University of Southern California EE511

Continuous Sampling

Project #6

Name: Wu Jiawei
USCID: 9600392575

Abstract

In the project of continuous sampling, three experiments are conducted using Matlab. The core method of the project is to generate continuous random variables from different distributions and finish simulation using mathematical methods. The theories of alpha-stable distribution, Gamma distribution, the Box-Muller and the Polar Marsaglia methods are applied comprehensively in the lab. The experiments are repeated multiple times and the outcomes are shown in diagrams and calculations. The comparisons are conducted between the experimental results and the theoretical probability density functions.

Introduction

Three experiments are conducted in the lab. All the samples are generated by random selection and the experiments are repeated multiple times. The goal of the first trial is to generate independent random variables from normal distribution using Box Muller and Polar Marsaglia methods and compute the statistical values of random variables. The aim of the second experiment is to generate samples following the Gamma distribution using an accept-reject method. The objective of the third experiment is running simulation to sample from alpha-stable distribution of different parameters using Chambers-Mallows-Stuck method and verify the results by comparing them with the theoretical alpha-stable probability density function.

Methodology & Results

Experiment No.1

Description of Algorithm:

The Boxer-Muller transform method is utilized in this experiment

1.Generate U_1 and U_2 ~ $\mathrm{U}[0,1]$

2.$\mathrm{R}^2 = -2\log \mathrm{U}_1$ and $\theta = 2\pi \mathrm{U}_2$

3.$\mathrm{X} = \mathrm{R}\cos\theta = \sqrt{-2\log\mathrm{U}_1}\cos(2\pi\mathrm{U}_2)$

4. $\mathrm{Y} = \mathrm{R}\cos\theta = \sqrt{-2\log\mathrm{U}_1}\sin(2\pi\mathrm{U}_2)$

5. X and Y are N(0, 1) random variables, and independent.

The Polar method is also utilized in the experiment

1.Take U_1 and U_2 from uniform distribution on (-1,1)

2.Accept if $\mathrm{s} = \mathrm{U}_1^2 + \mathrm{U}_2^2 < 1$, otherwise, get new U_1 and U_2

3.Let $\mathrm{X} = \sqrt{-2\log(\mathrm{s})/\mathrm{s}}\,\mathrm{U}_1$, $\mathrm{Y} = \sqrt{-2\log(\mathrm{s})/\mathrm{s}}\,\mathrm{U}_2$

4.It can be proved that X and Y are independent N(0,1) random variables.

Description of Method:

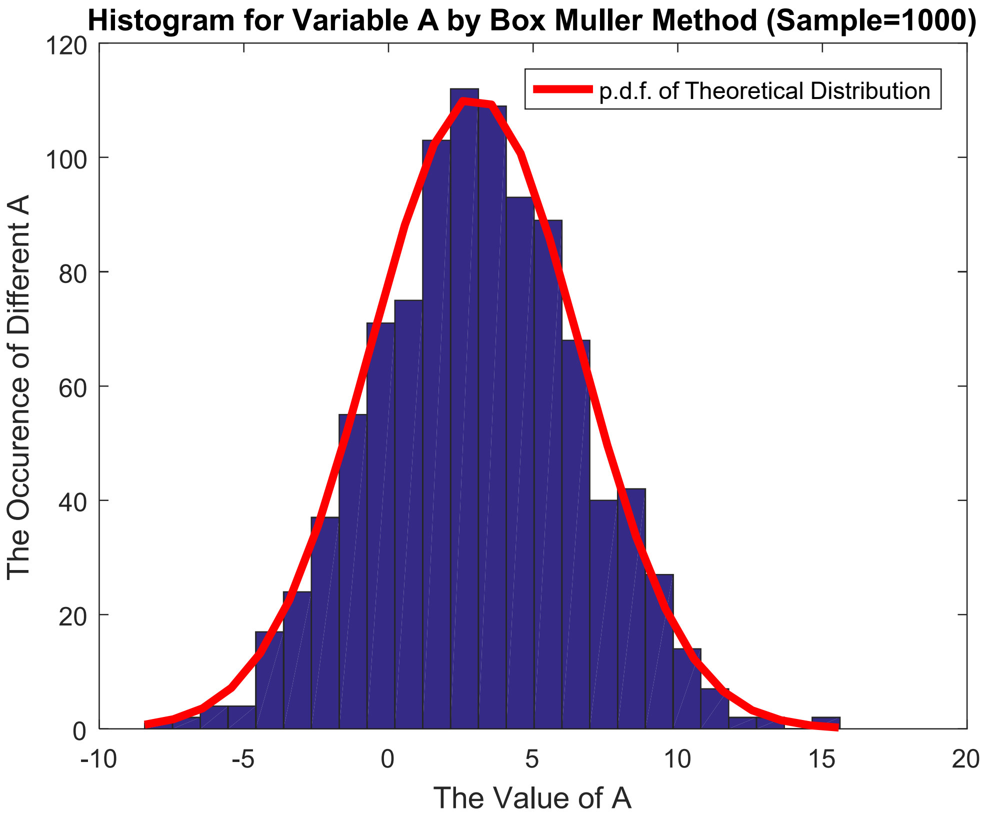
The Box-Muller method is utilized to generate 1000 samples. The sample mean and sample variance are computed. The theoretical mean and variance are compared with the experimental data. The Box-Muller method and Polar Marsaglia method are both used multiple times to generate 1000000 samples. The comparison is conducted between the computation time between two methods.

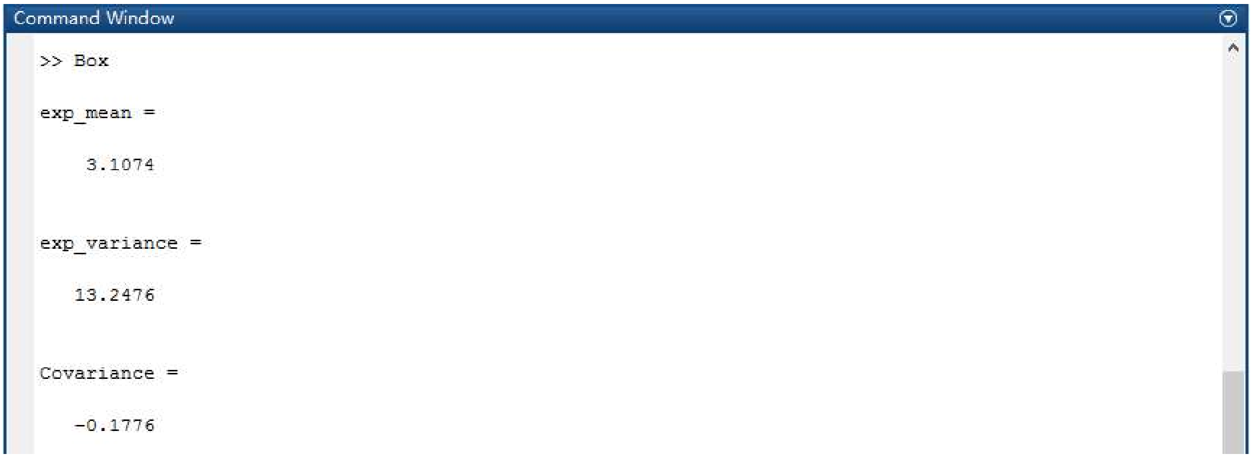
Part 1.

Code:

```
%%% Box-Muller Method
N=1000;
M1 = 1; % Mean of X
M2 = 2; % Mean of Y
V1 = 4; % Variance of X
V2 = 9; % Variance of Y

u1 = rand(N,1);
u2 = rand(N,1);

% Geberate X and Y that are N(0,1) random variables and independent
X = sqrt( - 2*log(u1)).*cos(2*pi*u2 );
Y = sqrt( - 2*log(u1)).*sin(2*pi*u2 );

% Scale them to a particular mean and variance
x = sqrt(V1)*X + M1; % x~ N(M1,V1)
y = sqrt(V2)*Y + M2; % y~ N(M2,V2)

A=x+y;
exp_mean=mean(A)
exp_variance=var(A)
temp=cov(x,y);
Covariance=temp(1,2)

i=min(A);
j=max(A);
hist(A,[j-i+1]);
hold on

t=i:j;
R=normpdf(t,3,sqrt(13));
M=R*N;
T1=plot(t,M,'-r','LineWidth', 3);
title('Histogram for Variable A');
xlabel('The Value of A');
ylabel('The Occurence of Different A');
legend([T1],'p.d.f. of Theoretical Distribution');
hold off
```

Simulation Result:





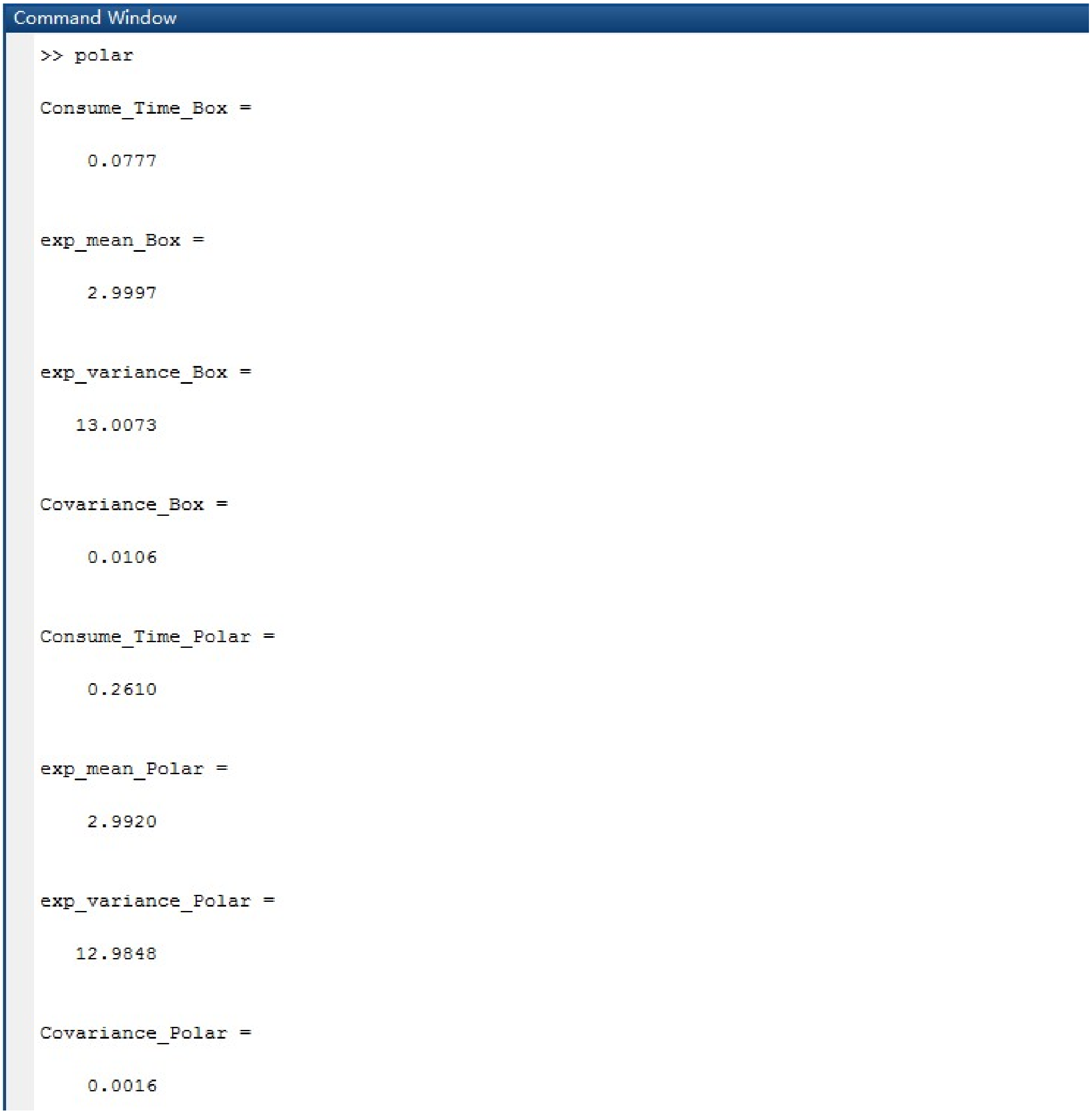
```
>> Box

exp_mean =

    3.1074


exp_variance =

   13.2476


Covariance =

   -0.1776
```

Part 2.

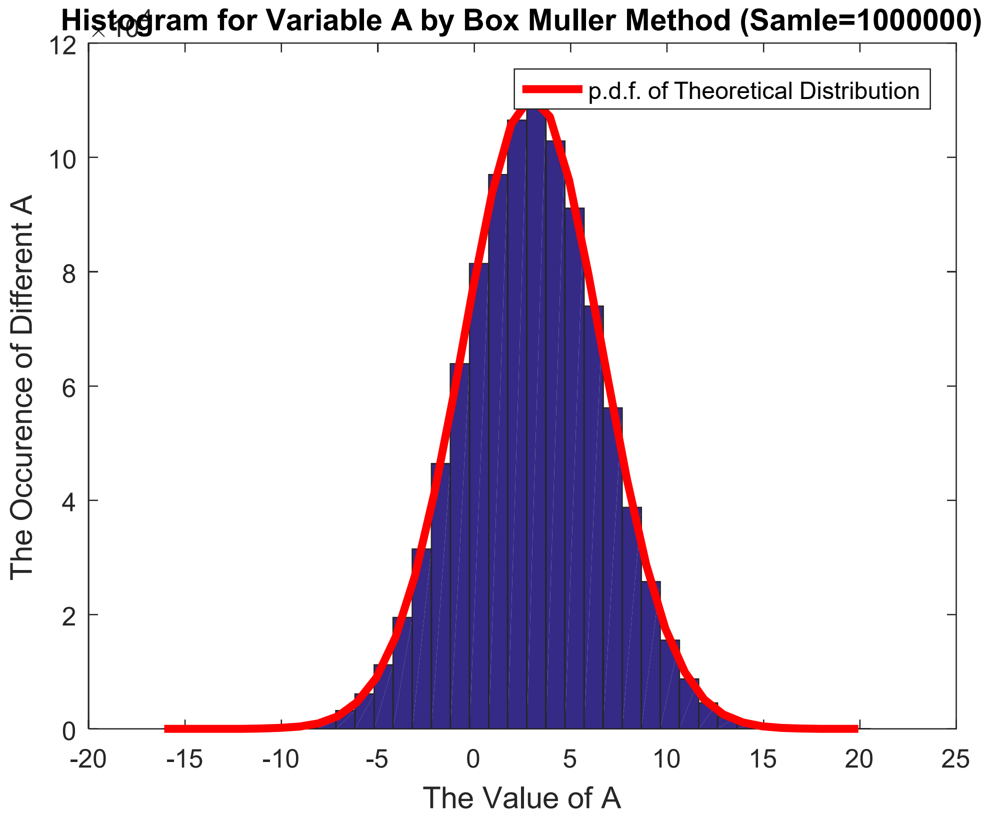
Code:

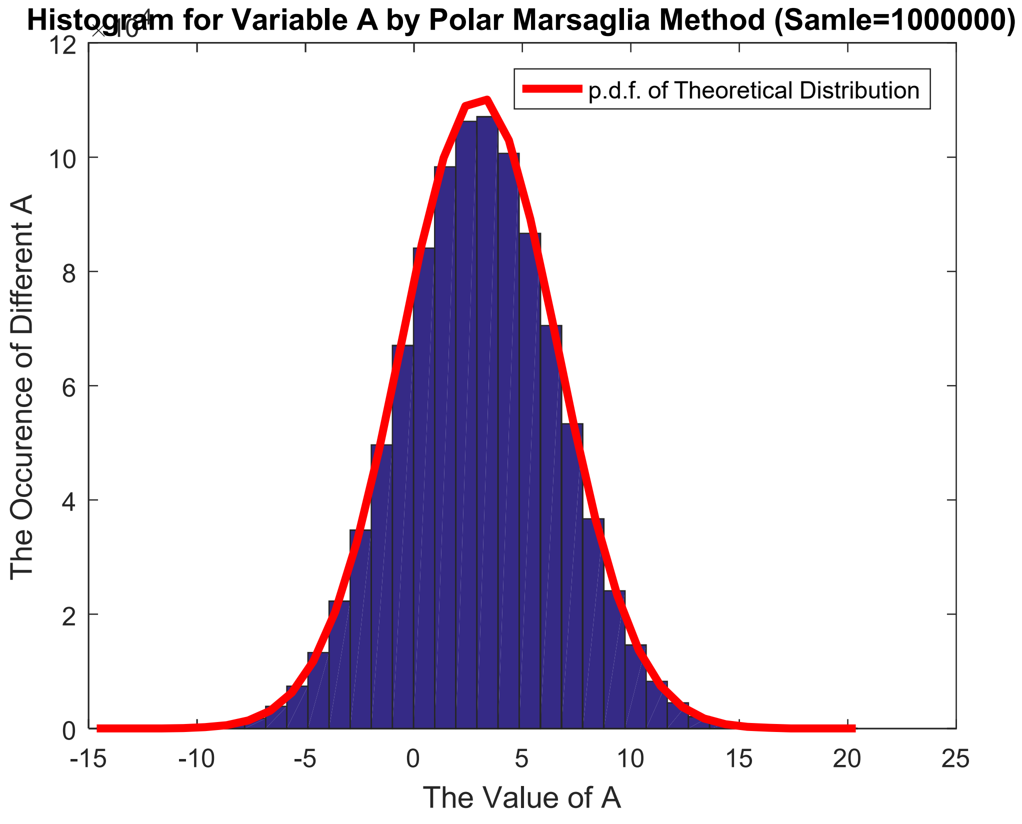
```
%%% Box-Muller Method
N=1000000;
Begin = zeros(100,1);
End = zeros(100,1);
for Loop = 1:100
    start = tic;
    Begin(Loop)=start;% Start the timer
    M1 = 1; % Mean of X
    M2 = 2; % Mean of Y
    V1 = 4; % Variance of X
    V2 = 9; % Variance of Y

    u1 = rand(N,1);
    u2 = rand(N,1);

    % Geberate X and Y that are N(0,1) random variables and independent
    X = sqrt( - 2*log(u1)).*cos(2*pi*u2 );
    Y = sqrt( - 2*log(u1)).*sin(2*pi*u2 );

    % Scale them to a particular mean and variance
    x = sqrt(V1)*X + M1; % x~ N(M1,V1)
    y = sqrt(V2)*Y + M2; % y~ N(M2,V2)


    End(Loop)=toc(start); % Read elapsed time from stopwatch
end
Consume_Time_Box=mean(End)
A=x+y;
exp_mean_Box=mean(A)
exp_variance_Box=var(A)
temp=cov(x,y);
Covariance_Box=temp(1,2)
i=min(A);
j=max(A);
figure();
hist(A,[j-i+1]);
hold on

t=i:j;
R=normpdf(t,3,sqrt(13));
M=R*N;
T1=plot(t,M,'-r','LineWidth', 3);
title('Histogram for Variable A by Box Muller Method (Samle=1000000)');
xlabel('The Value of A');
ylabel('The Occurence of Different A');
legend([T1],'p.d.f. of Theoretical Distribution');
hold off
%%%Polar Marsaglia Method
for Loop = 1:100
    start = tic;
    Begin(Loop)=start;% Start the timer
    M1 = 1; % Mean of X
    M2 = 2; % Mean of Y
    V1 = 4; % Variance of X
    V2 = 9; % Variance of Y
    i = 0; % the random number generated by the algorithm

    % Geberate X and Y that are N(0,1) random variables and independent
    while(i<=N-1)
        u1 = 2*rand()-1;
        u2 = 2*rand()-1;
        s = u1^2 + u2^2;
        if(s < 1)
            i = i + 1;
            X(i) = sqrt(-2*log(s)/s)*u1;
            Y(i) = sqrt(-2*log(s)/s)*u2;
        end
    end

    % Scale them to a particular mean and variance
    x = sqrt(V1)*X + M1; % x~ N(M1,V1)
    y = sqrt(V2)*Y + M2; % y~ N(M2,V2)
    End(Loop)=toc(start); % Read elapsed time from stopwatch
end
Consume_Time_Polar=mean(End)
A=x+y;
exp_mean_Polar=mean(A)
exp_variance_Polar=var(A)
temp=cov(x,y);
Covariance_Polar=temp(1,2)
i=min(A);
j=max(A);
figure();
hist(A,[j-i+1]);
hold on

t=i:j;
R=normpdf(t,3,sqrt(13));
M=R*N;
T1=plot(t,M,'-r','LineWidth', 3);
title('Histogram for Variable A by Polar Marsaglia Method (Samle=1000000)');
xlabel('The Value of A');
ylabel('The Occurence of Different A');
legend([T1],'p.d.f. of Theoretical Distribution');
hold off
```

Simulation Result:



```
>> polar

Consume_Time_Box =

    0.0777


exp_mean_Box =

    2.9997


exp_variance_Box =

   13.0073


Covariance_Box =

    0.0106


Consume_Time_Polar =

    0.2610


exp_mean_Polar =

    2.9920


exp_variance_Polar =

   12.9848


Covariance_Polar =

    0.0016
```


Histogram for Variable A by Box Muller Method (Samle=1000000)
$\times 10^4$
12
10
8
6
4
2
0
The Occurence of Different A
-20
-15
-10
-5
0
5
10
15
20
25
The Value of A
p.d.f. of Theoretical Distribution


Histogram for Variable A by Polar Marsaglia Method (Samle=1000000)
$\times 10^4$
12
10
8
6
4
2
0
The Occurence of Different A
-15
-10
-5
0
5
10
15
20
25
The Value of A
p.d.f. of Theoretical Distribution

Finding:

1). When 1000 samples are generated by Box-Muller method.

The covariance between X and Y is -0.1776.

The theoretical mean is 3, the sample mean is 3.1074 and they are close.

The theoretical variance is 13, the sample variance is 13.2476 and they are close.

2). When 1000000 samples are generated by Polar Marsaglia method.

The covariance between X and Y is 0.0016.

The theoretical mean is 3, the sample mean is 2.9920 and they are close.

The theoretical variance is 13, the sample variance is 12.9848 and they are close.

3). When 1000000 samples are generated by Box-Muller method.

The covariance between X and Y is 0.0106.

The theoretical mean is 3, the sample mean is 2.9997 and they are close.

The theoretical variance is 13, the sample variance is 13.0073 and they are close.

4). When 1000000 samples are generated both by Box-Muller and Polar Marsaglia methods.

The computational time for Box-Muller method is 0.0777.

The computational time for Polar Marsaglia method is 0.2610.

The computational time for Box-Muller method is less than the Polar Marsaglia methods.

Experiment No.2

Description of Algorithm:

The Accept-Reject method is utilized in the lab.

Generate random samples from density $\mathrm{f(x)}$ by sampling from $\mathrm{g(x)}$ with $\mathrm{f(x) \leq cg(x)}$ for all x and some $\mathrm{c > 0}$.

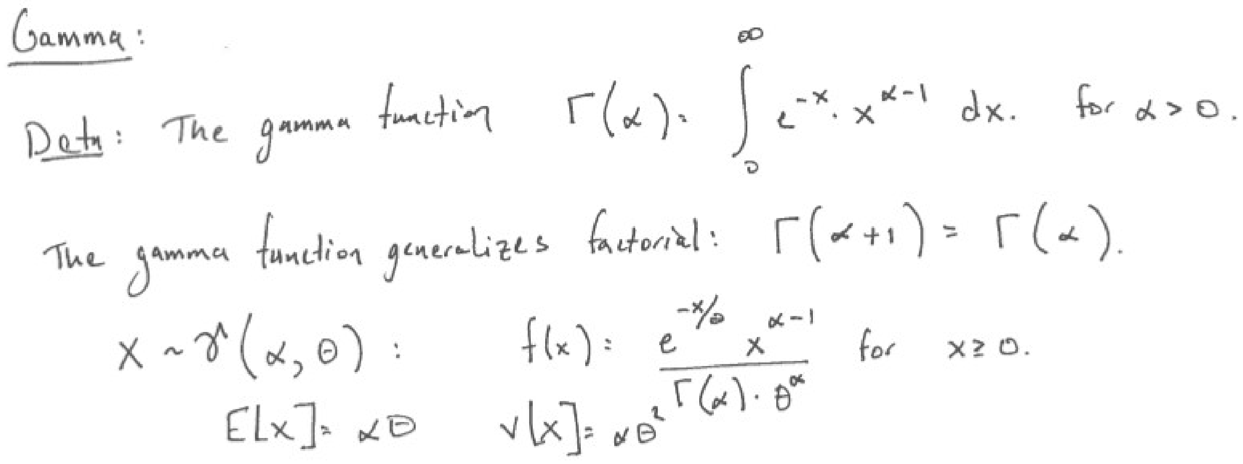
1.Generate random variable Y with density g

2.Generate u~Uniform[0,1]

3.If $\mathrm{u \leq \frac{f(Y)}{cg(Y)}}$ then X=Y, else go to step 1.

Random variable generated by accept-reject algorithm has density f, and the number of iterations of the algorithm to terminate is a geometric random variable with mean c.

The Gamma distribution is used in the experiment.



Gamma:

Defn: The gamma function $\Gamma(\alpha) = \int_0^{\infty} e^{-x} \cdot x^{\alpha-1} \, dx.$ for $\alpha > 0$.

The gamma function generalizes factorial: $\Gamma(\alpha+1) = \Gamma(\alpha).$

$X \sim \gamma(\alpha, \theta)$: $f(x) = \frac{e^{-x/\theta} \, x^{\alpha-1}}{\Gamma(\alpha) \cdot \theta^{\alpha}}$ for $x \geq 0$.

$E[x] = \alpha\theta$ $V[x] = \alpha\theta^2$

Description of Method:

In order to obtain samples from Gamma distribution, the samples are generated from the exponential distribution according to the target distribution. In order to generate samples $\{x_i\}$from the Gamma distribution, I have generated samples $\{y_i\}$ from exponential distribution and accepted it (let $x_i=y_i$) with probability $\frac{f(Y)}{cg(Y)}$.The sample acceptance rate is computed and compared with the theoretical acceptance rate.

Code:

```
clear all;
k=5.5;
theta=1;
% Sample code to find the maximum ratio c
pdfX = @(x)  ( (x.^(k-1)).*exp(-x./theta) )./( (theta.^k).*gamma(k) ); % one-
line function handle for f
pdfY = @(y) 1/k * exp(-y/k); % one-line function handle for g
t = 0:0.01:20;
ratio = pdfX(t)./pdfY(t);
c = max(ratio); % find the maximum ratio

figure(); % visualize the distributions
plot(t,pdfX(t),'linewidth',2);
hold on
plot(t,pdfY(t),'linewidth',2);
plot(t,c * pdfY(t),'r--','linewidth',2);
title('Diagram for Distributions');
xlabel('The Value of t');
ylabel('The Probability of Different t');
legend('pdf of Gamma(5.5,1)', 'pdf of Exponential(5.5)', 'c \cdot pdf of
Exponential(5.5)')
hold off

N=1000;
for i = 1:N, k = 0;
    while 1
        k = k + 1;
        u=rand;
        j=-5.5*log(u);
            if c*rand < pdfX(j)/pdfY(j) % Accept p(j) if U<p(j)/c, q(j)= 0.1
                X(i) = j;
                C(i) = k;
                break
            end
        end
end

accept_rate=N/sum(C)
figure();
hist(X,16);
hold on
T1=plot(t,pdfX(t)*N,'-r','linewidth',3);
title('Histogram for the Distribution');
xlabel('Number of Outcomes');
ylabel('Quantity of Occurence Times');
legend([T1],'p.d.f. of Theoretical Distribution');
hold off
```

Simulation Result

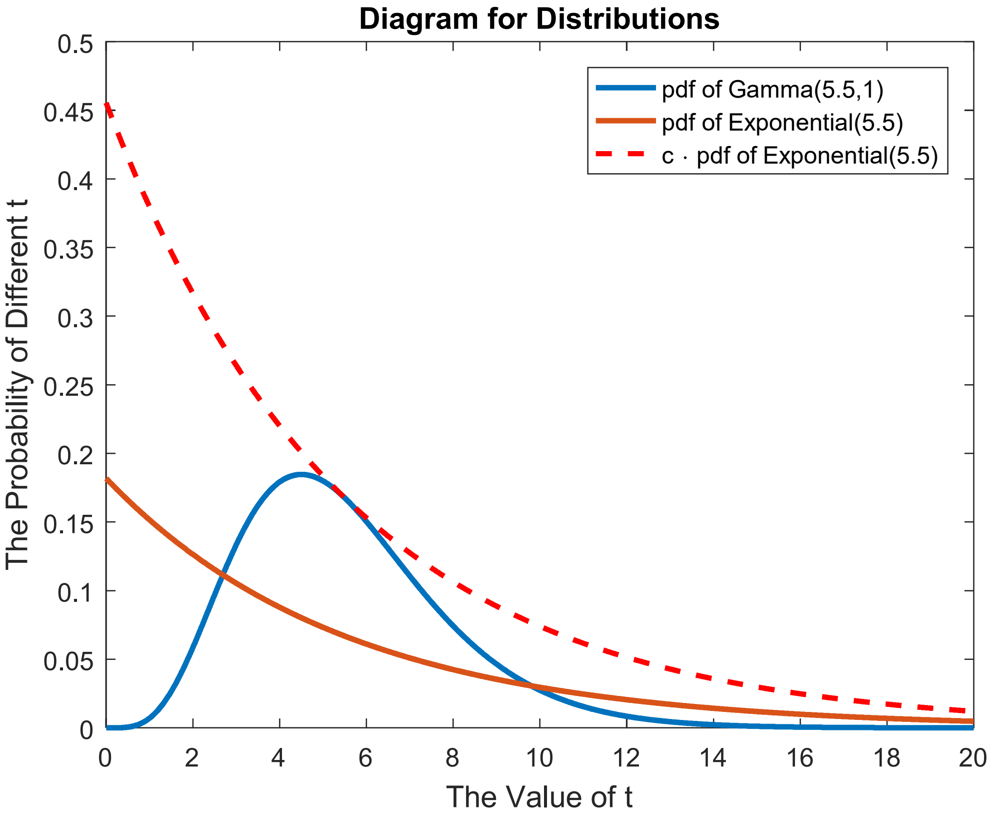
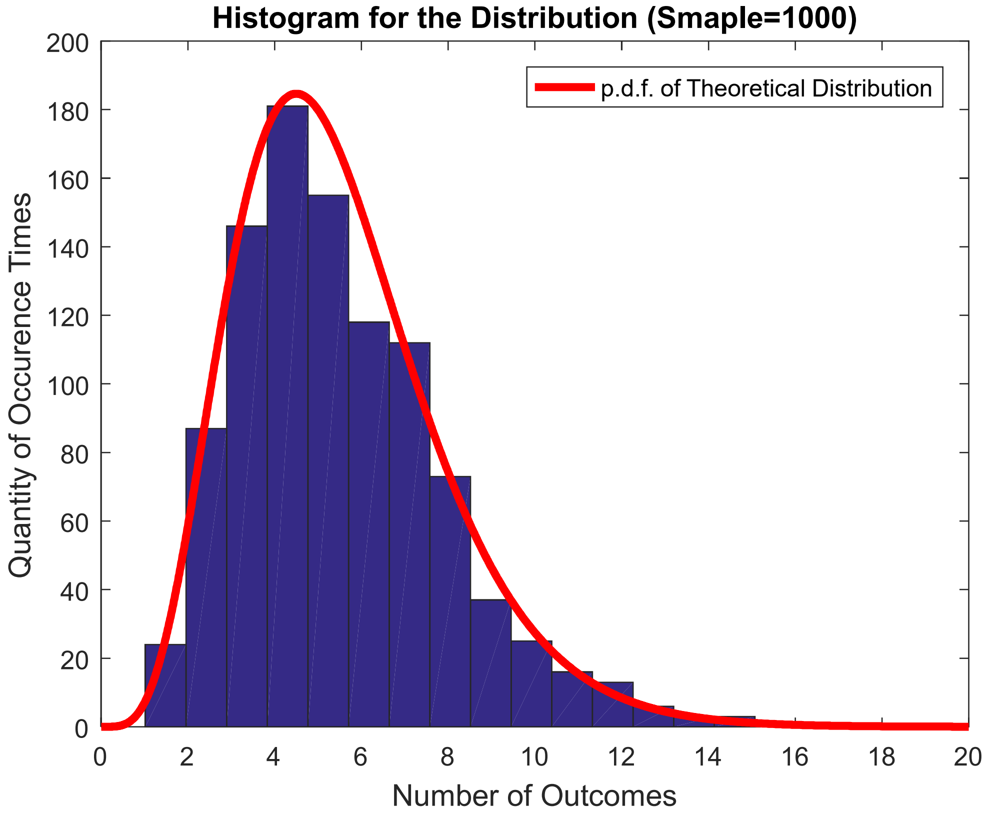
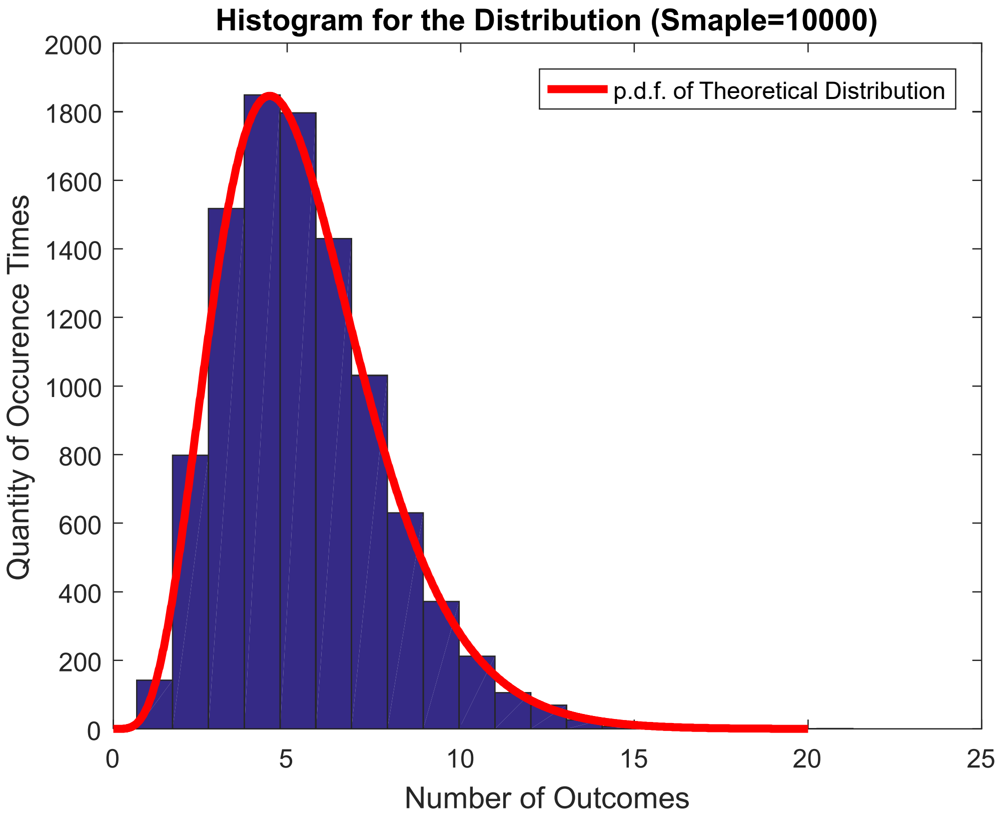
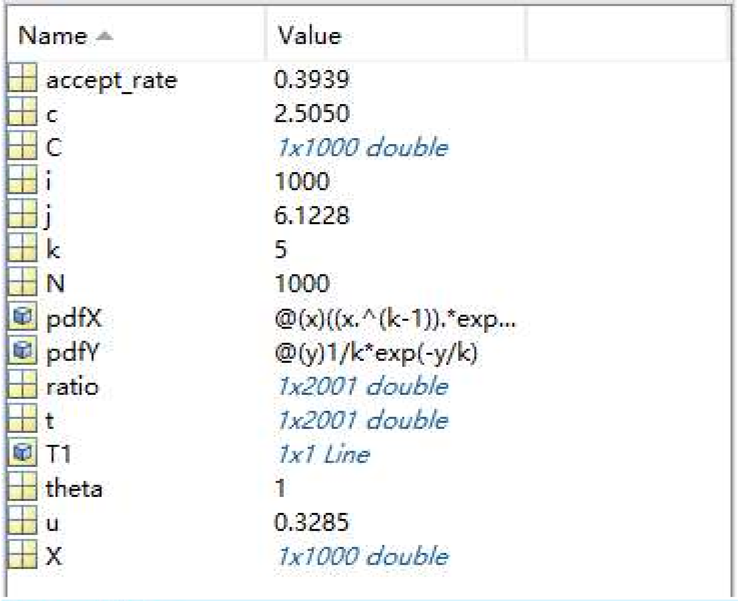


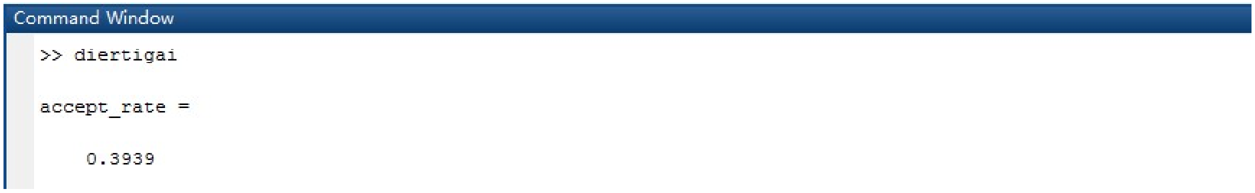
Diagram for Distributions
pdf of Gamma(5.5,1)
pdf of Exponential(5.5)
c · pdf of Exponential(5.5)
The Probability of Different t
The Value of t
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12
14
16
18
20


Histogram for the Distribution (Smaple=1000)
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
Number of Outcomes
200
180
160
140
120
100
80
60
40
20
0
0
2
4
6
8
10
12
14
16
18
20





Name ▲	Value
accept_rate	0.3939
c	2.5050
C	*1x1000 double*
i	1000
j	6.1228
k	5
N	1000
pdfX	@(x)((x.^(k-1)).*exp...
pdfY	@(y)1/k*exp(-y/k)
ratio	*1x2001 double*
t	*1x2001 double*
T1	*1x1 Line*
theta	1
u	0.3285
X	*1x1000 double*



```
>> diertigai

accept_rate =

    0.3939
```

Finding:

1). It is shown in the result that the experimental acceptance rate is 0.3939.

The theoretical acceptance rate is

$${}^{1}/_{\mathrm{c}} = 0.4$$

The experimental acceptance rate is close to the acceptance rate.

2). It is shown in the figure that the p.d.f. of theoretical distribution matches the histograms of experimental distribution, and the matching becomes better as the number of sample increases.

Experiment No.3

Description of Algorithm:

The alpha-stable distribution is utilized in this lab.

The alpha-stable distribution is a four-parameter family of distributions and is (usually) denoted by $S(\alpha, \beta, \gamma, \delta)$. Then $\alpha \in (0, 2]$ describes the tail of the distribution. $\beta \in [-1, 1]$ specifies if the distribution is right- ($\beta > 0$) or left- ($\beta < 0$) skewed. $\gamma > 0$ defines the scale and $\delta \in R$ specifies the location.

Let $X \sim S(\alpha, \beta, \gamma, \delta)$. The alpha-stable distribution function can be defined.

For α ≠1, we have

$$\phi(t) = E\exp(itX) = \exp\left(-\gamma^\alpha|t|^\alpha[1 - i\beta sign(t)\tan(\tfrac{\pi\alpha}{2})] + i\delta t\right).$$

and for α =1,

$$\phi(t) = E\exp(itX) = \exp\left(-\gamma|t|(1 + i\beta sign(t)\tfrac{2}{\pi}\log|t|) + i\delta t\right).$$

Description of Method:

The stblrnd function is defined to obtain random numbers form the alpha-stable distribution. The stblpdf function is defined to generate theoretical p.d.f. of alpha-stable distribution. Two functions are called to simulate alpha-stable distribution under different values of parameter alpha and beta. Eight histograms are generated and overlayed with the corresponding theoretical alpha-stable pdf. The figures of time series are plotted for each values of parameter alpha and bete and the results are analyzed to comment on the sample magnitude as a function of alpha.

Code:

```
close all, clear all;
alpha=[0.5,1,1.8,2.0];
beta=[0,0.75];
gamma=1;
delta=0;
for i=1:4
    for j=1:2
        X = stblrnd(alpha(i),beta(j),gamma,delta,1000,1);
        figure();
        subplot(2,1,1);
        histogram(X,'BinLimits',[-10,10])
        axis([-5 5 0 1000])
        hold on;
        x=-5:0.01:5;
        Y = stblpdf(x,alpha(i),beta(j),gamma,delta);
        T1= plot(x,Y*1000,'r','linewidth',2);
        hold off;
        title(['This is figure for alpha=' num2str(alpha(i)) ' and beta='
num2str(beta(j))]);
        xlabel('The Value of Random Variables');
        ylabel('Quantity of Occurence Times');
        legend([T1],'p.d.f. of Theoretical Distribution');
        subplot(2,1,2);
        t=1:1000;
        plot(t,X);
        title(['Time Series Plot']);
        xlabel('The Sample Numbers');
        ylabel('The outcome Values');
    end
end


function r = stblrnd(alpha,beta,gamma,delta,varargin)

if nargin < 4
    error('stats:stblrnd:TooFewInputs','Requires at least four input
arguments.');
end

% Check parameters
if alpha <= 0 || alpha > 2 || ~isscalar(alpha)
    error('stats:stblrnd:BadInputs',' "alpha" must be a scalar which lies in
the interval (0,2]');
end
if abs(beta) > 1 || ~isscalar(beta)
    error('stats:stblrnd:BadInputs',' "beta" must be a scalar which lies in
the interval [-1,1]');
end
if gamma < 0 || ~isscalar(gamma)
    error('stats:stblrnd:BadInputs',' "gamma" must be a non-negative
scalar');
end
if ~isscalar(delta)
    error('stats:stblrnd:BadInputs',' "delta" must be a scalar');
end



% Get output size
[err, sizeOut] = genOutsize(4,alpha,beta,gamma,delta,varargin{:});
if err > 0
    error('stats:stblrnd:InputSizeMismatch','Size information is
inconsistent.');
end



if alpha == 2                  % Gaussian distribution
    r = sqrt(2) * randn(sizeOut);
```

```
elseif alpha==1 && beta == 0   % Cauchy distribution
    r = tan( pi/2 * (2*rand(sizeOut) - 1) );

elseif alpha == .5 && abs(beta) == 1 % Levy distribution (a.k.a. Pearson V)
    r = beta ./ randn(sizeOut).^2;

elseif beta == 0                % Symmetric alpha-stable
    V = pi/2 * (2*rand(sizeOut) - 1);
    W = -log(rand(sizeOut));
    r = sin(alpha * V) ./ ( cos(V).^(1/alpha) ) .* ...
        ( cos( V.*(1-alpha) ) ./ W ).^( (1-alpha)/alpha );

elseif alpha ~= 1                % General case, alpha not 1
    V = pi/2 * (2*rand(sizeOut) - 1);
    W = - log( rand(sizeOut) );
    const = beta * tan(pi*alpha/2);
    B = atan( const );
    S = (1 + const * const).^(1/(2*alpha));
    r = S * sin( alpha*V + B ) ./ ( cos(V) ).^(1/alpha) .* ...
        ( cos( (1-alpha) * V - B ) ./ W ).^((1-alpha)/alpha);

else                             % General case, alpha = 1
    V = pi/2 * (2*rand(sizeOut) - 1);
    W = - log( rand(sizeOut) );
    piover2 = pi/2;
    sclshftV =  piover2 + beta * V ;
    r = 1/piover2 * ( sclshftV .* tan(V) - ...
        beta * log( (piover2 * W .* cos(V) ) ./ sclshftV ) );

end

% Scale and shift
if alpha ~= 1
    r = gamma * r + delta;
else
    r = gamma * r + (2/pi) * beta * gamma * log(gamma) + delta;
end

end

function [err, commonSize, numElements] = genOutsize(nparams,varargin)
try
    tmp = 0;
    for argnum = 1:nparams
        tmp = tmp + varargin{argnum};
    end
    if nargin > nparams+1
        tmp = tmp + zeros(varargin{nparams+1:end});
    end
    err = 0;
    commonSize = size(tmp);
    numElements = numel(tmp);

catch
```

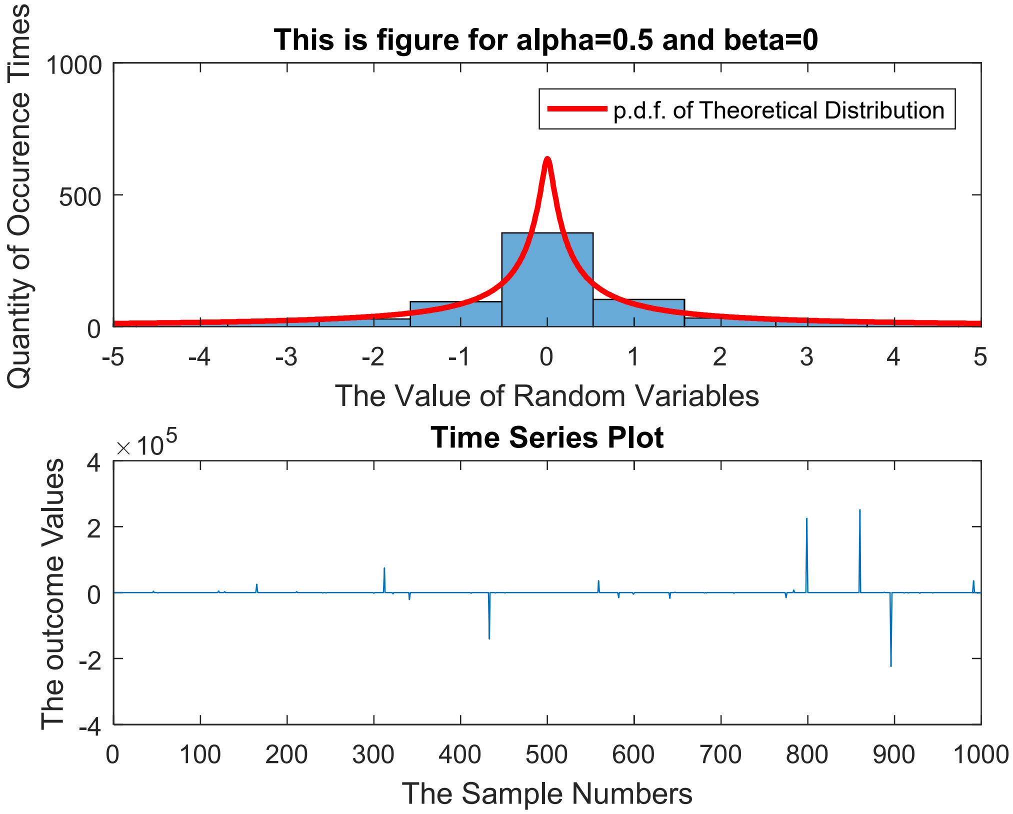
```
    err = 1;
    commonSize = [];
    numElements = 0;
end

end


function p = stblpdf(x,alpha,beta,gam,delta,varargin)

if nargin < 5
    error('stblpdf:TooFewInputs','Requires at least five input arguments.');
end

% Check parameters
if alpha <= 0 || alpha > 2 || ~isscalar(alpha)
    error('stblpdf:BadInputs',' "alpha" must be a scalar which lies in the
interval (0,2]');
end
if abs(beta) > 1 || ~isscalar(beta)
    error('stblpdf:BadInputs',' "beta" must be a scalar which lies in the
interval [-1,1]');
end
if gam < 0 || ~isscalar(gam)
    error('stblpdf:BadInputs',' "gam" must be a non-negative scalar');
end
if ~isscalar(delta)
    error('stblpdf:BadInputs',' "delta" must be a scalar');
end

% Warn if alpha is very close to 1 or 0
if ( 1e-5 < abs(1 - alpha) && abs(1 - alpha) < .02) || alpha < .02
    warning('stblpdf:ScaryAlpha',...
        'Difficult to approximate pdf for alpha close to 0 or 1')
end

% warnings will happen during call to QUADV, and it's okay
warning('off');

% Check and initialize additional inputs
quick = false;
tol = [];
for i=1:length(varargin)
    if strcmp(varargin{i},'quick')
        quick = true;
    elseif islogical(varargin{i})
        quick = varargin{end};
    elseif isscalar(varargin{i})
        tol = varargin{i};
    end
end

if isempty(tol)
    if quick
        tol = 1e-8;
    else
        tol = 1e-12;
    end
end

% Check to see if you are in a simple case, if so be quick, if not do
% general algorithm
if alpha == 2                  % Gaussian distribution
    x = (x - delta)/gam;                 % Standardize
    p = 1/sqrt(4*pi) * exp( -.25 * x.^2 ); % ~ N(0,2)
    p = p/gam; %rescale

elseif alpha==1 && beta == 0   % Cauchy distribution
    x = (x - delta)/gam;              % Standardize
    p = (1/pi) * 1./(1 + x.^2);
    p = p/gam; %rescale

elseif alpha == .5 && abs(beta) == 1 % Levy distribution
    x = (x - delta)/gam;              % Standardize
    p = zeros(size(x));
    if  beta ==1
        p( x <= 0 ) = 0;
        p( x > 0 ) = sqrt(1/(2*pi)) * exp(-.5./x(x>0)) ./...
                                                x(x>0).^1.5;
    else
        p(x >= 0) = 0;
        p(x < 0 ) = sqrt(1/(2*pi)) * exp(.5./x(x<0)  ) ./...
                                            ( -x(x<0) ).^1.5;
    end
    p = p/gam; %rescale

elseif abs(alpha - 1) > 1e-5          % Gen. Case, alpha ~= 1

    xold = x; % Save for later
    % Standardize in (M) parameterization ( See equation (2) in [1] )
    x = (x - delta)/gam - beta * tan(alpha*pi/2);

    % Compute pdf
    p = zeros(size(x));
    zeta = - beta * tan(pi*alpha/2);
    theta0 = (1/alpha) * atan(beta*tan(pi*alpha/2));
    A1 = alpha*theta0;
    A2 = cos(A1)^(1/(alpha-1));
    exp1 = alpha/(alpha-1);
    alpham1 = alpha - 1;
    c2 = alpha ./ (pi * abs(alpha - 1) * ( x(x>zeta) - zeta) );
    V = @(theta) A2 * ( cos(theta) ./ sin( alpha*(theta +
theta0) ) ).^exp1.*...
        cos( A1 + alpham1*theta ) ./ cos(theta);


    % x > zeta, calculate integral using QUADV
    if any(x(:) > zeta)
        xshift = (x(x>zeta) - zeta) .^ exp1;
        if beta == -1 && alpha < 1
            p(x > zeta) = 0;
        elseif ~quick % Locate peak in integrand and split up integral
            g = @(theta) xshift(:) .* V(theta) - 1;
            R = repmat([-theta0, pi/2 ],numel(xshift),1);
            if abs(beta) < 1
                theta2 = bisectionSolver(g,R,alpha);
            else
                theta2 = bisectionSolver(g,R,alpha,beta,xshift);
            end
            theta2 = reshape(theta2,size(xshift));
            % change variables so the two integrals go from
            % 0 to 1/2 and 1/2 to 1.
            theta2shift1 = 2*(theta2 + theta0);
            theta2shift2 = 2*(pi/2 - theta2);
            g1 = @(theta)  xshift .* ...
                V(theta2shift1 * theta - theta0);
            g2 = @(theta)  xshift .* ...
                V(theta2shift2 * (theta - .5) + theta2);
            zexpz = @(z) max(0,z .* exp(-z)); % use max incase of NaN

            p(x > zeta) = c2 .* ...
                (theta2shift1 .* quadv(@(theta) zexpz( g1(theta) ),...
                                        0 , .5, tol) ...
               + theta2shift2 .* quadv(@(theta) zexpz( g2(theta) ),...
                                       .5 , 1, tol) );

        else  % be quick - calculate integral without locating peak
              % Use a default tolerance of 1e-6
            g = @(theta) xshift * V(theta);
            zexpz = @(z) max(0,z .* exp(-z)); % use max incase of NaN
            p( x > zeta ) = c2 .* quadv(@(theta) zexpz( g(theta) ),...
                                        -theta0 , pi/2, tol );
        end
        p(x > zeta) = p(x>zeta)/gam; %rescale

    end

    % x = zeta, this is easy
    if any( abs(x(:) - zeta) < 1e-8 )
        p( abs(x - zeta) < 1e-8 ) = max(0,gamma(1 + 1/alpha)*...
            cos(theta0)/(pi*(1 + zeta^2)^(1/(2*alpha))));
        p( abs(x - zeta) < 1e-8 ) = p( abs(x - zeta) < 1e-8 )/gam; %rescale

    end

    % x < zeta, recall function with -xold, -beta, -delta
    % This doesn't need to be rescaled.
    if any(x(:) < zeta)
        p( x < zeta ) = stblpdf( -xold( x<zeta ),alpha,-beta,...
                        gam , -delta , tol , quick);
    end

else                    % Gen case, alpha = 1
```

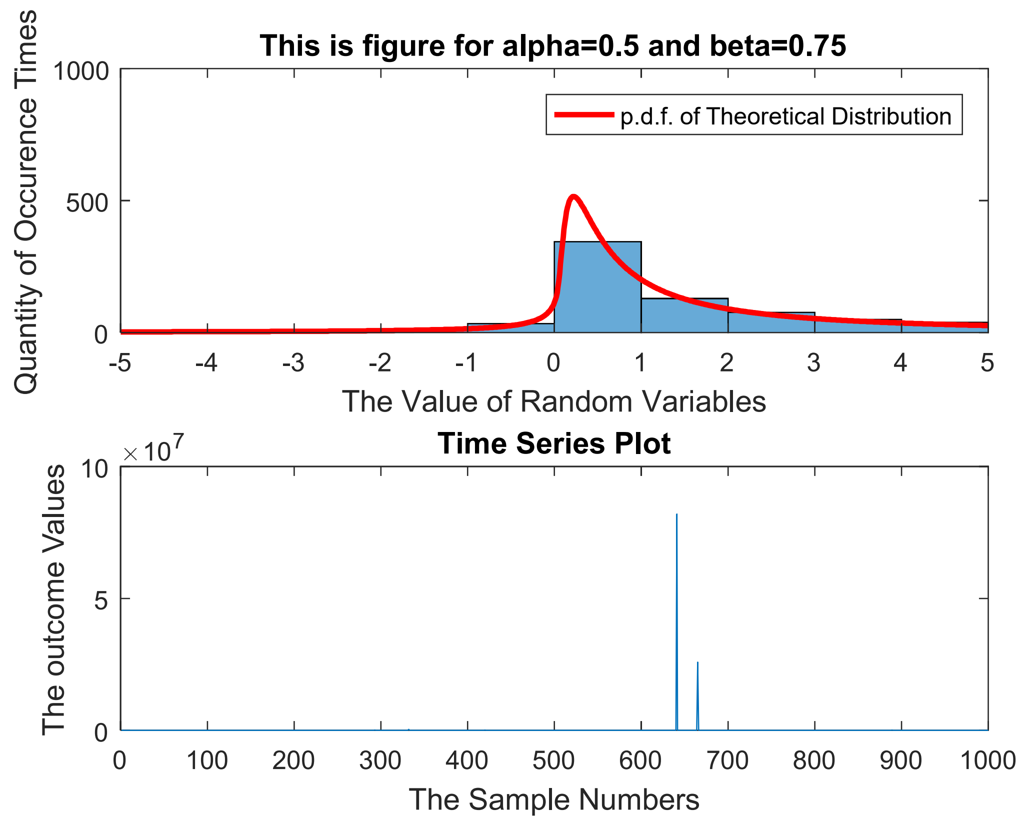
```
    x = (x - (2/pi) * beta * gam * log(gam) - delta)/gam; % Standardize

    % Compute pdf
    piover2 = pi/2;
    twooverpi = 2/pi;
    oneoverb = 1/beta;
    theta0 = piover2;
    % Use logs to avoid overflow/underflow
    logV = @(theta) log(twooverpi * ((piover2 + beta *theta)./cos(theta)))
+ ...
                  ( oneoverb * (piover2 + beta *theta) .* tan(theta) );
    c2 = 1/(2*abs(beta));
    xterm = ( -pi*x/(2*beta));

    if ~quick  % Locate peak in integrand and split up integral
             % Use a default tolerance of 1e-12
        logg = @(theta) xterm(:) + logV(theta) ;
        R = repmat([-theta0, pi/2 ],numel(xterm),1);
        theta2 = bisectionSolver(logg,R,1-beta);
        theta2 = reshape(theta2,size(xterm));
        % change variables so the two integrals go from
        % 0 to 1/2 and 1/2 to 1.
        theta2shift1 = 2*(theta2 + theta0);
        theta2shift2 = 2*(pi/2 - theta2);
        logg1 = @(theta)  xterm + ...
            logV(theta2shift1 * theta - theta0);
        logg2 = @(theta)  xterm + ...
            logV(theta2shift2 * (theta - .5) + theta2);
        zexpz = @(z) max(0,exp(z) .* exp(-exp(z))); % use max incase of NaN

        p = c2 .* ...
            (theta2shift1 .* quadv(@(theta) zexpz( logg1(theta) ),...
                                    0 , .5, tol) ...
           + theta2shift2 .* quadv(@(theta) zexpz( logg2(theta) ),...
                                   .5 , 1, tol) );


    else % be quick - calculate integral without locating peak
              % Use a default tolerance of 1e-6
        logg = @(theta) xterm + logV(theta);
        zexpz = @(z) max(0,exp(z) .* exp(-exp(z))); % use max incase of NaN
        p = c2 .* quadv(@(theta) zexpz( logg(theta) ),-theta0 , pi/2, tol );

    end

    p = p/gam; %rescale

end

p = real(p); % just in case a small imaginary piece crept in
             % This might happen when (x - zeta) is really small

end
```

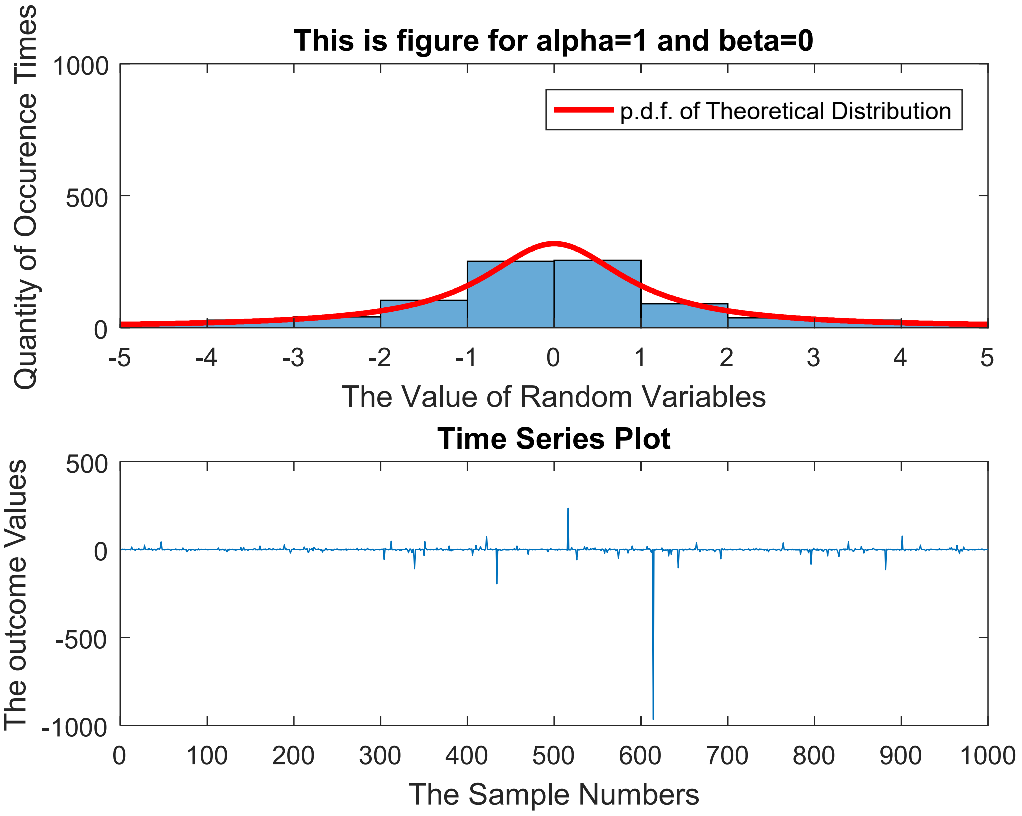
```
function X = bisectionSolver(f,R,alpha,varargin)

if nargin < 2
    error('bisectionSolver:TooFewInputs','Requires at least two input
arguments.');
end

noSolution = false(size(R,1));

tol = 1e-6;
maxiter = 30;

[N M] = size(R);
if M ~= 2
    error('bisectionSolver:BadInput',...
        '"R" must have 2 columns');
end

a = R(:,1);
b = R(:,2);
X = (a+b)/2;

try
    val = f(X);
catch ME
    error('bisectionSolver:BadInput',...
        'Input function inconsistint with rectangle dimension')
end

if size(val,1) ~= N
    error('bisectionSolver:BadInput',...
        'Output of function must be a column vector with dimension of
input');
end

% Main loop
val = inf;
iter = 0;

while( max(abs(val)) > tol && iter < maxiter )
    X = (a + b)/2;
    val = f(X);
    l = (val > 0);
    if alpha > 1
        l = 1-l;
    end
    a = a.*l + X.*(1-l);
    b = X.*l + b.*(1-l);
    iter = iter + 1;
end
```

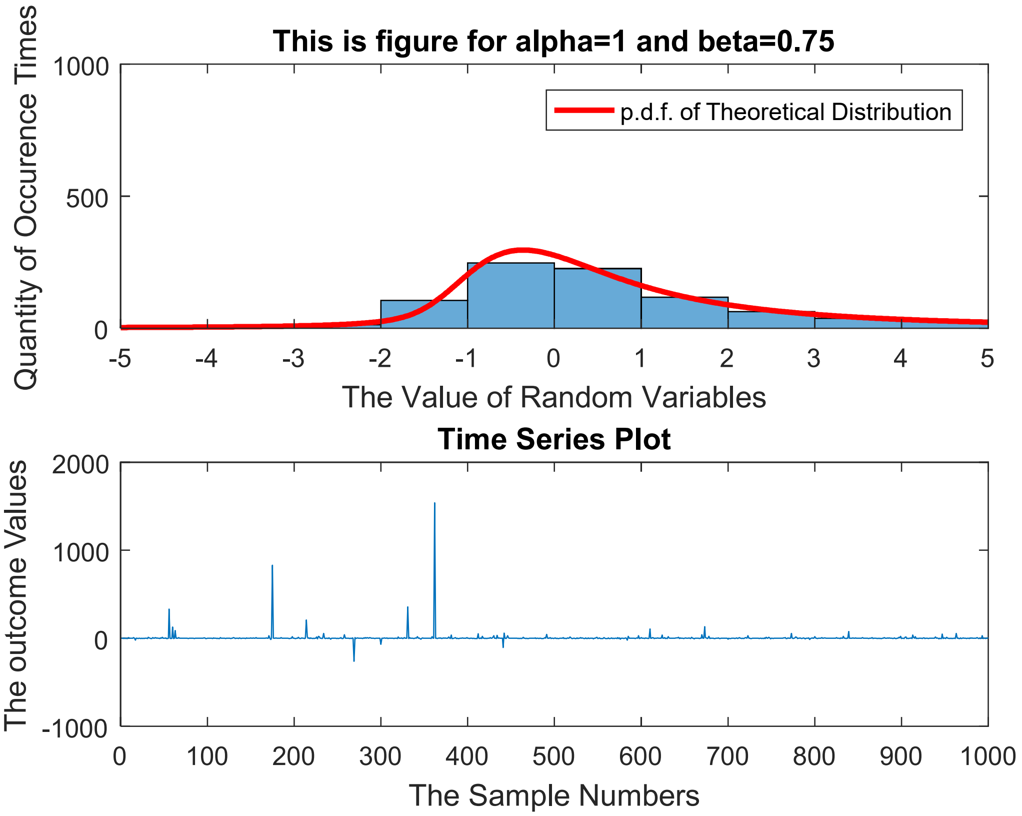
```
if any(noSolution(:))
    X(noSolution) = (R(1,1) + R(1,2))/2;
end

end
```

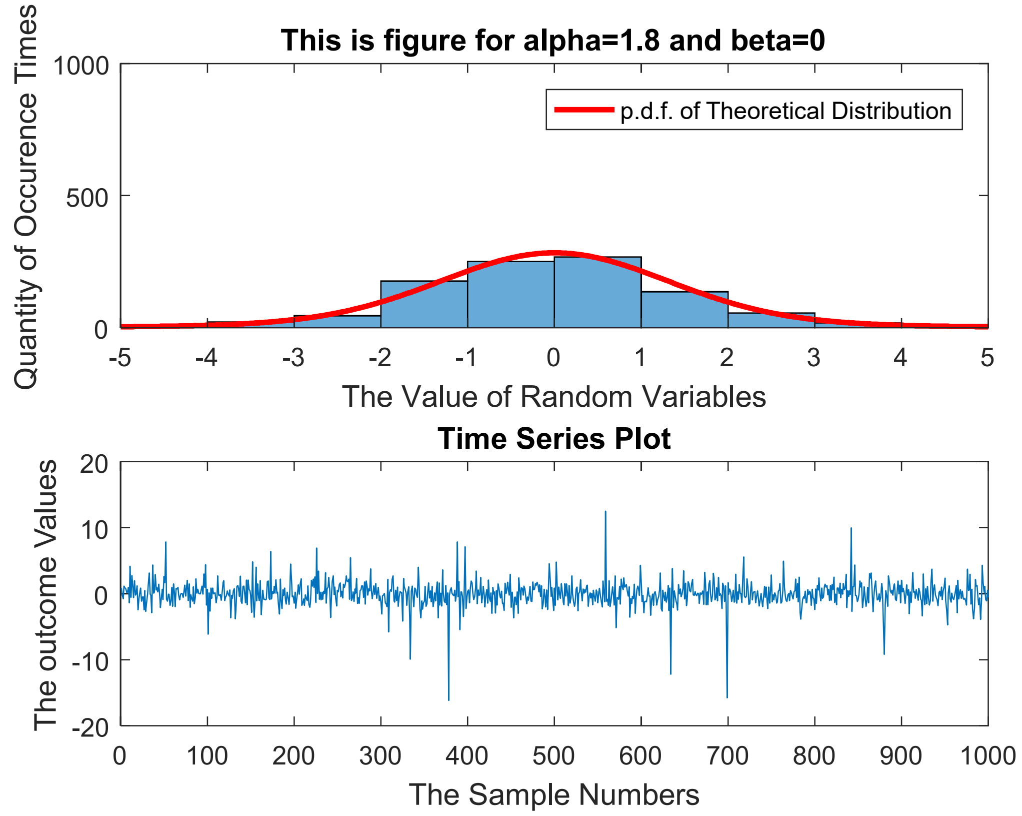
Simulation Result

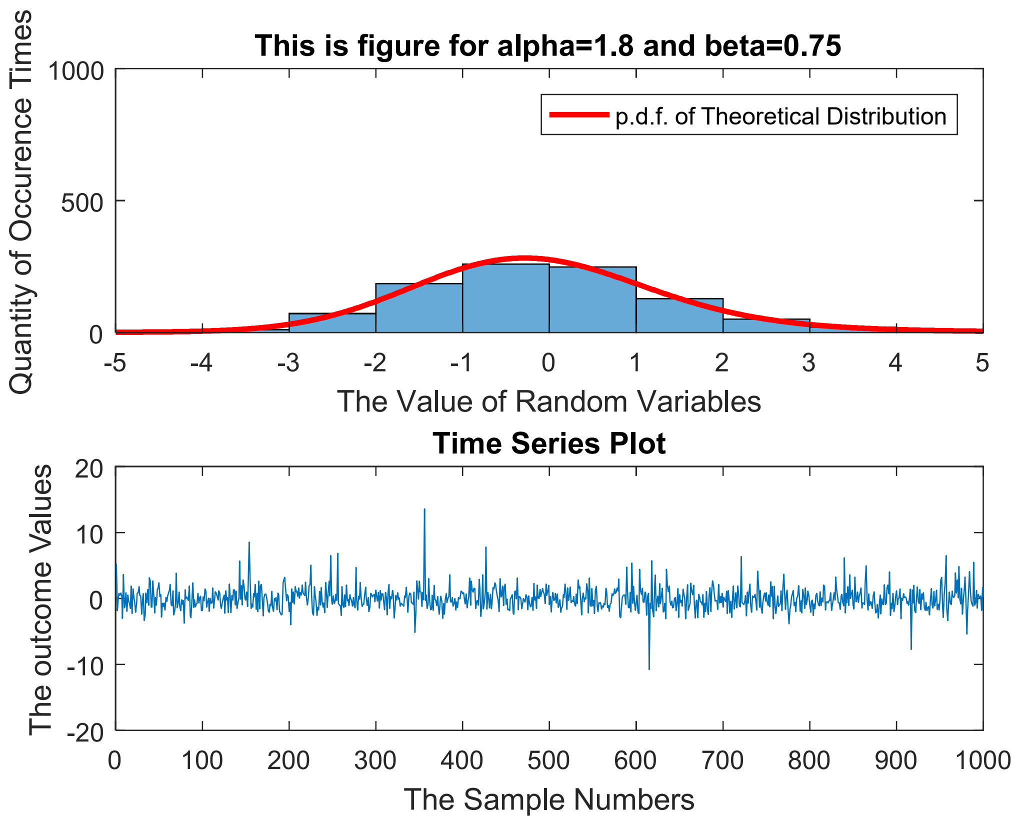


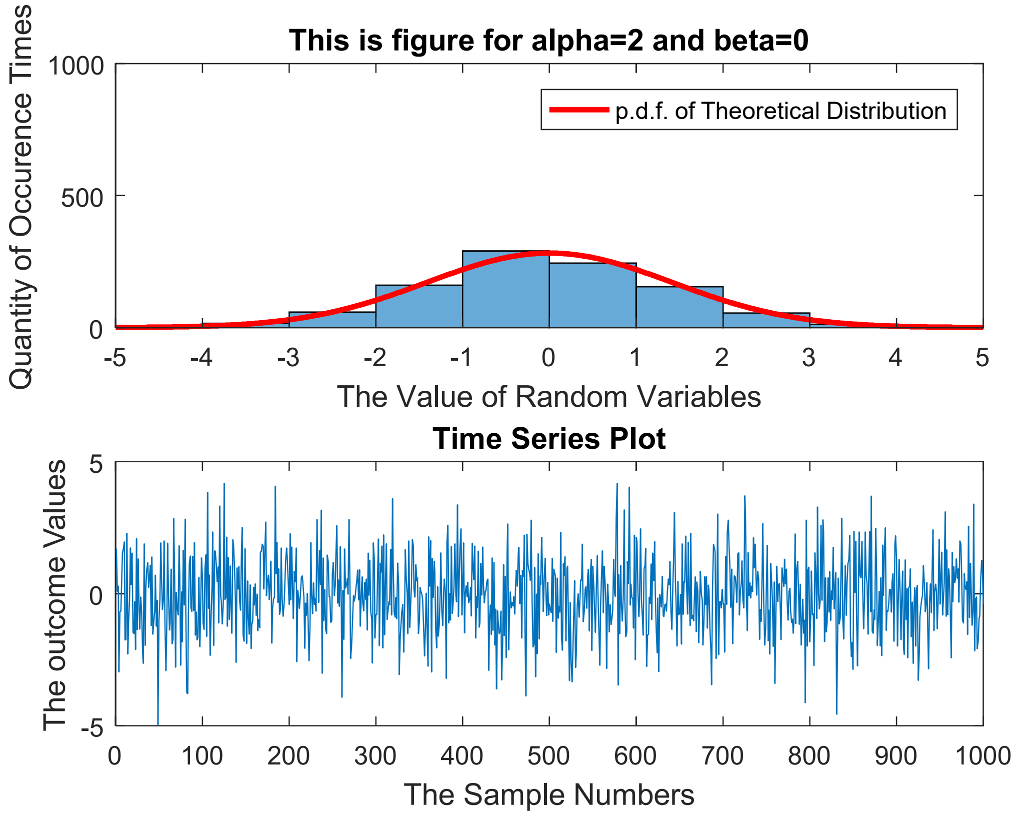

This is figure for alpha=0.5 and beta=0.75
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
×10^7
10
5
0
The outcome Values
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers

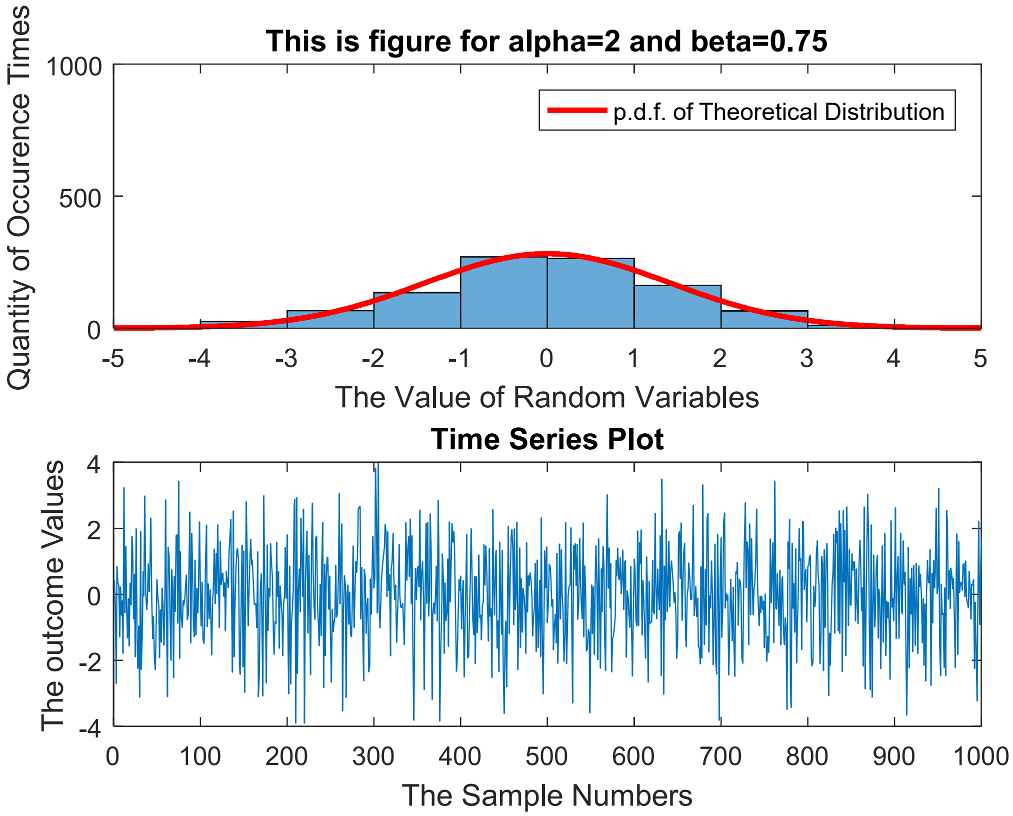

This is figure for alpha=1 and beta=0
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
500
0
-500
-1000
The outcome Values
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers


This is figure for alpha=1 and beta=0.75
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
The outcome Values
2000
1000
0
-1000
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers


This is figure for alpha=1.8 and beta=0
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
The outcome Values
20
10
0
-10
-20
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers


This is figure for alpha=1.8 and beta=0.75
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
The outcome Values
20
10
0
-10
-20
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers


This is figure for alpha=2 and beta=0
p.d.f. of Theoretical Distribution
Quantity of Occurence Times
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
The Value of Random Variables
Time Series Plot
The outcome Values
5
0
-5
0
100
200
300
400
500
600
700
800
900
1000
The Sample Numbers



Finding:

It is shown in the results that when the value of alpha is 0.5, a lot of samples are falling near the origin and a few rare sample which are large in magnitude. When the value of alpha increases, more samples are falling in the interval -5 to 0 and 0 to 5. When the value of alpha is 2, a lot of samples are falling in the interval -5 to 5.

Conclusion

Overall, the simulations of three experiments focus on continuous random variables. The Box Muller and Polar Marsaglia simulations are completed in the first experiment to generate normal distributed random variables. The second trial generates the Gamma random variables using an accept-reject method and conduct comparison with the theoretical probability density function. The third experiment simulates the alpha-stable distribution using different values of parameter and commenting on the magnitude as a function of alpha. In conclusion, the project uses mathematical methods to process independent samples, and solve continuous random variable problems.