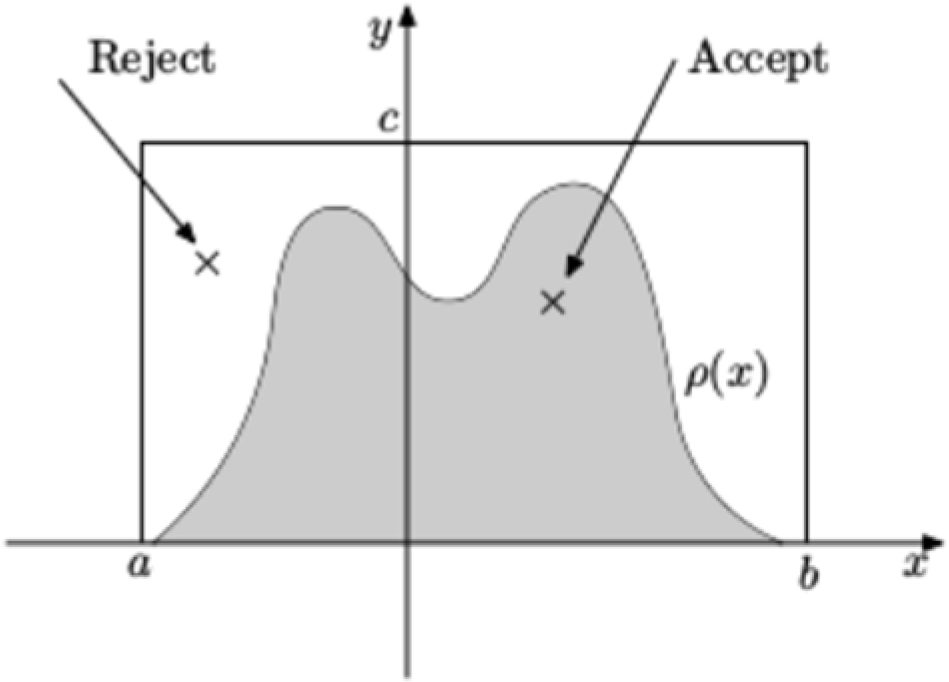
- **Skewness & Kurtosis**:
- -->sigmak = muk(3)/(muk(2).^1.5)
- --> kappa = muk(4)/(muk(2).^2)

skewness: $\sigma k = \mu_3 / \mu_2^{1.5}$

kurtosis: $\kappa = \mu_4 / \mu_2^{2}$

Acceptance/Rejection Method

Suppose that $\rho(x)$ is zero outside of the interval $[a, b]$ and furthermore that $\rho(x)$ is bounded above by c.



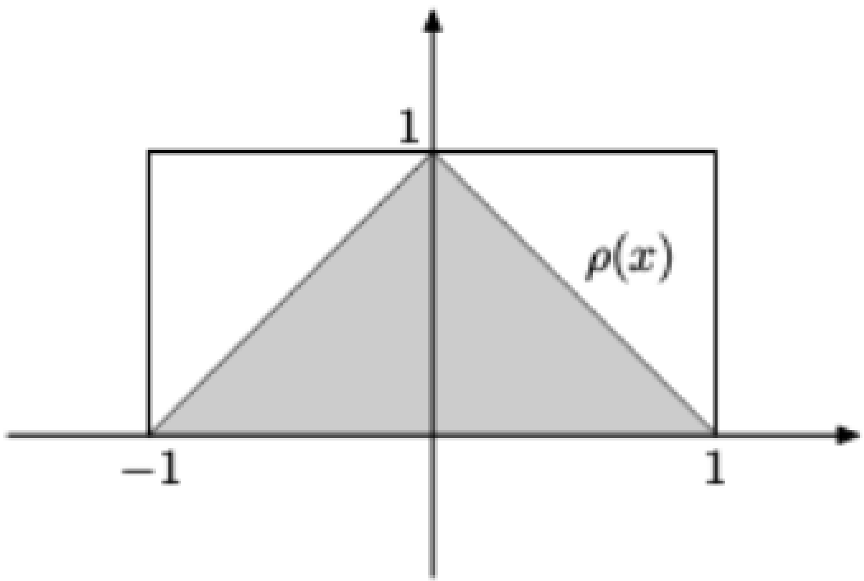
Now generate points (x_i, y_i) with x_i uniformly distributed in $[a, b]$ and y_i uniformly distributed in $[0, c]$.

If $y_i \leq \rho(x_i)$ then we accept the value x_i, if $y_i \geq \rho(x_i)$ then we reject the value x_i. The values x_i which are accepted will have the probability density $\rho(x)$.

Acceptance/Rejection Method

Consider the hat-shaped probability density defined on $[-1, 1]$ by

$$\rho(x) = \begin{cases} x + 1 & -1 \le x \le 0 \\ 1 - x & 0 \le x \le 1 \end{cases}$$



Write a Scilab function to generate n random numbers with this density using the acceptance/rejection method.

Acceptance/Rejection Method

```
function [y] = rhohat(x)
// First we need the density function
if (x < 0)
    y = x + 1;
else
    y = 1 - x;
end
endfunction
```

Acceptance/Rejection Method

```
function [x] = randhat(n)
// Now the random number generator
x = zeros(1,n);
k = 0 // keep count of numbers generated
while (k < n)
    xx = -1 + 2*rand(1,1); // uniform on [-1,1]
    yy = rand(1,1);
    if (yy <= rhohat(xx)) // accept xx
        k = k + 1;
        x(k) = xx;
    end
end
endfunction
```

```
-->x = randhat(1000);
--> histplot(100,x);
```

Simulate Bayesian Inference

$$\underbrace{p(\theta \mid y)}_{\text{Posterior}} = \frac{\overbrace{p(y \mid \theta)}^{\text{Likelihood}}\,\overbrace{p(\theta)}^{\text{Prior}}}{\underbrace{p(y)}_{\text{Evidence}}}$$

Simulate Bayesian Inference

- Let $X \in \{0,1\}$ represent tails/ heads.
- Suppose $P(X=1)=\theta$. Then

$$P(x|\theta) = \text{Be}(X|\theta) = \theta^{x}(1-\theta)^{1-x}$$

- Given $D=(x_1,\ldots,x_N)$, the likelihood is

$$p(D|\theta) = \prod_{n=1}^{N} p(x_n|\theta) = \prod_{n=1}^{N} \theta^{x_n}(1-\theta)^{1-x_n} = \theta^{N_1}(1-\theta)^{N_0}$$

where $N_1 = \sum_n x_n$ is the number of heads and $N_0 = \sum_n (1-x_n)$ is the number of tails (sufficient statistics). Obviously $N = N_0 + N_1$.

Simulate Bayesian Inference

- Let $X \in \{1, \ldots, N\}$ represent the number of heads in N trials. Then X has a binomial distribution

$$p(X|N) = \binom{N}{X} \theta^X (1-\theta)^{N-X}$$

where

$$\binom{N}{X} = \frac{N!}{(N-X)!X!}$$

is the number of ways to choose X items from N.

- Suppose we have a coin with probability of heads θ. How do we estimate θ from a sequence of coin tosses $D = (X_1, \ldots, X_n)$, where $X_i \in \{0, 1\}$?

MLE

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(D|\theta)$$

Bayesian

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{\int_{\theta'} p(\theta', D)}$$

Simulate Bayesian Inference

Prior: $$p(\theta) = Be(\theta|\alpha_1, \alpha_0) \propto [\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1}]$$

Posterior: $$P(\theta|D) \propto Be(\theta|\alpha_1 + N_1, \alpha_0 + N_0)$$

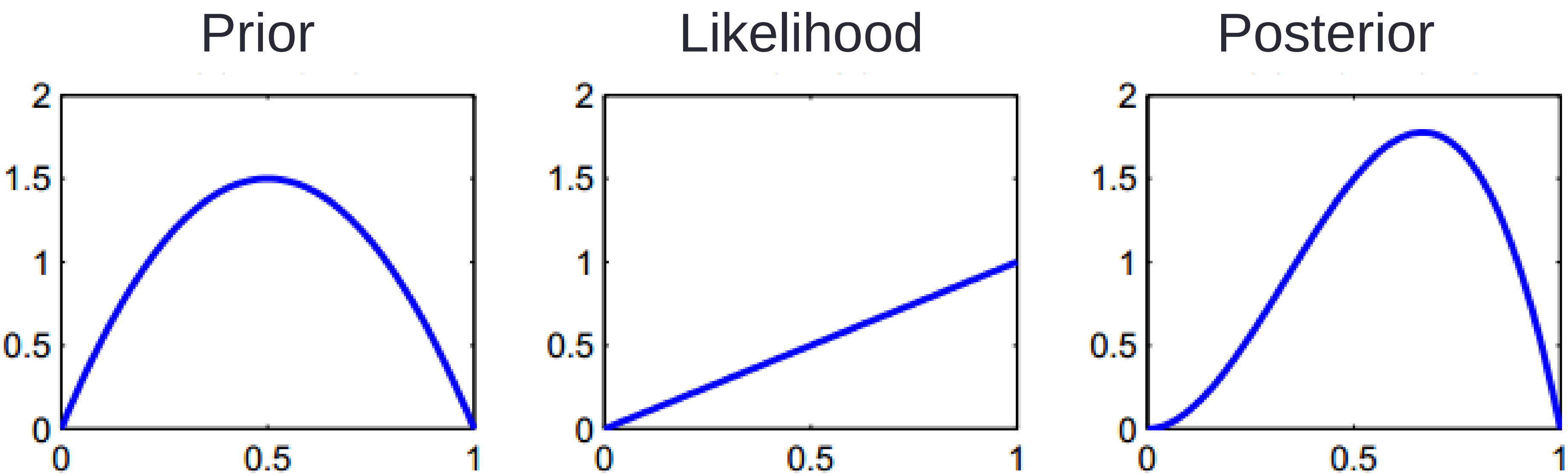
Start with Be($\vartheta|\alpha_0 = 2, \alpha_1 = 2$) and observe $x = 1$, so the posterior is Be($\vartheta|\alpha_0 = 3, \alpha_1 = 2$).

```
thetas = 0:0.01:1;
alpha1 = 4;
alpha0 = 2;
N1 = 1; N0 = 0; N = N1 + N0;

prior = distfun_betapdf(thetas, alpha1, alpha0);
lik = (thetas.^N1) .* (1-thetas).^N0;
post = distfun_betapdf(thetas, alpha1+N1, alpha0+N0);

subplot(1,3,1);plot(thetas, prior);
subplot(1,3,2);plot(thetas, lik);
subplot(1,3,3);plot(thetas, post);
```

Simulate Bayesian Inference



Simulate Central Limit Theorem

```
samples = 10000;
bins = 20;
N = [1 5 10 50];
for i=1:length(N)
   figure(i);
   X = mean(rand(samples,N(i)),2);
   [counts] = histc(bins,X,normalization=%t);
   bar(counts);
end
```

Simulate Entropy of Bernoulli RV

$$H[p(\mathbf{x})] = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}$$

$$\begin{aligned} \mathbb{H}(X) &= -[p(X=1)\log_2 p(X=1) + p(X=0)\log_2 p(X=0)] \\ &= -[\theta \log_2 \theta + (1-\theta)\log_2(1-\theta)] \end{aligned}$$

```
// Bernoulli Entropy
x = 0.0001:0.0001:0.9999;
H = -(x.*log2(x) + (1-x).*log2(1-x));
plot(x,H);
```

Simulate Monte Carlo Integration

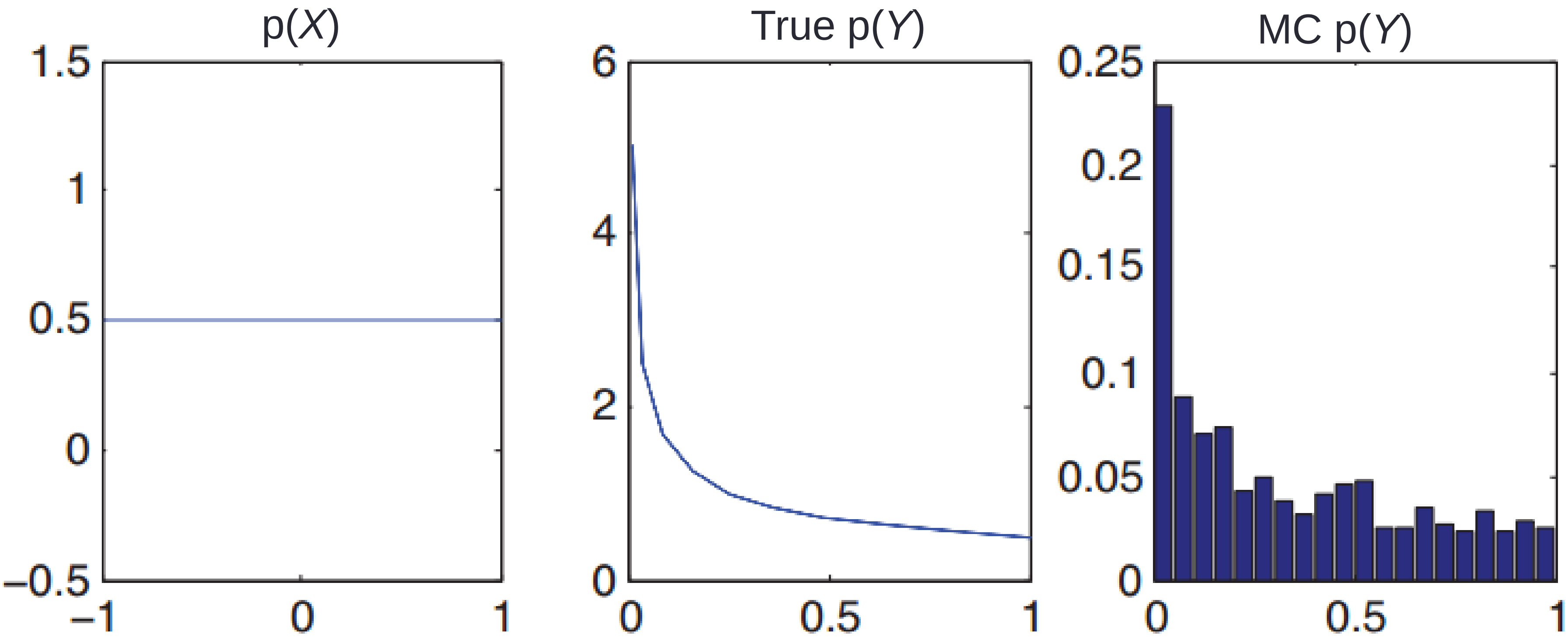
- Used to approximately calculate an integral analytically
- Generate S samples from the distribution, call them $x_1, \ldots, x_S$
- Given the samples, we can approximate the distribution of $f(X)$ by using the empirical distribution of $\{f(x_s)\}_{s=1}^{S}$

- Eg: Using Monte Carlo, we can approximate the expected value of any function of RV: Simply draw samples and compute arithmetic mean!

$$\mathbb{E}[f(X)] = \int f(x)p(x)dx \approx \frac{1}{S}\sum_{s=1}^{S} f(x_s)$$

Simulate Monte Carlo Integration

- Eg: **Change of Variables using Monte Carlo**
- Let $X \sim \text{Unif}(-1, 1)$, $Y = X^2$.
- Approximate p(Y) by drawing samples from p(X), squaring them and computing the resulting empirical distribution.



Simulate Monte Carlo Integration

Compute the distribution of $Y = X^2$, where $X \sim$ Unif(-1,1) using Monte Carlo approximation.

```
// Change of Variables Demo 1D
// x ~ U(-1,1), y=x^2, plot pdf of y
xs = -1:0.1:1;
a = -1; b = 1;
px = 1/(b-a) * ones(1,length(xs));
ys = xs.^2;
ys(ys==0) = [];

// True distribution
ppy = 1 ./(2*sqrt(ys));

// Monte Carlo approximation
n=1000;
samples = rand(1, 1000)*(b-a) + a; // sample from U(a,b)
samples2 = samples.^2;
[h] = histc(20,samples2,normalization=%t);

figure(2); nr = 1; nc = 3;
subplot(nr,nc,1); plot(xs, px, '-');
subplot(nr,nc,2); plot(ys, ppy, '-');
subplot(nr,nc,3);  bar(h);
```

Box-Muller Simulation

a) Generate two random numbers u_1, u_2 ~ Unif(0,1)

b) Use these to find radius and angle

$$R = \sqrt{-2log\left(u_1\right)} \qquad \theta = 2\pi u_2$$

c) Convert from polar to Cartesian co-ordinates: $(Rcos\theta, Rsin\theta)$

```
function [y1,y2] = myrandnorm(n)
x1 = rand(1,n);
x2 = rand(1,n);
y1 = sin(2*%pi*x1).*sqrt(-2*log(x2));
y2 = cos(2*%pi*x1).*sqrt(-2*log(x2));
endfunction
```

```
-->[y1, y2] = myrandnorm(10000);
-->histplot(100,y1);
-->histplot(100,y2);
```

We can compare the histogram to exact density:

```
-->xx = -4:0.01:4; -->yy = exp(-(xx.^ 2)/2)/sqrt(2*%pi);
-->plot2d(xx, yy)
```

Linear Algebra with Scilab

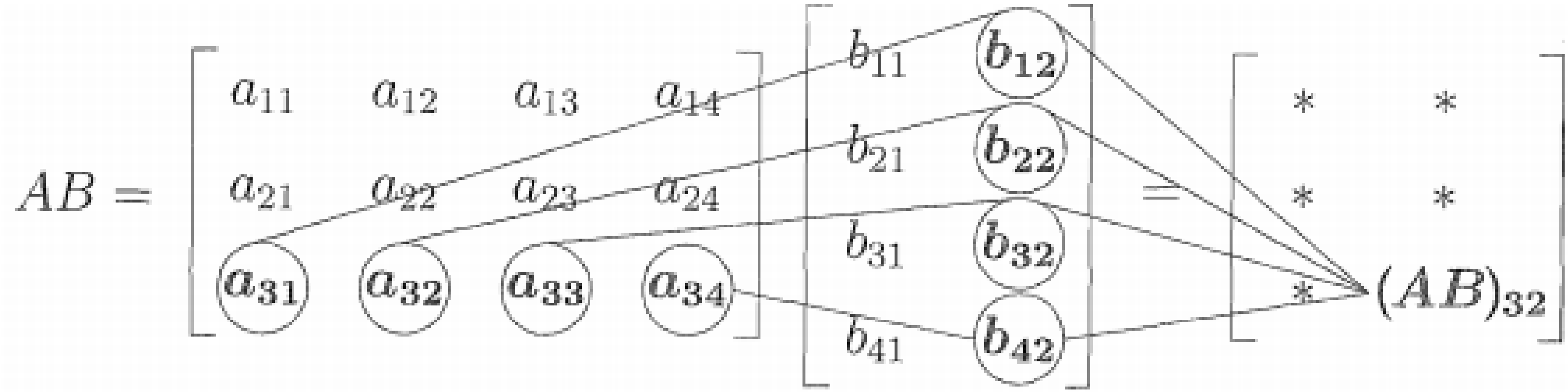
Simulate Solving System of Equations

$$Ax = b$$

Determine *x*, given *A* and *b*.

```
--> A=[0 -1 1; 1 -1 -1; -1 0 -1];
--> disp (A, 'A= ');
--> b=[3;0;-3];
--> disp (b, ‘b= ');
--> disp (x, 'A\b= ')
```

Simulate Matrix Multiplication




```
--> A =[2 3;4 0];
--> disp (A, 'A= ');
--> B =[1 2 0;5 -1 0];
--> disp (B, 'B= ');
--> disp (A*B, 'AB= ')
```

Simulate Triangular Factorization

$$A = LU$$

L is lower triangular with 1's on the diagonal.
The diagonal entries of U are the pivots.

```
--> A =[1 2;3 8];
--> disp (A, 'A= ')
--> [L,U]= lu(A);
--> disp (L, 'L= ');
--> disp (U, 'U= ')
--> disp (L*U, 'LU= ')
```

Simulate Cholesky Factorization

$$A = R'*R$$

R is upper triangular matrix.

```
W=rand(5,5)+%i*rand(5,5);

X=W*W';

R=chol(X); // upper triangular

matrix s.t. R'*R = X

norm(R'*R - X)
```

Simulate Linear Independence

Suppose c_1v_1 +…+c_kv_k = 0 only happens when c_1 = … = c_k = 0.
Then the vectors v_1, … v_k are ***linearly independent***.
If any c's are nonzero, the v's are ***linearly dependent***.

$$A = \begin{bmatrix} 1 & 3 & 3 & 2 \\ 2 & 6 & 9 & 5 \\ -1 & -3 & 3 & 0 \end{bmatrix}$$

```
--> A =[1 3 3 2;2 6 9 5; -1 -3 3 0];
--> disp (A)
--> B=A;
--> disp ('C2−>C2−3∗C1')
--> A(: ,2)=A(: ,2) -3*A(: ,1);
--> disp (A)
--> disp ('R3−>R3−2∗R2+5∗R1')
--> B(3 ,:)=B(3 ,:) -2*B(2 ,:) +5*B(1 ,:);
--> disp (B)
```

Simulate Four Fundamental Subspaces

1. $C(A)$ = column space of A; dimension r.

2. $N(A)$ = nullspace of A; dimension $n-r$.

3. $C(A^{\mathrm{T}})$ = row space of A; dimension r.

4. $N(A^{\mathrm{T}})$ = left nullspace of A; dimension $m-r$.

$A = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}$ has $m = n = 2$, and rank $r = 1$.

1. The ***column space*** contains all multiples of $\left[\begin{smallmatrix} 1 \\ 3 \end{smallmatrix}\right]$. The second column is in the same direction and contributes nothing new.

2. The ***nullspace*** contains all multiples of $\left[\begin{smallmatrix} -2 \\ 1 \end{smallmatrix}\right]$. This vector satisfies $Ax = 0$.

3. The ***row space*** contains all multiples of $\left[\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right]$. I write it as a column vector, since strictly speaking it is in the column space of A^{T}.

4. The ***left nullspace*** contains all multiples of $y = \left[\begin{smallmatrix} -3 \\ 1 \end{smallmatrix}\right]$. The rows of A with coefficients -3 and 1 add to zero, so $A^{\mathrm{T}}y = 0$.

Simulate Four Fundamental Subspaces

```
--> A =[1 2;3 6];
--> disp (A, 'A= ');
--> [m,n]= size (A);
--> [v, pivot ]= rref (A);
--> r= length ( pivot );
--> disp (r, 'rank= ')
--> cs=A(:, pivot );
--> disp (cs , 'Column space= ');
--> ns= kernel (A);
--> disp (ns , 'Null space= ');
--> rs=v (1:r ,:)';
--> disp (rs , 'Row space= ')
--> lns = kernel (A');
--> disp (lns , 'Leftnull space= ');
```

Simulate Orthogonality

(2, 2, -1) is orthogonal to (-1, 2, 2).

```
--> x1 =[2;2;-1]; --> disp(x1); --> x2 =[-1;2;2]; --> disp(x2);
--> disp(x1'*x2)
```

Null space is orthogonal to row space.

Rank-1 matrix

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \\ 3 & 9 \end{bmatrix}$$

Row space = [1 3] & Null space = [-3 1].

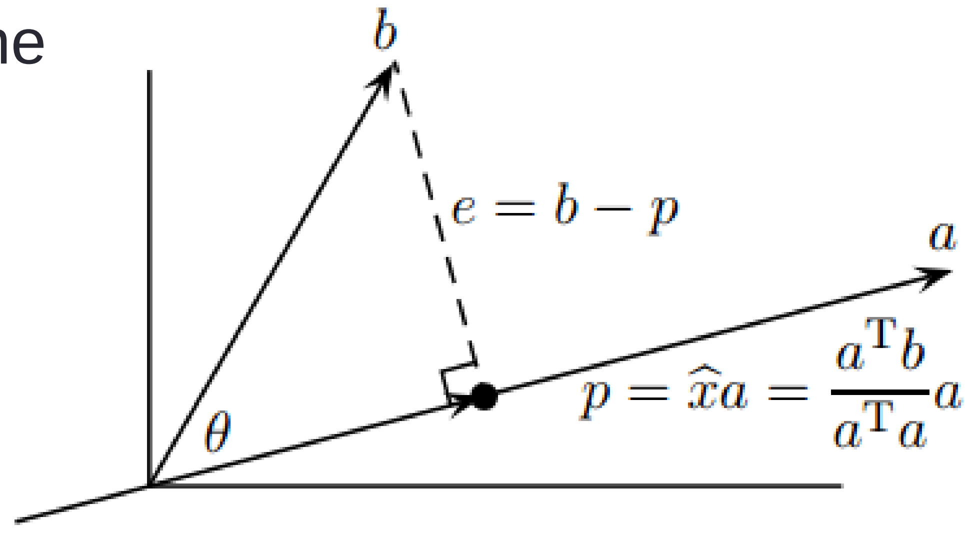
```
--> A =[1 3;2 6;3 9];
--> disp (A);
--> ns= kernel (A);
--> disp (ns);
--> disp (A(1 ,:)*ns);
--> disp (A(2 ,:)*ns);
--> disp (A(3 ,:)*ns);
```

Simulate Projections onto Lines

Project $b = (1;2;3)$ onto the line through $a = (1;1;1)$

$$\widehat{x} = \frac{a^{\mathrm{T}}b}{a^{\mathrm{T}}a} = \frac{6}{3} = 2.$$

$$\cos\theta = \frac{a^{\mathrm{T}}b}{\|a\|\,\|b\|} = \frac{6}{\sqrt{3}\sqrt{14}}$$



The projection p of b onto a, with $\cos\theta = \frac{Op}{Ob} = \frac{a^{\mathrm{T}}b}{\|a\|\,\|b\|}$

```
--> b =[1;2;3]; disp (b);
--> a =[1;1;1]; disp (a);
--> x=(a'*b)/(a'*a); disp (x*a);
--> cosTheta = (a'*b)/( sqrt(a'*a)*sqrt(b'*b));
--> disp(cosTheta);
```

Simulate Projection Matrix

Projection matrix $P = A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$

Suppose *A* is invertible. If it is 4 by 4, then its four columns are independent and its column space is all of $\mathbf{R}^4$. What is the projection onto the whole space?

$$P = A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} = AA^{-1}(A^{\mathrm{T}})^{-1}A^{\mathrm{T}} = I$$

```
--> A= rand (4 ,4); disp (A);
--> P=A*inv(A'*A)*A'; disp (P);
```

Simulate Least Squares Fitting of Data

$$Ax = b$$

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

```
--> A =[1 -1;1 1;1 2];
--> b =[1;1;3];
--> x= zeros (1, 2);
--> x=(A'*A)\(A'*b);
--> disp(x(1,1));
--> disp(x(2,1));
```

$$A^{\mathrm{T}}A\hat{x} = A^{\mathrm{T}}b$$

Simulate Gram-Schmidt Orthogonalization

orthogonalize	normalize
$v_1 \leftarrow u_1,$	$w_1 \leftarrow v_1 / \|v_1\|$
$v_2 \leftarrow u_2 - \mathrm{proj}_{v_1}(u_2)$	$w_2 \leftarrow v_2 / \|v_2\|$
$v_3 \leftarrow u_3 - \mathrm{proj}_{v_1}(u_3) - \mathrm{proj}_{v_2}(u_3)$	$w_3 \leftarrow v_3 / \|v_3\|$
$\vdots$	
$v_n \leftarrow u_n - \sum_{i=1}^{n-1} \mathrm{proj}_{v_i}(u_n)$	$w_n \leftarrow v_n / \|v_n\|$

Simulate Gram-Schmidt Orthogonalization

```
function Q = mygramschmidt( A )
// Computes orthonormal basis from independent vectors in A.
n = size( A, 2 );
Q = zeros( n, n );

for j = 1 : n
    u = A( :, j );
    for i = 1 : j - 1
        u = u - myproj( Q(:,i), A(:,j) );
    end
    Q(:,j) = u ./ norm( u, 2 );
end
endfunction

// projects a vector "a" on a direction "e"
```

```
function p = myproj( e, a )
p = (e' * a) / (e' * e) .* e;
endfunction
```

```
--> A =[1 0 1;1 0 0;2 1 0];
--> Q = mygramschmidt(A)
```

0.4082483	- 0.5773503	0.7071068
0.4082483	- 0.5773503	- 0.7071068
0.8164966	0.5773503	7.850D-17

Simulate Cramer's Rule

$$Ax = b$$

Cramer's rule: The jth component of $x = A^{-1}b$ is the ratio

$$x_j = \frac{\det B_j}{\det A}, \quad \text{where} \quad B_j = \begin{bmatrix} a_{11} & a_{12} & b_1 & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & b_n & a_{nn} \end{bmatrix} \text{ has } b \text{ in column } j$$

$$\begin{aligned} x_1 &+ 3x_2 &= 0 \\ 2x_1 &+ 4x_2 &= 6 \end{aligned}$$

$$x_1 = \frac{\begin{vmatrix} 0 & 3 \\ 6 & 4 \end{vmatrix}}{\begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}} = \frac{-18}{-2} = 9, \qquad x_2 = \frac{\begin{vmatrix} 1 & 0 \\ 2 & 6 \end{vmatrix}}{\begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}} = \frac{6}{-2} = -3$$

Simulate Cramer's Rule

$$\begin{aligned} x_1 &+ 3x_2 &= 0 \\ 2x_1 &+ 4x_2 &= 6 \end{aligned}$$

$$x_1 = \frac{\begin{vmatrix} 0 & 3 \\ 6 & 4 \end{vmatrix}}{\begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}} = \frac{-18}{-2} = 9, \qquad x_2 = \frac{\begin{vmatrix} 1 & 0 \\ 2 & 6 \end{vmatrix}}{\begin{vmatrix} 1 & 3 \\ 2 & 4 \end{vmatrix}} = \frac{6}{-2} = -3$$

```
--> A =[1 3;2 4];

--> b =[0;6];

--> X1 =[0 3;6 4];

--> X2 =[1 0;2 6];

--> x1 = det(X1)/det (A); disp(x1);

--> x2 = det(X2)/det (A); disp(x2);
```

Simulate Eigen Analysis

$$(A - \lambda I)x = 0$$

$$A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{has} \quad \lambda_1 = 3 \quad \text{with} \quad x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \lambda_2 = 2 \quad \text{with} \quad x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
--> A =[3 0;0 2];
--> eig = spec (A);
--> [V, Val ]= spec (A);
--> disp (eig);
--> x1=V(: ,1); disp(x1);
--> x2=V(: ,2); disp(x2);
```

Simulate Singular Value Decomposition

Any m by n matrix A can be factored into

$$A = U\Sigma V^{\mathrm{T}} = (\textbf{orthogonal})(\textbf{diagonal})(\textbf{orthogonal})$$

$$A = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = U_{3\times 3}\Sigma_{3\times 1}V_{1\times 1}^{\mathrm{T}}$$

```
--> A=[ -1 2 2]'; disp (A);

--> [U S V]= svd (A);

--> disp (U, 'U='); disp (S, 'S='); disp (V','V=');

--> result = U*diagonal*V'; disp(result);
```

Simulate Simplex Method for LPP

Minimize $7x_3 - x_4 - 3x_5$ subject to
$$\begin{aligned} x_1 \quad & + x_3 + 6x_4 + 2x_5 = 8 \\ x_2 + x_3 & \quad\quad\quad + 3x_5 = 9 \end{aligned}$$

$x^* = (0, 0, 0, \frac{1}{3}, 3)$

cost $-9\frac{1}{3}$

```
--> A =[1 0 1 6 2;0 1 1 0 3];
--> b =[8 9]';
--> c =[0 0 7 -1 -3]';
--> lb =[0 0 0 0 0]'; ub =[];
--> [x, lagr, f]= linpro(c, A, b, lb, ub);
--> disp (x, 'New corner: ');
--> disp (f, 'Minimum cost: ');
```