APPENDIX 4

A-4.1. Program for generation from beta- binomial by Gibbs Sampler –

```
\\ Program for beta- binomial by Gibbs Sampler
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
  {
  int i,j,j1,j2,k;
  float u,u1,u2,x,x1,x2,y,s,s1,s2,sq,m,v;
  FILE *fp;
  fp=fopen("gibbs1.xls","w");
  clrscr();
  y=0.5;
  randomize();
  s=0; sq=0;
  for(i=1;i<=500;i++)
  {
    for(k=1;k<=100;k++)
    {
      x=0;s1=0;s2=0;
      for (j=1;j<=16;j++)
      {
        u=(float)random(RAND_MAX)/RAND_MAX;
        if (u<=y)
        {x=x+1;}
      }
      for (j1=1;j1<=x+2;j1++)
      {
        u1=(float)random(RAND_MAX)/RAND_MAX;
        x1=-log(1-u1);
        s1=s1+x1;
      }
```

```
        for (j2=1;j2<=20-x;j2++)
        {
          u2=(float)random(RAND_MAX)/RAND_MAX;
          x2=-log(1-u2);
          s2=s2+x2;
        }
    y=s1/(s1+s2);
    }
    s=s+x; sq=sq+(x*x);
    fprintf (fp,"\n %f ",x);
  }
m=s/500; v=(sq/500)-(m*m);
printf("\n mean = %f Variance= %f ",m,v);
fclose (fp);
getch();
}
```

A-4.2. Program for generation from bivariate normal by Gibbs Sampler–

```
//program for bivariate normal by Gibbs sampler
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
 {
  int i,j,k,m;
  float u1,u2,z1,z2,m1,m2,v1,v2,x,y,s1,s2,sq1,sq2,s12,r;
  FILE *fp;
  fp=fopen("gibbs4.xls","w");
  clrscr();
  y=2;
  printf("Enter k and sample size =");
  scanf("%d %d",&k,&m);
  randomize();
  s1=0; s2=0;sq1=0;sq2=0;s12=0;
  for(i=1;i<=m;i++)
  {
    for(j=1;j<=k;j++)
    {
       u1=(float)random(RAND_MAX)/RAND_MAX;
       u2=(float)random(RAND_MAX)/RAND_MAX;
       z1=sqrt(-2*log(1-u1))*sin(360*u2);
       z2=sqrt(-2*log(1-u1))*cos(360*u2);
       x=(0.9*y)+(0.43589*z1)-0.8;
       y=(0.9*x)+(0.43589*z2)+1.1;
    }
    s1=s1+x; sq1=sq1+(x*x);
    s2=s2+y; sq2=sq2+(y*y);
    s12=s12+(x*y);
    fprintf (fp,"\n  %1.4f \t %1.4f",x,y);
  }
```

```
m1=s1/m; v1=(sq1/m)-(m1*m1);
m2=s2/m; v2=(sq2/m)-(m2*m2);
r=((s12/m)-(m1*m2))/(sqrt(v1*v2));
printf("\n mean = %f %f Variance= %f %f",m1,m2,v1,v2);
printf("\n correlation Coefficient= %f",r);
fclose (fp);
getch();
}
```

A-4.3. Program for generation from trivariate distribution by Gibbs Sampler–

```
//program for trivarate  distribution by Gibbs sampler
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void main()
  {
  int i,j,j1,j2,k,x,z,n,n1;
  float u,u1,u2,u3,x1,x2,y,p,s,s1,s2,sq,m,v;
  FILE *fp;
  fp=fopen("gibbs3.xls","w");
  clrscr();
  y=0.5;z=5;
  randomize();
  s=0; sq=0;
  for(i=1;i<=10000;i++)
  {
    for(k=1;k<=100;k++)
    {
       x=0;s1=0;s2=0;n=0;p=1;
       for (j=1;j<=z;j++)
       {
         u=(float)random(RAND_MAX)/RAND_MAX;
         if (u<=y)
         {x=x+1;}
       }
       for (j1=1;j1<=x+2;j1++)
       {
         u1=(float)random(RAND_MAX)/RAND_MAX;
         x1=-log(1-u1);
         s1=s1+x1;
       }
       for (j2=1;j2<=z-x+4;j2++)
```

```
        {
          u2=(float)random(RAND_MAX)/RAND_MAX;
          x2=-log(1-u2);
          s2=s2+x2;
        }
        y=s1/(s1+s2);
        do
        {
          u3=(float)random(RAND_MAX)/RAND_MAX;
          p=p*u3;
          n=n+1;
        }
     while (p>exp(-16*(1-y)));
     n1=n-1;
     z=n1+x;
     }
   s=s+x; sq=sq+(x*x);
   fprintf (fp,"\n %d ",x);
   }
m=s/10000; v=(sq/10000)-(m*m);
printf("\n mean = %f Variance= %f ",m,v);
fclose (fp);
getch();
}
```