Name: Jian Wang Course: MonteCarlo FSUID: Jw09r

1. Write a computer program that implements the Inverse Transformation Method for a discrete density. The program should take the density function $\{p_1, p_2, \dots, p_n\}$ as input, and output a random integer from {1, 2,…, n} with this density. To test your, program apply it to the density {0.2, 0.2, 0.4, 0.1, 0.1}. Generate 1000 random integers in {1, 2, 3, 4, 5} from this density, and plot a histogram for the data you obtain. Visually inspect whether the histogram is a good approximation of the theoretical density.

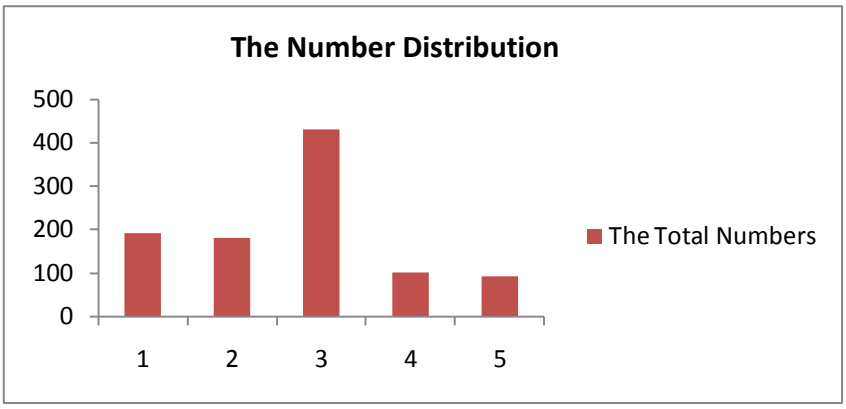
Answer:
The theorem of the inverse transformation is as follows:
Let U1, U2,…,Un be a random sample of size n, i.e., i.i.d. random variables, from the uniform distribution U(0, 1). Let F be a distribution function and Xi = F^{-1}(Ui): Then X1, X2,…,Xn is a random sample of size n from the distribution F.

In this problem, the density is {0.2, 0.2, 0.4, 0.1, 0.1} and the random integers in {1, 2, 3, 4, 5} is from this density.

I used the Randu Method to generate the random numbers and plot the 1000 data in a histogram to see the result. As we already know that the Randu is not a very good method to generate the random numbers, so the result of the histogram may not be follow the density function {0.2, 0.2, 0.4, 0.1, 0.1}. I start from the seed which was equal to 1

So of the $x = \begin{cases} 1, & if\ u < 0.2 \\ 2, & if\ 0.2 \le u < 0.4 \\ 3, & if\ 0.4 \le u < 0.8 \\ 4, & if\ 0.8 \le u < 0.9 \\ 5, & if\ 0.9 \le u < 1 \end{cases}$

The formula of the Randu is: $X_n = 65539X_{n-1}\ Mod\ 2^{31}$,



And the data of the results are as follows:

Integer	Numbers
1	193
2	182
3	430
4	102
5	93

From the above results, we can see that the Randu results are relatively good under the inverse transformation method test.
The Program is as the attachment 1

2. Let T be the number of arithmetical operations $(+,-,\times,\div)$ and comparisons (checking if a number is less than another) that need to be computed to generate a value from X _ Poisson(λ) using Algorithm (Poisson). Find E[T] and Var(T): (Ignore any operations in initialization and assignments a := b:)

Answer: The algorithm of the Poisson distribution is as follows:

1) Set i=0, p=$e^{-\lambda}$, $F = P$
2) If u<F, then set X = i and stop
3) Update: P =$\frac{\lambda}{i+1}$P, F = F+P, i = i+1
4) Go to step 2

Input is a uniform random number u. The output is a random value from the X∼Poisson(λ)

From the above algorithm, we can see that in each process, we have one comparison and totally five numbers of the operations which are $\frac{\lambda}{i+1}, i+1, \frac{\lambda}{i+1} * P, F+P \; and \; i+1.$

We can see that if the X = i then the algorithm will have exactly gone through the i+1 steps.

However the situation of the operations and the comparison is different, for example, if i=0 then comparison be 1 and the operations will be 0, since it stops before the operations conduct.

Similarly, if the i=1 then the number of comparison is 2 and the numbers of the operations are 5*1 =5. In the following, I can get the common formula of the total numbers:

The total numbers of the operations and the comparisons are :

T=X+1+5*X=6X+1

So I can get the expectation and the variance of T as follows:

E(T)=E(x+1+5*X) = 6E(x) +1=6λ + 1

Var(T) = Var(6X+1)= 36 λ

3. Write a computer program that implements the Box-Muller method. Your program will output independent standard Normals X1,X2 when you input independent uniform variables U1,U2: Then generate 1000 pairs (X2i-1,X2i) i = 1,…,1000; and plot these pairs in R^2; using the following two generators to compute the corresponding pairs (U2i-1; U2i):

(a) Mersenne twister,

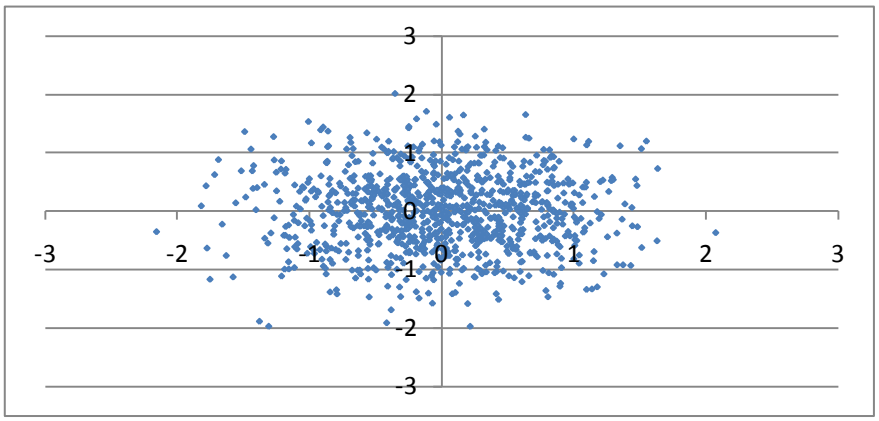
(b) A clearly poor LCG: xn+1 =1229xn + 1(mod 2048)

What conclusions do you make based on these graphs?

Answer: The algorithm of the Box-Muller method is as follows:

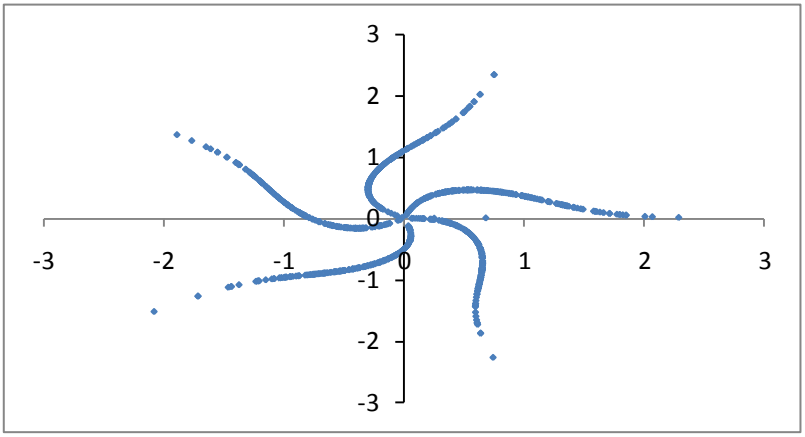
1) Generate $u_1, u_2\ form\ U(0,1)$;
2) Compute: $R^2 = -2logu_1\ and\ \theta = 2\pi u_2$
3) $X = Rcos\theta = \sqrt{-2logu_1}\cos(2\pi u_2)\ and\ Y = Rsin\theta = \sqrt{-2logu_1}\sin(2\pi u_2)$

In the first method, I used the Twister to generate the 1000 random numbers. Here I let the $\pi\ equals\ to\ 3.14$, the results are as follows:



In the second method, generate the random value via the LCG: xn+1=1229xn + 1(mod 2048) with the seed equals to 1

The result is as follows:



Conclusion: From the figure, we can see that the random value that generated by the Twister are fairly good compared with the numbers generated by the LCG. The points of the LCG mainly lied on the five curves, which showed that the numbers are not random.

The program is as attachment 2

4. Write a computer program that implements the Beasley-Springer-Moro algorithm (see page 67-68 of Glasserman's book or Blackboard). This algorithm gives an approximation to the inverse normal cdf. Then generate 2000 standard Normals, N(0; 1); using this method. Test the normality of this data, and the Box-Muller data you generated in Problem 3 (a), using the Anderson-Darling test. (You will need a table for the percentiles for the Anderson-Darling statistic. See the paper "EDF Statistics for goodness of Fit and Some Comparisons" on Blackboard) Which data set is closer to the normal distribution?

Answer: The algorithm of the Beasley- Springer_Moro algorithm for approximating the inverse normal is as follows:

Input: u between 0 and 1

Output: x. approximation to $\Phi^{-1}(u)$

Y= u-0.5

If |Y| <0.42

r = y*y

X= y*(((($a_3 * r + a_2$) $* r + a_1$) $* r + a_0$/((((($b_3 * r +$ b_2) $* r + b_1$) $* r + b_0$))*r+1

Else r=u

If(y>0) r= 1-u

R=Log(-log(r))

X=$c_0 + r * (c_1 + r * \left(c_2 + r * \left(c_3 + r * \left(c_4 + r * \right. \right. \right. \right.$

$\left. \left. \left. \left. \left(c_5 + r * \left(c_6 + r * (c_7 + r * c_8) \right) \right) \right) \right) \right)$

If (y<0) x=-x

Return x

With the help of the software VBA, I generate the 2000 random numbers under the Beasley- Springer_Moro algorithm. The program is as attachment.

I generate the u with the Twister method, and use the Anderson-Darling test to test if the Beaslyey-Springer_Moro is a good method.

The formula of $A^2 is\ A^2 = \frac{-\{\sum_{i=1}^{n}(2i-1)[lnz_i + \ln(1-Z_{n+1-i})]\}}{n} - n$

The result of the $A^2\ is\ 1.01107861620699$, and using the table of the A-D statistics, the 15% percentile is the 1.610. **So my conclusion is that we can not reject the Beasley-Springer-Moro algorithm generating the normal distribution number under the 15% level. The 5% and the 1% percentile of**

A^2 is 2.492 and 3.857 $respectively$, Hence we also can not reject at 1% and the 5% level.
Therefore the Beasley-Springer-Moro algorithm is a good method.

Similarly the A^2 is 1.46500131162088, which is also less than 1.610. So we also can not **reject the** Box-Muller data **generating the normal distribution numbers under the 15% level and also we can not reject at the 1% and the 5% level, which the percentile is 2.492 and 3.857.**

In conclusion, since 1.01107861620699 is less than the 1.46500131162088, we may conclude that the data generate from the Beasley-Springer-Moro algorithm is closer to the normal distribution.

The program is as attachment 3

5. Give a method for generating the Weibull distribution function

$$F(x) = 1 - e^{-\alpha x^{\beta}} \quad 0 < x < \infty$$

Answer: The method of the inverse transformation is: To generate a random variable X with distribution function (c.d.f) F, we generate a uniform random number u from U(0; 1) and set X = F^{-1}(u)

In this case, the $F(x) = 1 - e^{-\alpha x^{\beta}}$, so the inverse function of F(x) is $F^{-1}(x) = (\frac{\log(1-x)}{-\alpha})^{\frac{1}{\beta}}$,

Also note that the distribution of the 1-U and U are both U(0,1), if U is U(0,1). Combining these two observations, we conclude that

Algorithm to Generating Weibull distribution

1. Generate u from $U(0,1)$
2. Set x=$(\frac{\log(u)}{-\alpha})^{\frac{1}{\beta}}$

6. Give a method for generating a random variable with density function

$$f(x) = \begin{cases} e^{2x}, & -\infty < x < 0 \\ e^{-2x}, & 0 < x < +\infty \end{cases}$$

Answer:

I used two methods to generate the random values.

The first method is as follows:

Firstly, I derived the C.D.F based on the density function. The result is listed as follows:

$$F(x) = \begin{cases} \frac{1}{2}e^{2x}, & -\infty < x < 0 \\ 1 - \frac{1}{2}e^{-2x}, & 0 < x < +\infty \end{cases}$$

Besides, I also calculated that $P\{X < 0\} = \frac{1}{2}e^{2x}|_{-\infty}^{0} = \frac{1}{2}$, so I write the algorithm as follows:

Algorithm

1. Generate u from $U(0,1)$
2. If u belongs to the interval (0,0.5), then $X = \frac{\log(2u)}{2}$
3. If u belongs to the interval (0.5,1), then $X = \frac{2\log(1-u)}{-2}$

The second method is as follows:

Generate the |z|.

From the problem, we can know that the density function is the symmetric. So the |Z| can take z=x when the x>0 and z=-x when the x<0.

So the algorithm can be showed as follows:

1) Generate a u from U(0,1)
2) Set |Z| = $\frac{\ln(2u)}{-2}$
3) Finally, I generate the Z form the |Z| as follows:

$$z = \begin{cases} |Z| \ \ with\ probability\ \frac{1}{2} \\ -|Z| with\ probability\ \frac{1}{2} \end{cases}$$

7. Verify the equation

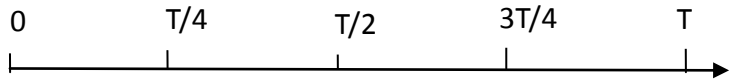
$$W\left(\frac{3}{4}T\right) = \frac{1}{2}\left(W\left(\frac{T}{2}\right) + W(T)\right) + \frac{\sqrt{2T}}{4}Z_4$$

using Lemma 5 in your lecture notes.

Answer: The Lemma 5 is as follows:
The conditional distribution of W(s) given that W(t1) = x1 and W(t2) = x2 where t1 < s < t2 is a normal distribution with

$E[W(s)|W(t_1) = x_1 \& W(t_2 = x_2)] = \frac{x_1(t_2 - s) + x_2(s - t_1)}{t_2 - t_1}$

$Var[W(s)|W(t_1) = x_1 \& W(t_2 = x_2)] = \frac{(t_2 - s)(s - t_1)}{t_2 - t_1}$



We also know that $\frac{W(3T/4) - u}{\sigma} \sim N(0,1)$,

The

$$E\left(w(\frac{3T}{4})\right) = \frac{W(T) * \left(\frac{T}{4}\right) + W\left(\frac{T}{2}\right) * \left(\frac{T}{4}\right)}{T - \frac{T}{2}} = \frac{1}{2}\left(W(T) + W\left(\frac{T}{2}\right)\right)$$

$$Var\left(W(\frac{3T}{4})\right) = \frac{\left(\frac{T}{4}\right) * (\frac{T}{4})}{T - \frac{T}{2}} = T/8$$

The order of the sequence is that W(T), W(T/2), W(T/4), W(3T/4), so we generate a Z_4 follow the standard normal distribution. Then we can get the following formula:

$$\frac{W(3T/4) - u}{\sigma} = Z_4 \Longrightarrow W\left(\frac{3}{4}T\right) = \frac{1}{2}\left(W\left(\frac{T}{2}\right) + W(T)\right) + \frac{\sqrt{2T}}{4}Z_4$$

Attachment 1

```
Sub Inverse_transformation_method()

Dim I as integer

For I = 1 to 1000

If worksheet(“sheet1”).cells(I,”b”).value< 0.2 then
   Worksheet(“sheet1”).cell(I,”c”).value=1
Else if
   worksheet(“sheet1”).cells(I,”b”).value< 0.4 then
   Worksheet(“sheet1”).cell(I,”c”).value=2
Else if
   worksheet(“sheet1”).cells(I,”b”).value< 0.8 then
   Worksheet(“sheet1”).cell(I,”c”).value=3
Else if
   worksheet(“sheet1”).cells(I,”b”).value< 0.9 then
   Worksheet(“sheet1”).cell(I,”c”).value=4
Else
   Worksheet(“sheet1”).cell(I,”c”).value=5

Endif
Endif
Endif
Endif

Next i
End sub
```

Attachment 2

The algorithm of the Box-Muller method is as follows:

1) Generate $u_1, u_2\ form\ U(0,1)$;
2) Compute: $R^2 = -2logu_1\ and\ \theta = 2\pi u_2$
3) $X = Rcos\theta = \sqrt{-2logu_1}\cos(2\pi u_2)\ and\ Y = Rsin\theta = \sqrt{-2logu_1}\sin(2\pi u_2)$

Sub Box-Muller_method()

Dim I as integer
Dim x(2000),y(2000) as double
Dim r as double dim theta as double
Dim u as double

For I = 1 to 2000
U= worksheets("sheet1").cells(I,"a").value
Theta=2*3.14* worksheets("sheet1").cells(I,"b").value
r= sqrt(-2*application.worksheetfunction.ln(u)
x(i)=r*Application.worksheetfunction.cos(theta)
Y(i)= r*Application.worksheetfunction.sin(theta)

Next i

End sub

Attachment 3 The program of the BSM Method and the AD Test

```
Sub BSM()
Dim r, y, x, sum As Double
Dim c0 As Double
a0 = 2.50662823884
a1 = -18.61500062529
a2 = 41.39119773534
a3 = -25.44106049637
b0 = -8.4735109309
b1 = 23.08336743743
b2 = -21.06224101826
b3 = 3.13082909833
c0 = 0.337475482272615
c1 = 0.976169019091719
c2 = 0.160797971491821
c3 = 2.76438810333863E-02
c4 = 3.8405729373609E-03
c5 = 3.951896511919E-04
c6 = 3.21767881768E-05
c7 = 2.888167364E-07
c8 = 3.960315187E-07
For i = 1 To 2000
Worksheets("sheet1").Select
u = Cells(i, "c").Value
y = Cells(i, "c").Value - 0.5
If Abs(y) < 0.42 Then
r = y * y
x = y * (((a3 * r + a2) * r + a1) * r + a0) / ((((b3 * r + b2) * r + b1) * r + b0) * r + 1)
Else
r = u
If (y > 0) Then
r = 1 - u
End If
r = Application.WorksheetFunction.Ln(-Application.WorksheetFunction.Ln(r))
x = c0 + r * (c1 + r * (c2 + r * (c3 + r * (c4 + r * (c5 + r * (c6 + r * (c7 + r * c8)))))))
If y < 0 Then
x = -x
End If
End If
Worksheets("sheet1").Cells(i, "d").Value = x
Next i
sum = 0
```

```
End Sub

Sub Adtest()
Dim a1, a2 As Double
Worksheets("sheet1").Select
For i = 1 To 2000
a1 = Cells(i, "e").Value
a2 = 1 - Cells(2001 - i, "e").Value
a3 = Application.WorksheetFunction.Ln(a1)
a4 = Application.WorksheetFunction.Ln(a2)
sum = sum + (2 * i - 1) * (a3 + a4)
Next i
Worksheets("sheet1").Range("g3").Value = -sum / 2000 - 2000
End Sub
```