APPENDIX 3

A-3.1. Program for generation of Markov Chain –

```
/*program to generate Markov Chain*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
{
int x,i,n;
float u,p;
p=0.5;
clrscr();
randomize();
printf("\n Enter value of n");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
  u=(float)random(RAND_MAX)/RAND_MAX;
  if (u<p)
  {
   x = 0;
   p = 0.5;
  }
  else
  {
   x = 1;
   p = 0.3333;
  }
  printf("%d",x);
}
getch();
}
```

A-3.2. Program for generation from G(2.43, 1) by Accept – Reject method –

```
//Program for Gamma distribution by Accept-Reject method
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
void main()
  {
  int i,j,n,n1;
  float u,u1,x,y,v,m,p,s,sq;
  FILE *fp;
  fp=fopen("acceptg.xls","w");
  clrscr();
  printf ("\n enter sample size n= ");
  scanf ("%d",&n);
  randomize();
  s=0; sq=0;n1=0;
  for(j=1;j<=n;j++)
  {
      x=0;
      for(i=1;i<=2;i++)
      {
            u=(float)random(RAND_MAX)/RAND_MAX;
            v=-(1/0.823045)*log(1-u);
            x=x+v;
      }
      u1=(float)random(RAND_MAX)/RAND_MAX;
      p=1.049388*(pow(x,0.43))*exp(-0.176955*x);
      if (u1<=p)
            {y=x;
            s=s+y;sq=sq+(y*y);
            n1=n1+1;}
      else
            {n1=n1;}
      fprintf (fp,"\n %f",y);
  }
m=s/n1; v=(sq/n1)-(m*m);
printf("\n mean = %f  Variance= %f",m, v);
fclose (fp);
getch();
}
```

A-3.3. Program for generation from G(2.43, 1) by M – H algorithm –

```
\\ Program for Gamma distribution by M-H algorithm
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
void main()
  {
  int i,j,n;
  float u,u1,x,y,v,m,p,p1,p2,s,sq;
  FILE *fp;
  fp=fopen("gamma.xls","w");
  clrscr();
  x=5;
  printf ("\n enter sample size n= ");
  scanf ("%d",&n);
  randomize();
  s=0; sq=0;
  for(j=1;j<=n;j++)
  {
      y=0;
      for(i=1;i<=2;i++)
      {
            u=(float)random(RAND_MAX)/RAND_MAX;
            v=-(2.43/2)*log(1-u);
            y=y+v;
      }
      u1=(float)random(RAND_MAX)/RAND_MAX;
      p2=(y/x)*exp((x-y)/2.43);
      //p2=pow(p , 0.43);
      p1=pow(p2,0.43);
      if (p1<=1)
            {p1=p1;}
      else
            {p1=1;}
```

```
        if (u1<=p1)
                {x=y;}
        else
                {x=x;}
        s=s+x; sq=sq+(x*x);
        fprintf (fp,"\n %f",x);
   }
m=s/n; v=(sq/n)-(m*m);
printf("\n mean = %f  Variance= %f",m, v);
fclose (fp);
getch();
}
```

A-3.4. Program for gen. from bi. normal by conditional method

```
\\Program for Bivariate Normal by conditional method
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
  {
  int j,n;
  float u1,u2,z1,z2,m1,m2,v1,v2,cov,r,x1,x2,s1,s2,s12,sq1,sq2;
  FILE *fp;
  fp=fopen("binorm2.xls","w");
  clrscr();
  x1=1.2;x2=1.8;
  printf ("\n enter sample size n= ");
  scanf ("%d",&n);
  randomize();
  s1=0; sq1=0;s2=0; sq2=0;s12=0;
  for(j=1;j<=n;j++)
  {
      u1=(float)random(RAND_MAX)/RAND_MAX;
      u2=(float)random(RAND_MAX)/RAND_MAX;
      z1=sqrt(-2*log(u1))*sin(360*u2);
      z2=sqrt(-2*log(u1))*cos(360*u2);
      x1=z1+1; x2=(0.43589*z2)+(0.9*x1)+1.1;
      s1=s1+x1; sq1=sq1+(x1*x1);
      s2=s2+x2; sq2=sq2+(x2*x2);
      s12=s12+(x1*x2);
      fprintf (fp,"\n\s %f  %f",x1,x2);
  }
m1=s1/n; v1=(sq1/n)-(m1*m1);m2=s2/n; v2=(sq2/n)-(m2*m2);
r=((s12/n)-(m1*m2))/sqrt(v1*v2);
printf("\n mean = %0.4f %0.4f  Variance= %0.4f %0.4f corr=
        %0.4f",m1,m2,v1,v2,r);
fclose (fp);
getch();
}
```

A-3.5. Program for generation from bivariate normal by Cholesky decomposition method –

```
\\Program for Bivariate Normal by Cholesky method
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
  {
  int j,n;
  float u1,u2,z1,z2,m1,m2,v1,v2,cov,r,x1,x2,s1,s2,s12,sq1,sq2;
  FILE *fp;
  fp=fopen("binorm3.xls","w");
  clrscr();
  x1=1.2; x2=1.8;
  printf ("\n enter sample size n= ");
  scanf ("%d",&n);
  randomize();
  s1=0; sq1=0;s2=0; sq2=0;s12=0;
  for(j=1;j<=n;j++)
  {
      u1=(float)random(RAND_MAX)/RAND_MAX;
      u2=(float)random(RAND_MAX)/RAND_MAX;
      z1=sqrt(-2*log(u1))*sin(360*u2);
      z2=sqrt(-2*log(u1))*cos(360*u2);
      x1=z1+1; x2=(0.43589*z2)+(0.9*z1)+2;
      s1=s1+x1; sq1=sq1+(x1*x1);
      s2=s2+x2; sq2=sq2+(x2*x2); s12=s12+(x1*x2);
      fprintf (fp,"\n\s %f  %f",x1,x2);
  }
m1=s1/n; v1=(sq1/n)-(m1*m1);m2=s2/n; v2=(sq2/n)-(m2*m2);
r=((s12/n)-(m1*m2))/sqrt(v1*v2);
printf("\n mean = %0.4f %0.4f  Variance= %0.4f %0.4f corr=
          %0.4f",m1,m2,v1,v2,r);
fclose (fp);
getch();
}
```

A-3.6. Program for generation from bivariate normal by M – H algorithm using indep. Uniform variates –

```
\\Program for bivariate Normal by M-H algorithm (indep.uniform)
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
  {
  int j,n;
  float u1,u2,u,y1,y2,p1,m1,m2,v1,v2,cov,r,x1,x2,s1,s2,s12, sq1, sq2;
  FILE *fp;
  fp=fopen("binormU.xls","w");
  clrscr();
  x1=0.2;x2=0.1;
  printf ("\n enter sample size n= ");
  scanf ("%d",&n);
  randomize();
  s1=0; sq1=0;s2=0; sq2=0;s12=0;
  for(j=1;j<=n;j++)
  {
      u1=(float)random(RAND_MAX)/RAND_MAX;
      u2=(float)random(RAND_MAX)/RAND_MAX;
      y1=1.5*u1-0.75;
      y2=2*u2-1;
      y1=y1+x1;
      y2=y2+x2;
      u=(float)random(RAND_MAX)/RAND_MAX;
      p1=exp(-2.63157894*((y1-1)*(y1-1)-1.8*(y1-1)*(y2-2)+(y2-2)
          *(y2-2)-(x1-1)*(x1-1)+1.8*(x1-1)*(x2-2)-(x2-2)*(x2-2)));
      if (p1<1)
            {p1=p1;}
      else
            {p1=1;}
      if (u<p1)
            {x1=y1;
             x2=y2;}
```

```
        else
                {x1=x1;
                 x2=x2;}
        s1=s1+x1; sq1=sq1+(x1*x1);
        s2=s2+x2; sq2=sq2+(x2*x2);
        s12=s12+(x1*x2);
        fprintf (fp,"\n\s  %f  %f",x1,x2);
  }
m1=s1/n; v1=(sq1/n)-(m1*m1);
m2=s2/n; v2=(sq2/n)-(m2*m2);
r=((s12/n)-(m1*m2))/sqrt(v1*v2);
printf("\n mean = %f %f  Variance= %f %f corr= %f", m1,m2,
        v1, v2,r);
fclose (fp);
getch();
}
```

A-3.7. Program for generation from bivariate normal by M – H algorithm using indep. Normal variates –

```
\\Program for bivariate Normal by M-H algorithm (indep. normal)
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
 {
 int j,n;
 float u1,u2,z1,z2,u,y1,y2,p1,m1,m2,v1,v2,cov,r,x1,x2,s1,s2,s12,sq1,
     sq2;
 FILE *fp;
 fp=fopen("binormal.xls","w");
 clrscr();
 x1=1.2;x2=1.8;
 printf ("\n enter sample size n= ");
 scanf ("%d",&n);
 randomize();
 s1=0; sq1=0;s2=0; sq2=0;s12=0;
 for(j=1;j<=n;j++)
 {
     u1=(float)random(RAND_MAX)/RAND_MAX;
     u2=(float)random(RAND_MAX)/RAND_MAX;
     z1=sqrt(-2*log(u1))*sin(360*u2);
     z2=sqrt(-2*log(u1))*cos(360*u2);
     y1=sqrt(0.6)*z1;
     y2=sqrt(0.4)*z2;
     y1=y1+x1;
     y2=y2+x2;
     u=(float)random(RAND_MAX)/RAND_MAX;
     p1=exp(-2.63157894*((y1-1)*(y1-1)-1.8*(y1-1)*(y2-2)+ (y2-2)
         *(y2-2)-(x1-1)*(x1-1)+1.8*(x1-1)*(x2-2)-(x2-2)*(x2-2)));
     if (p1<=1)
           {p1=p1;}
     else
           {p1=1;}
```

```
      if (u<=p1)
            {x1=y1;
             x2=y2;}
      else
            {x1=x1;
             x2=x2;}
      s1=s1+x1; sq1=sq1+(x1*x1);
      s2=s2+x2; sq2=sq2+(x2*x2);
      s12=s12+(x1*x2);
      fprintf (fp,"\n\s %f    %f",x1,x2);
  }
m1=s1/n; v1=(sq1/n)-(m1*m1);
m2=s2/n; v2=(sq2/n)-(m2*m2);
r=((s12/n)-(m1*m2))/sqrt(v1*v2);
printf("\n mean = %f %f  Variance= %f %f corr= %f",m1,m2,
        v1,v2,r);
fclose (fp);
getch();
}
```