Homework 4

Mihai Pasnicu

October 24, 2018

All the results in terms of errors and iterations are summarized in the table. Furthermore, a key factor in the code is the Miranda and Fackler's CE Tools collection.

Method	100	1000	1000
Pseudo-MC(Ind)	4.6624e-04	2.5365e-06	6.2830e-07
Quasi-MC(Ind)	1.0818e-04	8.7960e-07	1.1207e-09
Newton-Coates(Ind)	1.2828e-04	1.1066e-07	1.5048e-10
Pseudo-MC(Pyth)	3.7236e-05	3.7390e-08	3.7405e-11
Quasi-MC(Pyth)	1.3136e-02	1.7258e-03	1.7523e-04
Newton-Coates(Pyth)	6.4925e-03	8.2770e-04	7.7761e-05

1 Problem 1

The whole trick is using qnwequi function from the tool collection. The code is below.

2 Problem 2

I just defined a function that will determine the weights for the respective points. The code is below.

3 Problem 3

Same as problem 1, but adding $\sqrt{1-x^2}$.

4 Problem 4

Analogous, previous, with small changes.

5 Problem 5

In the table above, it seems that the Pseudo-MC(Pyth) is the most exact method, regardless of how many iterations are being taken.

Code:

```
% loading the prerequisites
addpath('/Users/mihai/Desktop/Second year/Empirical Methods/Lectures/CEtools'
format short e


%% Problem 1

[i, w]=qnwequi(50000, [0,0], [1,1], 'N');
test1=i(:,1).^2 + i(:,2) .^ 2 <= 1;
pi_1=4 * w' * test1;


%% Problem 2

f = @(x, y)(double(x .^ 2 + y .^ 2 <= 1));

% use function to get points
[x_2, w_2] = weights(25000, 0, 1);
f_val = zeros(25000, 25000);

for i = 1 : length(x_2)
    f_val(i, :) = f(repmat(x_2(i), 1, length(x_2)), x_2');
end


pi_2 = 4 * w_2'* f_val * w_2;

%% Problem 3

[x_3, w_3] = qnwequi(50000, 0,1, 'N');
pi_3 = 4 * w_3' * (1 - x_3 .^ 2) .^ 0.5;


%% Problem 4

[x_4, w_4] = weights(25000, 0, 1);
```

```
f_val2 = (1 - x_4 .^ 2) .^ 0.5;
pi_4 = 4 * w_4' * f_val2;


%% Problem 5

values = zeros(6,2);

% remmeber f = @(x, y)(double(x .^ 2 + y .^ 2 <= 1));
% basically run for every case again, for each number of iterations


it = [100, 1000, 10000];


for i = 1 : length(it)
    [a, b] = qnwequi(it(i), [0,0], [1,1], 'N');
    test1 = a(:,1) .^ 2 + a(:,2) .^ 2 <= 1;
    values(1, i) = 4 * b' * test1;

end

for i = 1 : length(it)

    [a, b] = qnwequi(it(i), 0, 1, 'N');
    values(2, i) = 4 * b' * (1 - a .^ 2) .^ 0.5;

end


for i = 1 : length(it)

    [a, b] = weights(it(i), 0, 1);
    f_val1 = zeros(length(a), length(a));

    for j = 1 : length(a)
    f_val1(j, :) = f(repmat(a(j), 1, length(a)), a');
    end

    values(3, i) = 4 * b' * f_val1 * b;

end
```

```
for i = 1 : length(it)

    [a, b] = weights(it(i), 0, 1);
    f_val3 = (1 - a .^ 2) .^ 0.5;
    values(4, i) = 4 * b' * f_val3;

end



% do the random generation from a seed
seed = 12345;
rng(seed);
pi_15 = zeros(200, length(it));
pi_25 = zeros(200, length(it));

for i = 1 : 200
    for j = 1 : length(it)

        tic;
        r1 = rand(it(j), 1);
        r2 = rand(it(j), 1);
        f_val4 = zeros(it(j), 1);

        for k = 1 : it(j)
            f_val4(k) = mean(f(repmat(r1(k), it(j), 1), r2));
        end

        pi_15(i, j) = 4 * mean(f_val4);
        pi_25(i, j) = 4 * mean((1 - r1 .^ (2)) .^ 0.5);
    end

    time = toc;
    fprintf('Iteration: %d Time: %f\n', i, time);

end

results = (values - pi) .^ 2;
pi_1_error = pi_15 - pi;
pi_2_error = pi_25 - pi;


for i = 1 : length(it)

    results(5, i) = mean(pi_1_error(:, i) .^ 2);
```

```
    results(6, i) = mean(pi_2_error(:, i) .^ 2);
end
```