Monte Carlo MAP 5615 HW5

Sen Zhang

**2**

**(a)**

Using random shifting the halton sequence together with the Box Muller method, I gained 40 estimates for the call option price. Please see code in appendix.

|  |  |  |  |
| --- | --- | --- | --- |
| 8.3351 | 8.3765 | 8.3587 | 8.3679 |
| 8.3667 | 8.3828 | 8.3591 | 8.3324 |
| 8.4359 | 8.3263 | 8.3714 | 8.334 |
| 8.3678 | 8.3991 | 8.323 | 8.3508 |
| 8.3952 | 8.3819 | 8.3359 | 8.3863 |
| 8.3607 | 8.2773 | 8.3657 | 8.3756 |
| 8.4193 | 8.3224 | 8.354 | 8.3553 |
| 8.3467 | 8.3599 | 8.3372 | 8.3935 |
| 8.3974 | 8.3348 | 8.3332 | 8.3706 |
| 8.3846 | 8.4171 | 8.3475 | 8.3807 |

**(b)**

Using random shifting the halton sequence together with the Moro algorithm, I gained 40 estimates for the call option price.

|  |  |  |  |
| --- | --- | --- | --- |
| 8.4521 | 8.2651 | 8.3565 | 8.3265 |
| 8.2873 | 8.5268 | 8.2556 | 8.3156 |
| 8.3773 | 8.1957 | 8.5168 | 8.3326 |
| 8.3282 | 8.3015 | 8.3769 | 8.1995 |
| 8.2864 | 8.3824 | 8.3118 | 8.3898 |
| 8.4361 | 8.3902 | 8.2723 | 8.3002 |
| 8.3657 | 8.3199 | 8.3382 | 8.3966 |
| 8.4705 | 8.5041 | 8.4136 | 8.3729 |
| 8.3863 | 8.3899 | 8.2179 | 8.3865 |
| 8.2585 | 8.3765 | 8.5217 | 8.3533 |

**(c)**

Using Black-Scholes-Merton formula: 

Where ,



Considering,

**The call option price .**

**(d)**

Using Anderson darling test, the test statistics for the two methods are as following:

Box muller method: 0.2489;

Moro algorithm: 0.3838;

To determine the critical point for statistics, considering the mean is known and variance is unknown, Case 2 can be applied to this problem. The critical value at 5% significant level is 2.323, which is bigger than both statistics. So we accept that both methods generate standard normal variables.

The smaller statistics is the better. The two statistics are very close. It is hard to say which one is better. To some extent Box muller method is a little bit better than Moro algorithm method.

**(e)**

I don’t think so. As shown above, both two algorithms work well and Moro algorithm is worse than box muller method to some extent.

# Appendix

% Halton\_QMC\_EuroOption.m

path = 10000;

num\_est = 40;

dim = 10

u=zeros(path,dim);

x=zeros(dim,path);

u = HaltonGenerator(dim,path);

option = zeros(1, num\_est);

for k = 1: num\_est

%Random Shifting

u\_shift=rand(dim,1);

u = u + repmat(u\_shift,1,path);

u = mod(u,1);

%Box-Muller

for i = 1:path

for j = 1:2:dim

[x(j,i),x(j+1,i)]=BoxMuller(u(j,i),u(j+1,i));

end

end

%Moro

%{

for i = 1:path

for j = 1:dim

x(j,i)= InverseNormal\_Moro(u(j,i));

end

end

%}

mean = 0.1;

%std = 0.3/sqrt(10);

std = 0.3;

sum = 0.0;

s0 = 50;

strike = 50;

for i = 1:path

s = s0;

for j = 1:dim

s = s \* exp( ( mean - std \* std / 2 ) \* 0.1 + std \* sqrt(0.1) \* x( j , i ) );

end

sum = sum + max( s - strike ,0 );

%sum = sum + s;

end

%Average and Discount to Present Value

option(1,k) = exp(-mean)\*(sum/path);

end

%HaltonGenerator.m

function [ arr ] = HaltonGenerator( dim ,num )

if nargin < 2

dim = 10;

%num = 40\*10000;

num = 20;

end

arr = zeros(dim,num);

p = primes(29);

for i = 1 : dim

base = p(i);

for j = 1 : num

n0 = j;

h = 0;

ib = 1 / base;

while n0 > 0

n1 = floor(n0 / base);

coef = n0 - n1 \* base;

h = h + ib \* coef;

ib = ib / base;

n0 = n1;

end

arr(i,j) = h;

end

end

end

function U=BoxMuller(u1, u2)

U=[];

U(1) = sqrt(-2\*log(u1)).\*cos(2\*pi\*u2);

U(2) = sqrt(-2\*log(u1)).\*sin(2\*pi\*u2);

end