MA 226
Monte Carlo Simulation
Assignment - 4

Amit Mittal
11012304
Department of Mathematics
I.I.T. Guwahati
Date:-04-02-2013

Assignments:-

1. ***Simulate 5000 sample of exponential with mean 5. Draw the histogram and the calculate the mean, maximum and minimum. (Use R and C/C++)***

2. ***Generate 5000 sample from Gamma with parameter n = 5 and λ = 5. Draw the histogram and the calculate the mean, maximum and minimum.***
   (Use R and C/C++)

3. ***Use the rejection method to generate from***
   $$f(x) = 20x(1 - x)^3$$
   (Use R)

1.Solution:-

C++ code:

“

```
#include <iostream>
#include <cstdio>
#include <cmath>

#define LL long long

using namespace std;

double func_rand(LL int val, LL int m){
	return (double)val/m;
}

LL int func_val(LL int val, LL int a, LL int b, LL int m){
	LL next_val;
	next_val = ((a * val) + b) % m;

	return next_val;
}

void function(LL int initial, LL int a, LL int b, LL int m, int end, double array[]){
	LL int val, count = 0;
	LL int index, quot;
	double random, prev;

	val = initial;
	random = func_rand(val, m);

	while(count<end){
		val = func_val(val, a, b, m);
		random = func_rand(val, m);

		array[count] = random;
		++count;
	}

	for(index = 0 ; index<20 ; ++index){
		printf("\"%.2lf\", %lld\n", 0.05*(index+1), array[index]);
	}
```

```
}

void exponential(double lambda, double array[], LL int bound){
	int index;
	double quot, min, max, mean, sum = 0.0;
	LL int count[100] = {0};

	min = max = array[0];
	for(index=0; index<bound; ++index){
		array[index] = -(double)log(1.0 - array[index])/lambda;
		printf("%lf\n", array[index]);

		++count[(int)array[index]];
		sum+=array[index];

		if(array[index] < min)
			min = array[index];

		if(array[index] > max)
			max = array[index];
	}

	mean = (double)sum/bound;

	printf("Min=%lf\nMax=%lf\nMean=%lf\n", min, max, mean);

	for(index = 0; index<=(int)max; ++index){
		printf("\"%d\" , %lld\n", index, count[index]);
	}
}

int main(){
	LL int index;
	double array[5005];
	LL int init_value = 3452;
	LL int bound = 5000;
	double lambda = 0.2;
	function(init_value, 4696, 0, 5003, bound, array);
	exponential(lambda, array, bound);
	return 0;
}
”
```

R code:

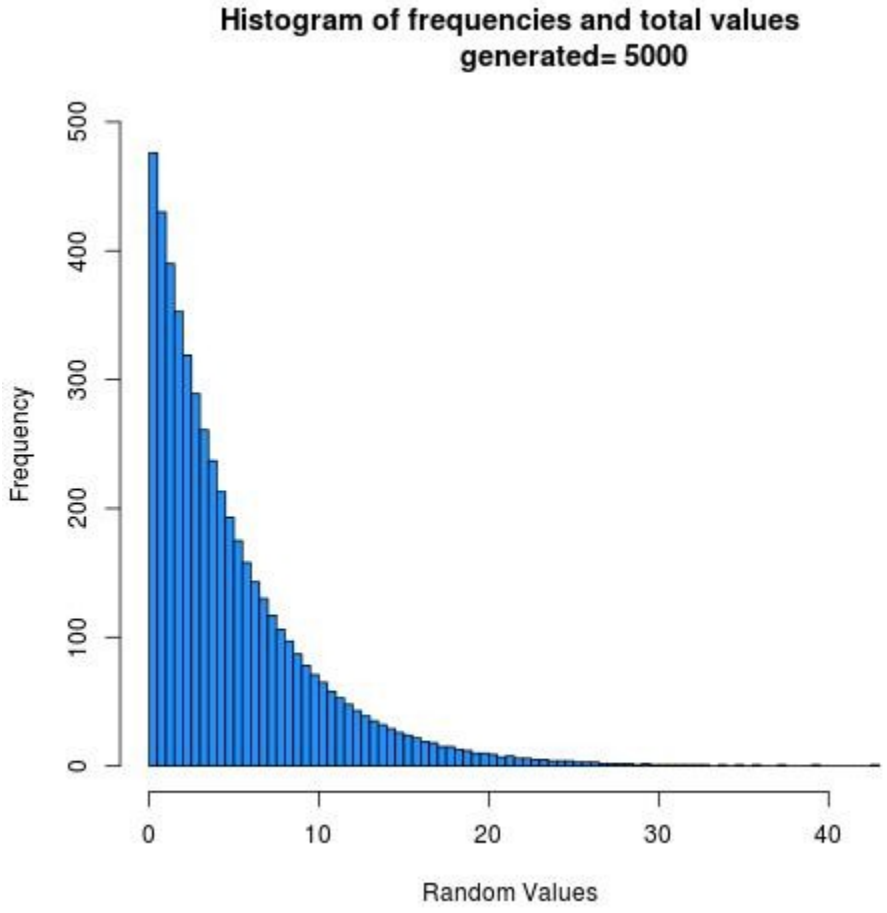
“

```
func_rand <- function(val, m){
	return(val/m);
}

func_val <- function(val, a, b, m){
	next_val <- ((a * val) + b) %% m;
	return(next_val);
}

func <- function(initial, a, b, m, end, lambda){
	arr <- array(end);
	cou <- array(end);

	val <- initial;
	random <- func_rand(val, m);

	for(count in 1:end){
		val <- func_val(val, a, b, m);
		random <- func_rand(val, m);
		arr[count] <- random;
	}

	sum <- 0.0;
	min <- arr[1];
	max <- arr[1];

	for(index in 1:end){
		arr[index] <- -log(1 - arr[index])/lambda;
		cat(sprintf("%g\n", arr[index]));

		sum <- sum + arr[index];

		if(arr[index] < min){
			min <- arr[index];
		}

		if(arr[index] > max){
			max <- arr[index];
		}
```

```
        }

        mean <- sum/end;

        cat(sprintf("Min=%g\nMax=%g\nMean=%g\n", min, max, mean));

        hist(arr, 100,freq = TRUE, include.lowest = TRUE, right = TRUE, density =
NULL,
                angle = 45, col = '#1E90FF', border = NULL,
                main = paste("Histogram of frequencies and total values
                generated=",end), xlim=range(0,max), ylim=range(0,(500)), xlab =
'Random Values' ,
                axes = TRUE, plot = TRUE, labels = FALSE);

        dev.copy(jpeg,"hist.jpg");
        dev.off();
}

main<-function(){

        init_value <- 3452;
        bound <- 5000;
        lambda <- 0.2;

        func(init_value, 4696, 0, 5003, bound, lambda);
}
”
```

***Histogram*:**

histogram for R code (using R-graphics)



Observation:

1) From the above histogram, we can say that the distribution resembles exponential distribution.

2) The mean calculated is 4.995750, which can be approximated (upto two decimal places) to 5.00, which is required.

So, this distribution ~ exp(0.2)

3) The closeness of the observed mean to the expected mean proves the efficacy of the Linear Congruential Generator used to generate the uniform random numbers

Result:

1) The mean is 4.995750

2) The minimum is 0.001000.

3) The maximum is 42.588965.

2.Solution:-

C++ code:

“

```
#include <iostream>
#include <cstdio>
#include <cmath>

#define LL long long

using namespace std;

double func_rand(LL int val, LL int m){
	return (double)val/m;
}

LL int func_val(LL int val, LL int a, LL int b, LL int m){
	LL next_val;
	next_val = ((a * val) + b) % m;

	return next_val;
}

void function(LL int initial, LL int a, LL int b, LL int m, int end, double array[]){
	LL int val, count = 0;
	LL int index, quot;
	double random, prev;

	val = initial;
	random = func_rand(val, m);

	for(count=0;count<end;++count){
		val = func_val(val, a, b, m);
		random = func_rand(val, m);

		array[count] = random;
	}
}

void exponential(double lambda, double array[], LL int bound){
	int index;
```

```
	for(index=0; index<bound; ++index){
		array[index] = -(double)log(1.0 - array[index])/lambda;
	}
}

int main(){
	LL int index;
	double array1[5005];
	double array2[5005];
	double array3[5005];
	double array4[5005];
	double array5[5005];
	double array[5005];
	LL int init_value[5] = {3452, 997, 34, 269, 5};
	LL int bound = 5000;
	double lambda = 5;
	double min, max, mean, sum = 0.0;

	function(init_value[0], 4696, 0, 5003, bound, array1);
	exponential(lambda, array1, bound);
	function(init_value[1], 4696, 0, 5003, bound, array2);
	exponential(lambda, array2, bound);
	function(init_value[2], 4696, 0, 5003, bound, array3);
	exponential(lambda, array3, bound);
	function(init_value[3], 4696, 0, 5003, bound, array4);
	exponential(lambda, array4, bound);
	function(init_value[4], 4696, 0, 5003, bound, array5);
	exponential(lambda, array5, bound);

	for(index=0; index<bound; ++index){
		array[index] = array1[index] + array2[index] + array3[index] +
array4[index] + array5[index];
		printf("%lf\n", array[index]);
	}

	min = max = array[0];
	for(index=0; index<bound; ++index){
		sum+=array[index];

		if(array[index] < min)
			min = array[index];
```

```
            if(array[index] > max)
                max = array[index];
        }
        mean = (double)sum/bound;

        printf("Min=%lf\nMax=%lf\nMean=%lf\n", min, max, mean);

        return 0;
}
```

”

R Code:

“

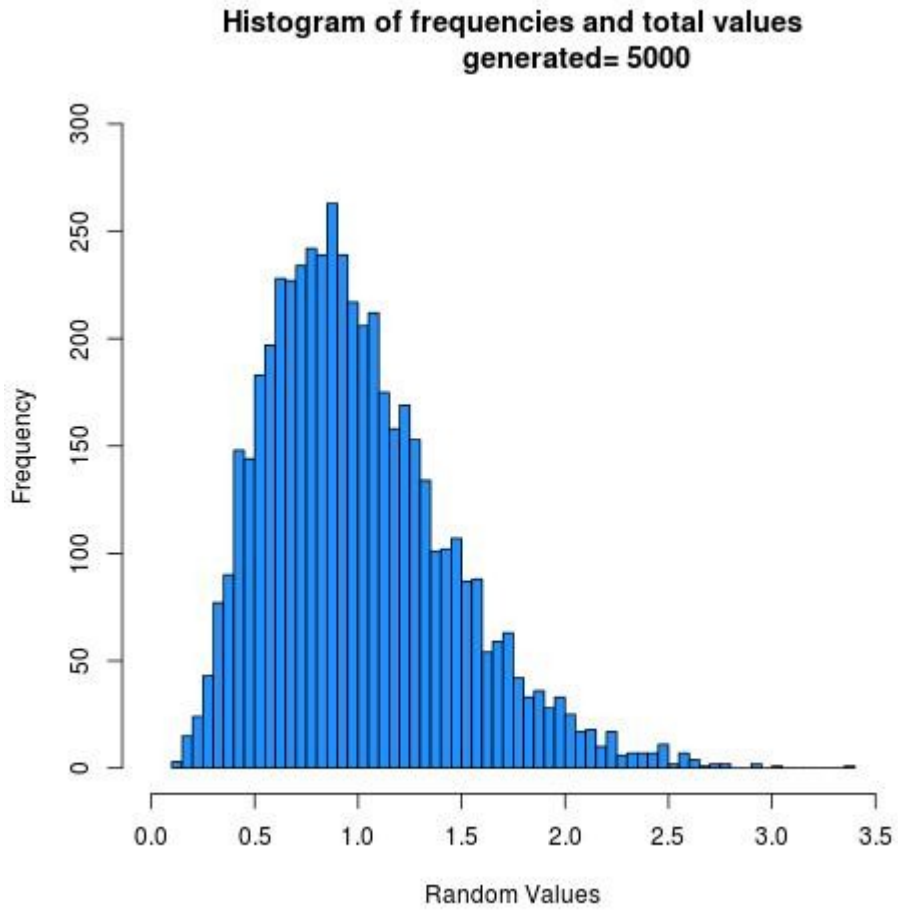
```
func_rand <- function(val, m){
	return(val/m);
}

func_val <- function(val, a, b, m){
	next_val <- ((a * val) + b) %% m;
	return(next_val);
}

func <- function(a, b, m, end, lambda){
	arr1 <- array(end);
	arr2 <- array(end);
	arr3 <- array(end);
	arr4 <- array(end);
	arr5 <- array(end);
	arr <- array(end);

	val1 <- 3452;
	val2 <- 997;
	val3 <- 34;
	val4 <- 269;
	val5 <- 5;

	for(count in 1:end){
		val1 <- func_val(val1, a, b, m);
		arr1[count] <- func_rand(val1, m);
		arr1[count] <- -log(1 - arr1[count])/lambda;

		val2 <- func_val(val2, a, b, m);
		arr2[count] <- func_rand(val2, m);
		arr2[count] <- -log(1 - arr2[count])/lambda;

		val3 <- func_val(val3, a, b, m);
		arr3[count] <- func_rand(val3, m);
		arr3[count] <- -log(1 - arr3[count])/lambda;
```

```
		val4 <- func_val(val4, a, b, m);
		arr4[count] <- func_rand(val4, m);
		arr4[count] <- -log(1 - arr4[count])/lambda;

		val5 <- func_val(val5, a, b, m);
		arr5[count] <- func_rand(val5, m);
		arr5[count] <- -log(1 - arr5[count])/lambda;

		val <- arr1[count] + arr2[count] + arr3[count] + arr4[count] +
arr5[count];
		arr[count] <- val;
		print(arr[count])
	}

	sum <- 0.0;
	min <- arr[1];
	max <- arr[1];

	for(index in 1:end){
		sum <- sum + arr[index];

		if(arr[index] < min){
			min <- arr1[index];
		}

		if(arr[index] > max){
			max <- arr1[index];
		}
	}

	mean <- sum/end;

	cat(sprintf("Min=%g\nMax=%g\nMean=%g\n", min, max, mean));

	hist(arr, 100,freq = TRUE, include.lowest = TRUE, right = TRUE, density =
NULL,
		angle = 45, col = '#1E90FF', border = NULL,
		main = paste("Histogram of frequencies and total values
		generated=",end), xlim=range(0,3.5), ylim=range(0,(300)), xlab =
```

```
'Random Values' ,
            axes = TRUE, plot = TRUE, labels = FALSE);

    dev.copy(jpeg,"hist1.jpg");
    dev.off();
}

main<-function(){
    bound <- 5000;
    lambda <- 5;

    func(4696, 0, 5003, bound, lambda);
}

”
```

***Histogram*:**

histogram for R code (using R-graphics)



Observation:

1) From the above histogram, we can say that the distribution resembles gamma distribution.

2) The mean calculated is 0.999173, which can be approximated (upto two decimal places) to 1.00, which is required.

So, this distribution ~ gamma(5,5)

3) The gamma distribution is obtained by using inverse transformation on 5 different exponential distributions.

Result:

1) The mean is 0.999173.

2) The minimum is 0.124129.

3) The maximum is 3.361399.

3.Solution:-

R code:

“

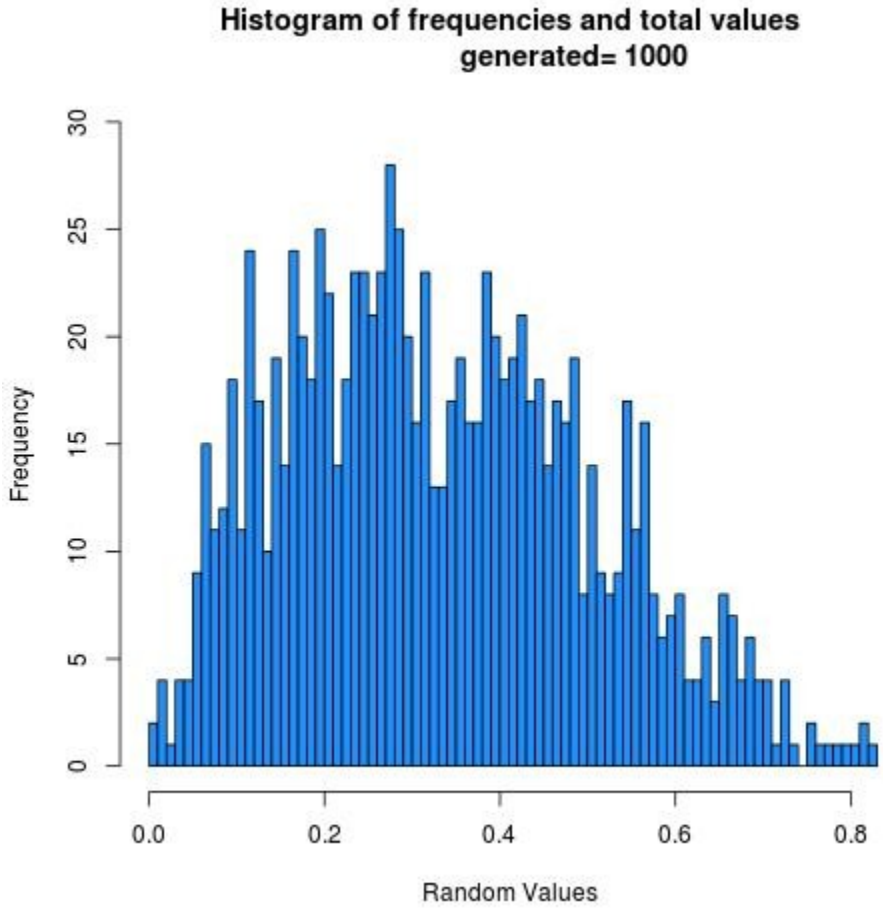
```
func_rand <- function(val, m){
	return(val/m);
}

func_val <- function(val, a, b, m){
	return(((a*val)+b)%%m);
}

func_f <- function(val){
	c=2.109375;
	ans = (20*val*((1-val)**3))/c;
	return(ans);
}

func <- function(initial, a, b, m, end, initial_2){
	count<-0;
	arr<-array(1002);

	val <- func_val(initial, a, b, m);
	random <- func_rand(val, m);
	val_2 <- func_val(initial_2, a, b, m);
	random_2 <- func_rand(val_2, m);

	while(count<=end){
		if(random_2<=func_f(random)){
			count = count+1;
			arr[count] = random;
		}
		val <- func_val(val, a, b, m);
		random <- func_rand(val, m);

		val_2 <- func_val(val_2, a, b, m);
		random_2 <- func_rand(val_2, m);
```

```
	}

	max<-0;
	min<-1;
	sum<-0;
	for(i in 1:end){
		cat(sprintf("arr[%g]\t%g\n", i, arr[i]));

		if(arr[i]>max){
			max = arr[i];
		}

		if(arr[i]<min){
			min = arr[i];
		}
		sum = sum + arr[i];
	}
	mean = sum/end;
	cat(sprintf("min = %g\nmax = %g\nmean = %g\n", min, max, mean));

	hist(arr, 100,freq = TRUE, include.lowest = TRUE, right = TRUE, density =
NULL,
		angle = 45, col = '#1E90FF', border = NULL,
		main = paste("Histogram of frequencies and total values
		generated=",end), xlim=range(0,max), ylim=range(0,(30)), xlab =
"Random Values" ,
		axes = TRUE, plot = TRUE, labels = FALSE);

	dev.copy(jpeg,'img.jpg');
	dev.off();
}

initial = 1853;
a = 5301;
b = 0;
m = (2**31)-1;
bound = 1000;
initial_2 = 12451;

func(initial, a, b, m, bound, initial_2);
```

”

Histogram:

histogram for R code (using R-graphics)



Observation:

1) *From the above histogram, we can say that the distribution resembles gamma distribution.*
2) *The mean calculated is 0.332093, which can be approximated (upto two decimal places) to 0.33, which is required.*

So, this distribution ~ distribution with $f(x)=20x(1-x)^3$

Result:

1) *The mean is 0.332093.*
2) *The minimum is 0.00457408 .*
3) *The maximum is 0.824751.*