Econ 512 HW 4

Yinshi Gao
yzg115

1. Using Dart-Throwing Method with Quasi-Monto Carlo

Using Neiderrieter, draw 1000 points in the unit square, the approximation gotten is $\pi = 3.140000000000000$.

Program

```
%% Problem 1 Using Quasi-Monto Carlo
% Define indicator function
f = @(x,y) double((x.^2 + y.^2 <= 1))

% Use `N'eiderrieter, 'W'eyl and 'H'aber points do the 1000 draw
addpath ('../CETools');
[n] = qnwequi(1000, [0 0], [1, 1], 'N');

% Compute the value of f with each element of n
f_val = f(n(:,1), n(:,2));

% Compute PI
PI_1 = 4 / size(n,1) * sum(f_val)
```

2. Using Dart-Throwing Method with Newton-Cotes

Since Newton-Cotes is designed for one-dimensional integration, I first equally draw 1000 points in the y-axis. Then given each y, use Newton-Cotes method to integrate respect to x in the interval $[0, 1]$. Finally, take mean along the y-axis. The approximation gotten is $\pi = 3.140410000000001$.

Program

```
%% Problem 2 Using Newton-Cotes
f_1 = @(x) double(x.^2 + y.^2<=1)

% Cut y-axis and get 1000 points
Y = linspace(0,1,1000);
fval1 = []; % try to evaluate intergration of f_1 given each y value
for i = 1:size(Y,2)
    y = Y(i);
    f_1 = @(x) double(x.^2 + y.^2<=1);
    fval1(i) = Int_trap(f_1, 0,1,1000);
end

PI_2 = 4 / 1000 * sum(fval1)
```

3. Using 2nd Approach with Quasi-Monto Carlo

Same in problem 1, I use Neiderrieter draw 1000 points in the interval $x \in [0, 1]$. The approximation gotten is $\pi = 3.142530524029050$.

Program

```
%% Problem 3 Using Quasi-Monto Carlo
% Define function
f2 = @(x) sqrt(1-x.^2)

% Use `N'eiderrieter, 'W'eyl and 'H'aber points do the 1000 draw
[n] = qnwequi(1000, 0, 1, 'N');

% Compute PI
PI_3 = 4 / size(n,1) * sum(f2(n))
```

4. Using 2nd Approach with Newton-Cotes

Draw 1000 points along the interval $x \in [0, 1]$. The approximation gotten is $\pi = 3.141555466911027$.

Program

```
%% Problem 4 Using Newton-Cotes
%Compute PI
PI_4 = 4 * Int_trap(f2, 0,1,1000)
```

5. Table and Comparison

First, calculate mean squared error of 200 simulations for Quasi-Monto Carlo and Newton-Cotes methods, and squared error for Newton-Cotes method.

```
%% Problem 5

% Calculate Mean Squared Error for Quasi-Monto Carlo

Qmc_mse = [0,0,0];
m = [1000,10000,1000000];
for i = 1: size(m,2)
    j = 1;
    while j <=200
        [n] = qnwequi(m(i), 0, 1, 'N');
        Qmc_mse(i) = Qmc_mse(i) + (pi - 4 / size(n,1) * sum(f2(n)))^2;
        j = j + 1;
    end
    Qmc_mse(i) = Qmc_mse(i)/200;
end

% Calculate Mean Squared Error for Newton-Cotes

Nc_mse = [0,0,0];
m = [1000,10000,1000000];
for i = 1: size(m,2)
    j = 1;
    while j <=200
        Nc_mse(i) = Nc_mse(i) + (pi - 4 * Int_trap(f2, 0,1, m(i)))^2;
        j = j + 1;
    end
    Nc_mse(i) = Nc_mse(i)/200;
end

% Calculate Squared Error for Newton-Cotes

Nc_se = [0,0,0];
m = [1000,10000,1000000];
for i = 1: size(m,2)
    Nc_se(i) =(pi - 4 * Int_trap(f2, 0,1, m(i)))^2;
end
```

And then generate table for comparison,

```
% Generate Table for Comparison

Method  = {'Quasi-MC MSE'; 'Newton-Cotes MSE'; 'Newton-Cotes SE'};
Draw_1000 = [Qmc_mse(1); Nc_mse(1); Nc_se(1)];
Draw_10000 = [Qmc_mse(2); Nc_mse(2); Nc_se(2)];
Draw_1000000 = [Qmc_mse(3); Nc_mse(3); Nc_se(3)];

T = table(Method,Draw_1000,Draw_10000,Draw_1000000)
```

The table we get is,

Method	Draw_1000	Draw_10000	Draw_1000000
'Quasi-MC MSE'	8.79600960831437e-07	1.12073141383911e-09	2.9589673915603e-13
'Newton-Cotes MSE'	1.38284907761753e-09	1.38292536227552e-12	1.38293465815156e-18
'Newton-Cotes SE'	1.38284907761754e-09	1.38292536227552e-12	1.38293465815156e-18

We can see that, within each method, the more the draw, the less the (mean) squared error.

Across the method, we can see that mean squared error of Quasi-Monto Carlo method converges to 0 at the order of 2 with number of draws increases. Newton-Cotes method converges at the order of 3 with draw increases. Moreover, using same number of draws, Newton-Cotes performs better than Quasi-Monto Carlo.

The mean squared error for 200 simulations of Newton-Cotes is basically the same with squared error or Newton-Cotes method when using same number of draws.