Homework #4

Carlos Rangel

Question 1

Write an algorithm that uses the dart-throwing method to approximate pi using a quasi-Monte Carlo method

```
% Number of Points
npoints=10000000;

% Draw npoints Neiderrieter points on the unit square:

[points, weights] = qnwequi(npoints, [0 0], [1 1], ...
    'N');

% Evaluate the function at these points

fun_evals=4*ind_function(points);

% Average the function evaluations

pi_qmc=mean(fun_evals)
```

```
pi_qmc = 3.1416
```

Question 2

```
% Write an algorithm that uses the dart throwing ...
    method to approximate pi
% using a Newton-Cotes method

% We are going to use the trapezoid rule

% Number of nodes in the x direction
xnodes=10000;

% Number of nodes in the y direction
ynodes=10000;

% Obtain nodes and weights
[points, weights] = qnwtrap([xnodes ynodes], [0 0], ...
    [1 1]);
```

```
% Approximate pi
pi_nc=weights'*4*ind_function(points)
```

```
pi_nc = 3.1416
```

Question 3

```
% Approximate pi based on the above integral using ...
    a quasi-Monte Carlo
% approach

% Draw equidistributed sequence

% Number of Points
npoints=10000000;

% Draw npoints Neiderrieter points on the unit ...
    interval:
[points, weights] = qnwequi(npoints, 0, 1, 'N');

% Evaluate the function at these points

fun_evals=integrand(points);

% Average the function evaluations

pi_qmc=mean(fun_evals)
```

```
pi_qmc = 3.1416
```

Question 4

Write an algorithm that uses the dart-throwing method to approximate pi using a Newton-Cotes approach to computing integrals

```
% We are going to use the trapezoid rule

% Number of nodes in the x direction
xnodes=10000000;

% Obtain nodes and weights
[points, weights] = qnwtrap(xnodes, 0, 1);
```

```
% Approximate pi
pi_nc=weights'*integrand(points)
```

```
pi_nc = 3.1416
```

Question 5

Prepare a table which shows the mean-squared error of 200 simulations of pseudo-MC integration using 100, 1000, and 10,000 draws. Compare this to the squared error of the quasi-MC and Newton-Coates methods for the same number of quadrature draws (i.e. nodes).

For 100 draws

```
ndraws=100;

% Number of Simulations
ns=200;

% Vector for Storing integral estimates of each ...
    simulation
pi_mc_100=zeros(ns,1);

for i=1:ns
  % Draw 100 pseudo-random numbers
  [nodes, weights] = qnwequi(ndraws, [0 0], [1 1], ...
      'R');
  % Evaluate the pi approximation
  pi_mc_100(i)=weights'*4*ind_function(nodes);
end

% Mean Squared Error
mse_100= mean ( ( pi_mc_100 - pi ).^2 )
```

```
mse_100 = 0.0277
```

For 1000 draws

```
ndraws=1000;

% Number of Simulations
ns=200;

% Vector for Storing integral estimates
```

```
pi_mc_1000 = zeros(ns,1);

for i = 1:ns
  % Draw 1000 pseud0-random numbers
  [nodes, weights] = qnwequi(ndraws, [0 0], [1 1], ...
      'R');
  % Evaluate the indicator function at these nodes
  pi_mc_1000(i) = weights'*4*ind_function(nodes);
end

% Mean Squared Error
mse_1000 = mean( ( pi_mc_1000 - pi ).^2 )
```

```
mse_1000 = 0.0025
```

For 10,000 draws

```
ndraws = 10000;

% Number of Simulations
ns = 200;

% Vector for Storing integral estimates
pi_mc_10000 = zeros(ns,1);

for i = 1:ns
  % Draw 100 pseudo-random numbers
  [nodes, weights] = qnwequi(ndraws, [0 0], [1 1], ...
      'R');
  % Evaluate the indicator function at these nodes
  pi_mc_10000(i) = weights'*4*ind_function(nodes);
end

% Mean Squared Error
mse_10000 = mean( ( pi_mc_10000 - pi ).^2 )
```

```
mse_10000 = 2.8866e-04
```

Calculate squared errors for quasi-Monte Carlo and Newton-Coates

```
% Vector for Storing Squared Errors of quasi-Monte ...
    Carlo
sqe_qmc = zeros(3,1);
```

```
% First do quasi-Monte Carlo with 100 nodes
nodes=100;

% Starting Position for storing Squared Errors
i=1;

while nodes<=10000

 % Draw npoints Neiderrieter points on the unit ...
    square:
 [points, weights] = qnwequi(nodes, [0 0], [1 1], ...
    'N');

 % Evaluate the function at these points
 fun_evals=4*ind_function(points);

 % Obtain pi approximation
 pi_qmc=mean( fun_evals );

 % Calculate the squared error of the current ...
    integration approximation
 sqe_qmc(i) = (pi_qmc-pi)^2;

 % Increase the number of nodes by an order of ...
    magnitude
 nodes=10*nodes;

 % Increase counter variable i
 i=i+1;

end
```

Squared Errors for Newton-Coates

```
% Vector for Storing Squared Errors of Newton-Coates
sqe_nc=zeros(3,1);

% First do Newton-Coates with 100
nodes=100;

% Starting position for storing squared errors
i=1;
```

```
while nodes <= 10000

 % Number of nodes in the x direction
 xnodes = nodes;
 % Number of nodes in the y direction
 ynodes = nodes;

 % Obtain nodes and weights
 [points, weights] = qnwtrap([xnodes ynodes], [0 ...
     0], [1 1]);

 % Approximate pi
 pi_nc = weights'*4*ind_function(points);

 % Calculate the squared error of the current ...
     integration approximation
 sqe_nc(i) = ( pi_nc - pi )^2;

 % Increase the number of nodes by an order of ...
     magnitude
 nodes = 10*nodes;

 % Increase counter variable
 i = i + 1;

end
```

Display Tables comparing Mean Squared Errors

```
mse_mc = [mse_100; mse_1000; mse_10000];

varNames = {'MonteCarlo', 'QuasiMonteCarlo', ...
    'NewtonCoates'};

rowNames = {'100', '1000', '10000'};
disp('Mean Squared Error Comparison');
```

```
Mean Squared Error Comparison
```

```
MSE = table(mse_mc, sqe_qmc, sqe_nc, ...
    'VariableNames', varNames, 'RowNames', rowNames)
```

	MonteCarlo	QuasiMonteCarlo	NewtonCoates
1 100	0.0277	4.6624e-04	1.3305e-05
2 1000	0.0025	2.5365e-06	2.4940e-08
3 10000	0.0003	6.2830e-07	1.1405e-11