UNIVERSITY OF LIVERPOOL

# Experiment 22 - Monte Carlo Simulation

ELEC273*

October 31, 2019


**Abstract**

This report introduced the results and discussion of the experiment which aiming to explore the features and applications of Monte Carlo method. A penalty kicking model was established wit Matlab with random kicks following uniform distribution and Gaussian distribution. The situation with no goalkeeper, uniformly acted goalkeeper and goalkeeper with special strategy were considered. In the discussion section, the pro and con of Monte Carlo method, the influence of changing of standard deviation in Gaussian cases, and some possible suggestions for penalty kicks participants were studied.


**Declaration**

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with ant other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

*IMPORTANT: In a standard technical report, you would need to include here your personal details as the author of the document. However, remember that marking of coursework is anonymous and therefore you should remove this part before submitting your report for Year 2 labs! Do not include your name, student ID, email address or any other personal information.

# Contents

# 1 Introduction

Monte Carlo method was a method to solve problems that can be transformed to statistical problems. The theory of Monte Carlo method is to utilize random variables to simulate statistic problems and apply means to approach the theoretical result. This method was first put forwarded by scientists who worked for producing the first nuclear weapon in the world. The Monte Carlo method introduce procedures to simulate a realistic problem step by step. The first step is to design a set of random samples of size N from according probability distribution methods. Secondly, obtain the values of this part by implementing respective methods. Thirdly, calculate the value wanted from this set of samples. Fourthly, repeat the above steps for R times and examine the actual value with these partial results. [1]

Monte Carlo method was widely used in physics, computing Science, finance, telecommunications and games etc.. With the development of computer technology, the effective and efficiency of the application of Monte Carlo method were considerably improved, which indicated more general value of this method for modern industry. It was significant to explore and acknowledge some basic features and usages of Monte Carlo method.

# 2 Materials and Methods

To easily simulate penalty kicks events, a simple goal model was established according to the instruction on the lab script[1]. In this experiment, Matlab was utilized to modeling the situation with pseudo random number generated with Monte Carlo algorithm.

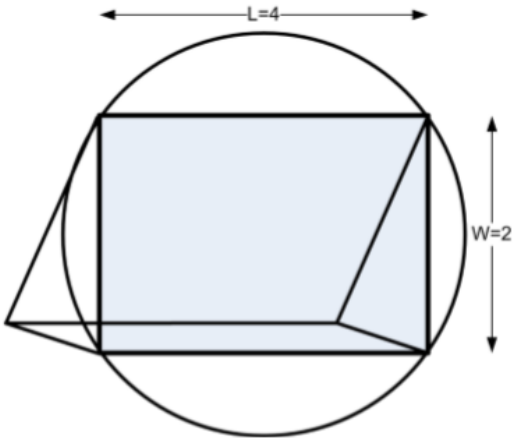


Figure 1: Goal Model (taken from [1])

It is assumed that the ball can be kicked randomly and will only fall in the circle area as it shown in Figure 1. The distribution of balls is expected to follow certain distribution model, such as uniform distribution and normal distribution. In order to explore the distinctions between among different distribution models under diverse conditions, this experiment was designed into two parts. The first part was for the situation with no goalkeepers while the second part focused on the condition that the goalkeeper acted.

### 2.1 Part 1: No Goalkeeper Tests

In this part, the situation with no goalkeeper was modeled. Firstly, the theoretical result of the hit rate was calculated under the assumption that the distribution of balls follows uniform distribution. Secondly, Matlab was utilized to simulate this process with a uniform random number generator. Thirdly, the influence of simulating shots number N and experiment repeat times R was studied with perspectives of line charts generated with Matlab. Furthermore, the hypothesis was changed to that the distribution of balls abides by Gaussian distribution and the experiment steps above were repeated for this new assumption.

### 2.2 Part 2: With Goalkeeper Tests

Figure 2: Goalkeeper Action to Penalty Shoot-outs (taken from [1])

Under the condition that there are goalkeeper, this football goal model can be further improved. To easily analyse the situations, the action of the goalkeeper was classified into five types ideally, as shown in Figure 2. Firstly, it was considered that the possibility of these five cases was equal following uniform random distribution. Matlab was used to simulate the shooting process and then the hit rate can be figured out. Secondly, similarly with the produce in Part 1, the hypothesis was changed from uniform random distribution to Gaussian random distribution and the simulation was implemented again. Thirdly, the actions of goalkeeper was updated from equally distributed to that 90% of the time the goalkeeper may choose position 4 and position 5. Under this circumstance, the probability of scoring was required to be simulated with kicking 100 and 1000 balls.

## 3 Results

### 3.1 Part 1: No Goalkeeper Tests

This section presents and comments the result from the simulation with no goalkeeper.

#### 3.1.1 Task 1: Calculation

In this subsection, the theoretical value of scoring probability will be calculated under the condition that balls distributed in the circle with a radius of $\sqrt{5}$ (as shown in Figure 1) uniformly. In this case, the hit rate can be obtained through the fraction of the goal area and the circle area presented in Equation 1.

$$P_{scoring} = \frac{S_{rect}}{S_{circle}}. \tag{1}$$

As the area of the rectangle goal is 8 and the area of the circle zone is $5\pi$, the possibility then can be computed, which is 51.0%.

### 3.1.2 Task 2&3: Matlab Simulation

To simulate a uniform random kicking situation, Matlab was utilized to generate virtual kicking and display the distribution in illustration. The code was shown in Listing 2 in Appendix. This code achieved the requirement of both Task 2 and Task 3, which allowed user to input the shots times N and experiment repeat times R and then the possibility of scoring would be given out and a illustration of shotting will be presented.

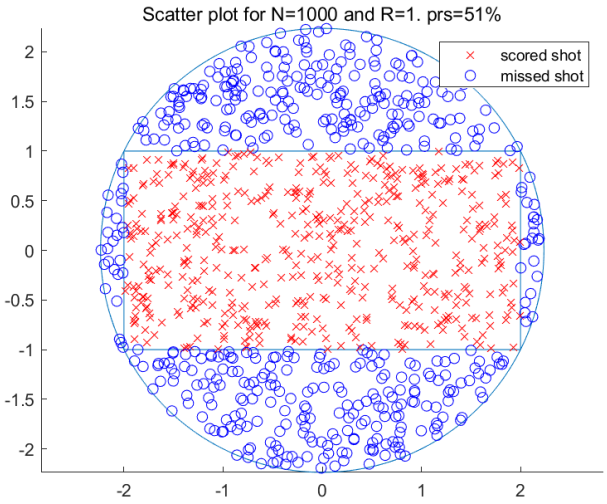


Figure 3: Shotting Illustration with N=1000 & R=1 (Task 3)

Run the code, with the input of $N = 1000$ and $R = 1$, it was indicated that the scoring possibility was approximate 51% and the generated shotting illustration was shown in Figure 3.

### 3.1.3 Task 4: Perspective Towards N

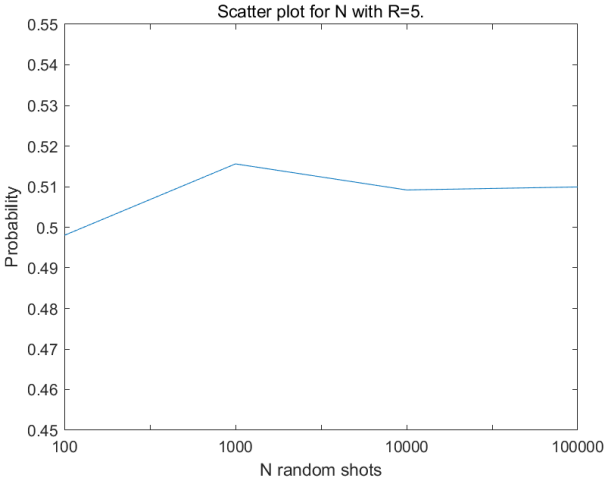


Figure 4: Line Chart of Probability against N with R=5 (Task 4)

With the code in Listing 3, a line chart of scoring probability against N can be obtained. As it presented in Figure 4, the value of the probability approached 51% when the N increased.

This may indicate that, with the shotting times increasing, the experimental value could become more stable and approach the theoretical value, which is 51.0% as it mentioned in Task 1 in this case.

### 3.1.4 Task 5: Perspective Towards R

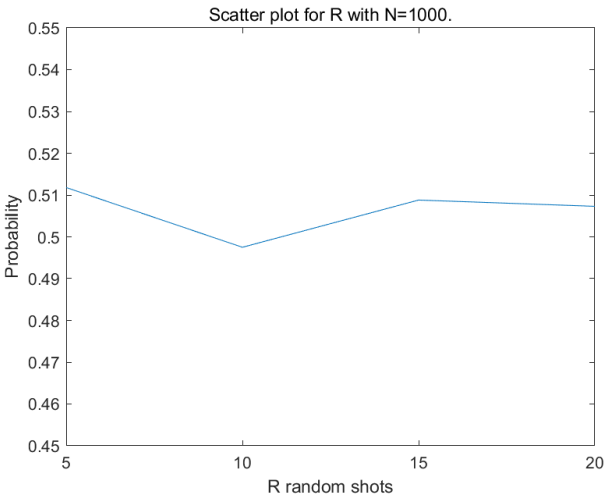


Figure 5: Line Chart of Probability against R with N=1000 (Task 5)

By practicing the source code of Listing 4 in Appendix, the probability against R with $N = 1000$ was explored. As it shown in Figure 5, similar with the result in Task 4, the probability became stable at 51% when the experiment repeat more times. However, this phenomenon was not so obvious as it shown in Task4. One reason for this is that the increasing range of Task 4 was around 100000 but the increasing for Task 5 is only 20.

### 3.1.5 Task 6: Compare R and N

From Task4 and Task 5, it was noticed that with the increasing of total shotting times, the probability of scoring would become stable and approach 51.0%, which is its theoretical value. It was also mentioned that the reason why the tendency of stabilisation in Task 5 was not as obvious as Task 4 was that Task 4 shotting considerably more times than Task 5. In this case, it can also be indicated that the increasing of R and N can both promote the accuracy of the experimental probability.

### 3.1.6 Task 7: Gaussian Distribution

In this case, it was required to substitute the distribution generation method from uniform random distribution to Gaussian random distribution. With a smart design of the program architecture, the only thing needed was to change the method (function) parameters in the code of Task 2&3, Task 4, Task 5 from unifrnd_circle to normrnd_circle.

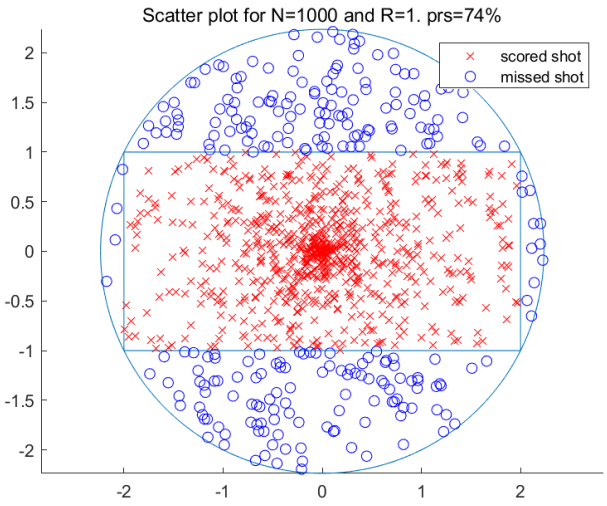


Figure 6: Shotting Illustration with N=1000 & R=1 (Task 7_3)

Using the code in Listing 5, Task 2 and Task 3 were repeated as required. Figure 6 illustrated the same simulation with Task 2 with Gaussian method. Comparing with the uniform situation, the distribution in this case seemed to be more centralized at the center of the goal. Inevitably, the probability increased to 74% in this case, which was considerable large than the uniform one.

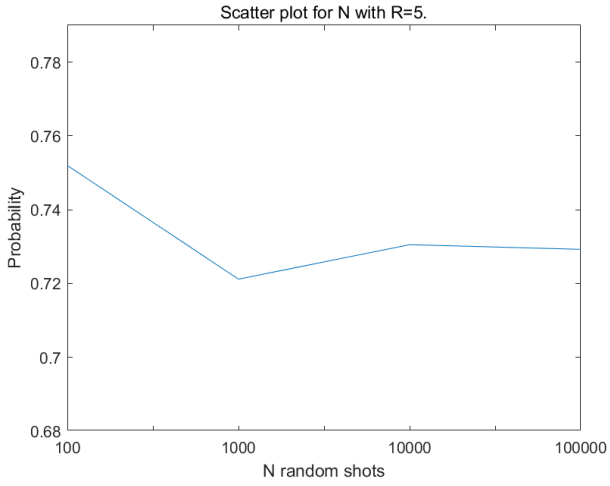


Figure 7: Line Chart of Probability against N with R=5 (Task 7_4)

Listing 6 presented the source code for the repeat of Task 4. By running Listing 6, Figure 7 which explored the relation between scoring probability and shooting times can be generated. Similar with the uniform distribution, with the shotting times increasing, the probability become stable at 73%, which was different from the value 51% in uniform case.

Listing 7 displayed the Matlab code for the exploration of the relation between the probability and experiment repeat times R. As the result shown in Figure 8, even if the increasing of R was slowly, it can also be figured out that the probability was trying to approach around 73%.

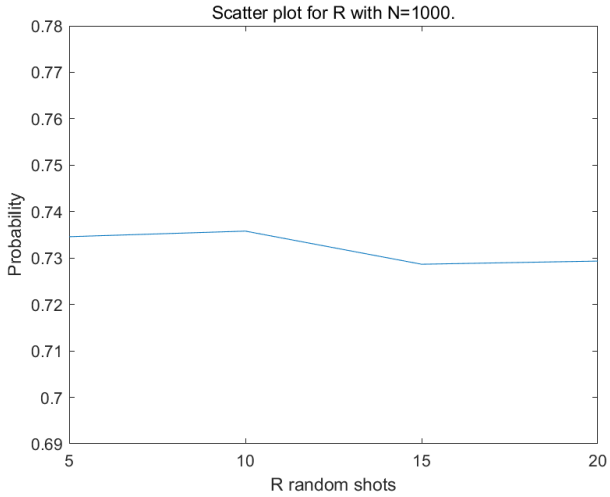


Figure 8: Line Chart of Probability against R with N=1000 (Task 7_5)

## 3.2 Part 2: With Goalkeeper Tests

This section presents and comments the result from the simulation with goalkeeper.

### 3.2.1 Task 8: Uniform Case

As the goalkeeper in this case was exist, several functions such as uniform_5case (Listing 18) was designed to simulate the action of the goalkeeper and one function isGoalKept (Listing 15) was utilized to judge whether a ball was scored. With a Matlab project architecture with high reusability, the addition of goalkeeper can be easily achieved by simply changing several parameters as it shown in Listing 8.

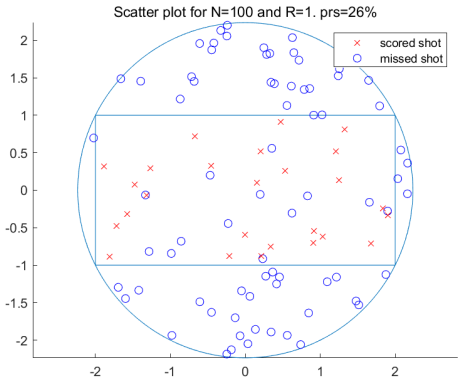


Figure 9: Scored Balls Distribution Simulation for R=100

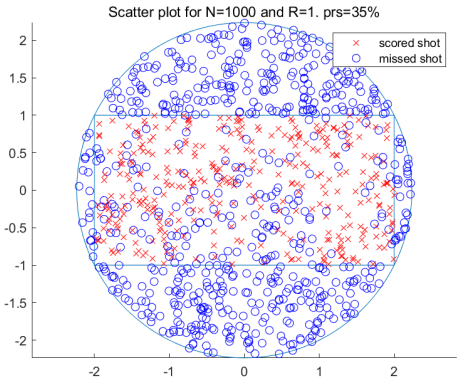


Figure 10: Scored Balls Distribution Simulation for R=1000

Figure 9 and Figure 10 displayed the situation of $R = 100$ and $R = 1000$ correspondingly. It can be noticed that the scoring probability for 100 and 1000 kicks were 26% and 35%. After repeating the case of $R = 100$ for several times, it was observed that the value of probability was shaken around 35%, thus it was explained that as the kicks increased, the scoring probability could approach 35%. Compared with the scoring probability 51% in Task 1 and Task 3, it can be figured that , with the same distribution method, the scoring probability considerably decreased after adding a goalkeeper.

### 3.2.2 Task 9: Gaussian Case

By substituting the distribution method parameter from Function unifrnd_circle (Listing 20) to Function normrnd_circle (Listing 17) as shown in Listing **??**, the behavior of the kickers can be switched to Gaussian mode.

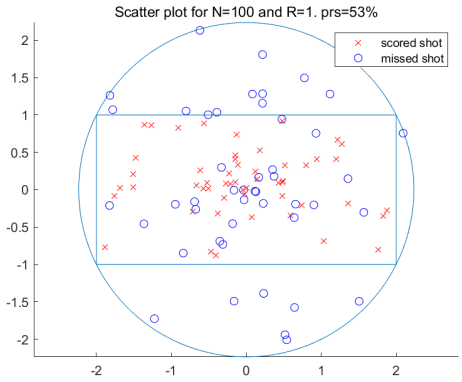


Figure 11: Scored Balls Distribution (Gaussian) Simulation for R=100

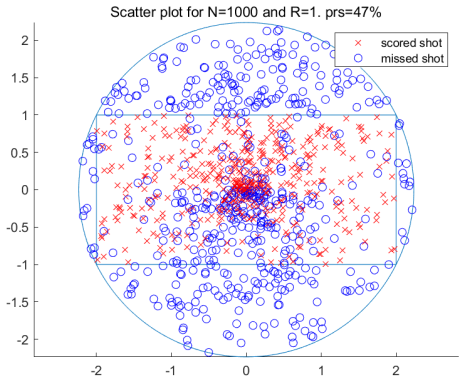


Figure 12: Scored Balls Distribution (Gaussian) Simulation for R=1000

The simulated scoring probability for Gaussian situation was 53% and 47% for $R = 100$ and $R = 1000$ correspondingly. With the increasing of the kicks, the probability approached 47.3%. The distribution illustrations were shown as Figure 11 and 12 respectively. Comparing with the result in Task 8, the concentration of the kicks was higher than the balls distribution in Task 8, which is also the distinct between uniform distribution and Gaussian distribution. Therefore, the final scoring probability was improved as more kicks fallen into the goal. Different from the result in Task 7 with no goalkeeper, the scoring probability was considerable decreased since a considerable number of kicks in the goal were blocked by the goalkeeper. This explained the drop down of the result from Task 7 to this Task.

### 3.2.3 Task 10: Smarter Goalkeepers

In this part, the goalkeeper was required to be simulated to be smarter. For technique reasons, it was assumed that the goalkeeper tended to choose the last two cases in Figure 2 with 90%. To achieve this, a uniform_5case_plus (Listing 19) Function was designed to substitute the former uniform_5case (Listing 18) Function to Listing **??** and Listing **??**. After the modification, the result can be generated with Function unifrnd_circle (Listing 20) in uniform case and Function normrnd_circle (Listing 17) in Gaussian case.

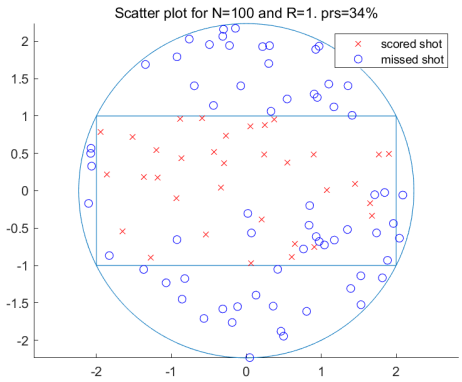


Figure 13: Scored Balls Distribution Simulation for R=100

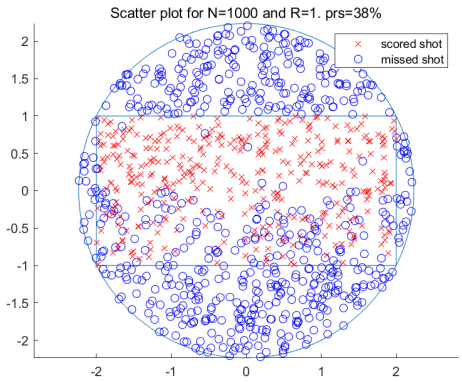


Figure 14: Scored Balls Distribution Simulation for R=1000

Figure 13 and Figure 14 presented the result of 100 kicks and 1000 kicks with uniform distribution, which were 37% and 38% correspondingly. Comparing with the result 26% and 35% in Task 8, it was observed that the scoring probability was increased since the goalkeeper only tended to block the kicks in the 4&5 cases, which may mean that there were more scored kicks on the upper and middle area.

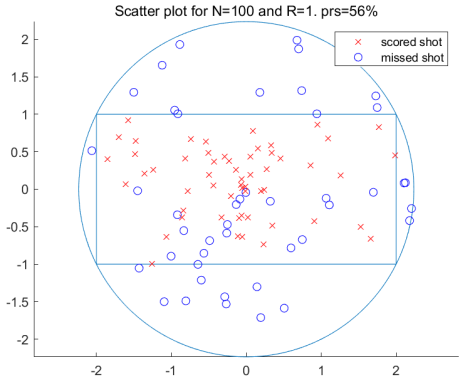


Figure 15: Scored Balls Distribution (Gaussian) Simulation for R=100

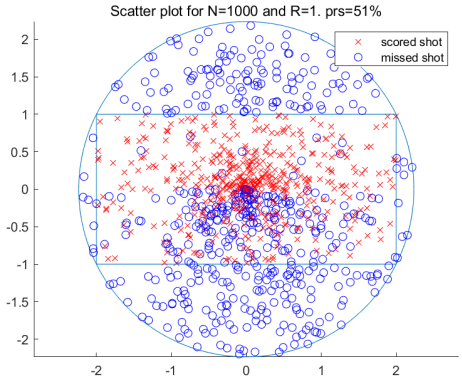


Figure 16: Scored Balls Distribution (Gaussian) Simulation for R=1000

Figure 15 and Figure 16 displayed the Gaussian method simulation result with the kicks of 100 and 1000, which were 56% and 51%. Unlike to the result 53% and 47% in Task 9, it can also be indicated that the scoring probability was decreased comparing to the situation that the goalkeeper uniformly randomly acted. The reason for this was considered quite similar to the reason mentioned in the last paragraph. Nevertheless, the result should be slight different as the distribution was changed to Gaussian distribution thus this could give rise to a distinct increased value of the final result, which was not clearly indicated by the simulation.

# 4 Discussion

In terms of the advantages of Monte Carlo method, from this experiment, four advantages can be concluded. Firstly, with Monte Carlo method, a complex math process, such as the calculation of the scoring probability in Gaussian distribution, can be easily achieved through a computer. Secondly, with the increasing of the experiment repeat times, the error of the result can be

considerably reduced. Furthermore, it was flexible when using Monte Carlo method to balance the accuracy of the result and the time consumed on computing, which means that, it would be possible to obtain a rough result in a short time in some situation. Additionally, nearly all problems which can be transformed to statistic problems can be approached via Monte Carlo method. **Q1**

The drawbacks of Monte Carlo method were mainly classified into the following three points. Firstly, a considerable number of computing resources and time are required when using Monte Carlo method to obtain a accuracy result. Secondly, the error generated by the Monte Carlo method is probability error and can not be effectively removed. Moreover, as the generation process is simulated artificially, Monte Carlo method seems to be limited in some periodical application and may give rise to generate same random values. **Q1**

Although the model in this experiment has fully considered the situation of the kicks and the actions of the goalkeeper, there are still several improvement can be proposed. Firstly, the reaction of the goalkeeper could be designed according to the kicking direction of the kicker. Secondly, the effect from the environment such as the wind and the sun, which may dazzling the goalkeeper, can also be considered and influence the action of the kickers and the goal keeper. Besides, the behavior classification of the goalkeeper can be further divided. Furthermore, it is also believed that even the goalkeeper choose the right action, it is still possible that the ball finally go through the goal and scores. **Q2**

After changing the standard deviation of the Gaussian distribution in Task 7 and Task 9, it was observed that the result of scoring probability would change. For both Task 7 and Task 8, when the standard deviation become smaller, the kicks will become more concentrated to the center of the goal, which indicates increasing precision and not changing accuracy. **Q3**

To find the relation of estimating the value of $\pi$, from Equation 1, the following Equation can be obtained. **Q4**.

$$P_{scoring} = \frac{WL}{\pi R^2}. \tag{2}$$

From Equation 2, the relation of $\pi$ and $P_{scoring}$ can be conducted as the following equation.

$$\pi = \frac{WL}{P_{scoring} R^2}. \tag{3}$$

After input the result of scoring probability 51% in Task 2 to Equation 3, the value of $\pi$ can be calculated, which is 3.137. For the result in the Gaussian case, after the value 74% is input, the $\pi$ is calculated to be 2.16, which is quite far from the right value. The reason for this is that the area of the circle which contains the $\pi$ is formed uniformly, which is similar to the definition of uniform distribution but different from the Gaussian distribution. In a word, the ununiformity of Gaussian distribution causes this deviation. **Q4**

From the result of Part 2 simulation (Figure 10, 12, 14, 16), it is advised to the penalty takers that the scoring opportunity could be higher if they kick towards the upper left corner and upper right corner. As for the goalkeepers, it is suggested that they should protect the corner of the goal more and not just focus on several special action modes, which means that, try to protect the area of the goal as much as possible. **Q5**

# 5 Conclusions

In this experiment, features and application of Monte Carlo method were explored and discussed with the assistance of Matlab. Penalty kicks were simulated with the distribution of kicks following uniform distribution and Gaussian distribution and the goalkeeper with different behaviors. It was further discussed about the advantages and limitation of Monte Carlo method as well as some suggestions against the penalty takers and the goalkeepers on the basis of the modeling result.

# References

[1] W Al-Nuaimy, A Al-Ataby, M Lopez-Benitez, "Experiment 22 - monte carlo simulation," https://vital.liv.ac.uk/bbcswebdav/pid-2008619-dt-content-rid-11366223_1/xid-11366223_1, University of Liverpool, 2019.

# Appendices

## A Matlab Source Code

### A.1 Overall Structure

This is the source code files structure instruction, which displays the list of main code files and function files.

Listing 1: Source Code Files Structure

```
src
|
|---main code
|    |---t2.m
|    |---t4.m
|    |---t5.m
|    |---t7_2.m
|    |---t7_4.m
|    |---t7_5.m
|    |---t8.m
|    |---t9.m
|    |---t10_8.m
|    |---t10_9.m
|
|
|---func
|    |---drawBackGround.m
|    |---getDisProb.m
|    |---getDivByPos.m
|    |---isGoalKept.m
|    |---isInRect.m
|    |---normrnd_circle.m
|    |---uniform_5case.m
|    |---uniform_5case_plus.m
|    |---unifrnd_circle.m
|
```

### A.2 Main Codes

#### A.2.1 Task 2&3

Listing 2: Matlab Source Code for Task 2

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");
```

```
% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @unifrnd_circle);

% add legend
legend([d_rx, d_bo], {'scored shot', 'missed shot'});
t_s = sprintf('Scatter plot for N=%d and R=%d. prs=%d%%', N, R, round(prs*100));
title(t_s);

display(prs);
```

### A.2.2 Task 4

Listing 3: Matlab Source Code for Task 4

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% declear params
N = [100, 1000, 10000, 100000];
R = 5;

% declear var
array_prs = [];

% calculate prs
for i = 1 : numel(N)
    [score, totalTimes, prs] = getDisProb(R, N(i), Radius, Length,...
        Width, false, @unifrnd_circle);
    array_prs = [array_prs, prs];
end

disp(array_prs);

%plot line chart
plot([1,2,3,4], array_prs);
ylim([0.45 0.55]);
xlabel('N random shots');
ylabel('Probability');
xticklabels({'100','','1000','','10000','','100000'});
t_s = sprintf('Scatter plot for N with R=%d.', R);
title(t_s);
```

### A.2.3 Task 5

Listing 4: Matlab Source Code for Task 5

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% declear params
N = 1000;
R = [5, 10, 15, 20];

% declear var
array_prs = [];

% calculate prs
for i = 1 : numel(R)
    [score, totalTimes, prs] = getDisProb(R(i), N, Radius, Length,...
        Width, false, @unifrnd_circle);
    array_prs = [array_prs, prs];
end

disp(array_prs);

%plot line chart
plot(R, array_prs);
ylim([0.45 0.55]);
xlabel('R random shots');
ylabel('Probability');
t_s = sprintf('Scatter plot for R with N=%d.', N);
title(t_s);
```

### A.2.4 Task 7

Listing 5: Matlab Source Code for Task 7_2

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");

% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @normrnd_circle);
```

```
% add legend
legend([d_rx, d_bo], {'scored_shot', 'missed_shot'});
t_s = sprintf('Scatter_plot_for_N=%d_and_R=%d._prs=%d%%', N, R, prs*100);
title(t_s);

display(prs);
```

Listing 6: Matlab Source Code for Task 7_4

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% declear params
N = [100, 1000, 10000, 100000];
R = 5;

% declear var
array_prs = [];

% calculate prs
for i = 1 : numel(N)
    [score, totalTimes, prs] = getDisProb(R, N(i), Radius, Length,...
        Width, false, @normrnd_circle);
    array_prs = [array_prs, prs];
end

disp(array_prs);

%plot line chart
plot([1,2,3,4], array_prs);
ylim([0.68 0.79]);
xlabel('N_random_shots');
ylabel('Probability');
xticklabels({'100','','1000','','10000','','100000'});
t_s = sprintf('Scatter_plot_for_N_with_R=%d.', R);
title(t_s);
```

Listing 7: Matlab Source Code for Task 7_5

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% declear params
N = 1000;
R = [5, 10, 15, 20];
```

```
% declear var
array_prs = [];

% calculate prs
for i = 1 : numel(R)
    [score, totalTimes, prs] = getDisProb(R(i), N, Radius, Length,...
        Width, false, @normrnd_circle);
    array_prs = [array_prs, prs];
end

disp(array_prs);

%plot line chart
plot(R, array_prs);
ylim([0.69 0.78]);
xlabel('R random shots');
ylabel('Probability');
t_s = sprintf('Scatter plot for R with N=%d.', N);
title(t_s);
```

### A.2.5 Task 8

Listing 8: Matlab Source Code for Task 8

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");

% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @unifrnd_circle, @uniform_5case);

% add legend
legend([d_rx, d_bo], {'scored shot', 'missed shot'});
t_s = sprintf('Scatter plot for N=%d and R=%d. prs=%d%%', N, R, round(prs*100));
title(t_s);

display(prs);
```

### A.2.6 Task 9

Listing 9: Matlab Source Code for Task 9

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");

% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @normrnd_circle, @uniform_5case);

% add legend
legend([d_rx, d_bo], {'scored shot', 'missed shot'});
t_s = sprintf('Scatter plot for N=%d and R=%d. prs=%d%%', N, R, round(prs*100));
title(t_s);

display(prs);
```

### A.2.7 Task 10

Listing 10: Matlab Source Code for Task 10_8

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");

% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @unifrnd_circle, @uniform_5case_plus);

% add legend
legend([d_rx, d_bo], {'scored shot', 'missed shot'});
t_s = sprintf('Scatter plot for N=%d and R=%d. prs=%d%%', N, R, round(prs*100));
title(t_s);

display(prs);
```

Listing 11: Matlab Source Code for Task 10_9

```
% init
clear
addpath(genpath('./func/'));

% declear const
Radius = sqrt(5);
Width = 2;
Length = 4;

% params input
N = input("Shots Times: ");
R = input("Repeating Times: ");

% draw background
drawBackGround(Radius, Width, Length);

% calculate prs
[score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius, Length,...
    Width, true, @normrnd_circle, @uniform_5case_plus);

% add legend
legend([d_rx, d_bo], {'scored shot', 'missed shot'});
t_s = sprintf('Scatter plot for N=%d and R=%d. prs=%d%%', N, R, round(prs*100));
title(t_s);

display(prs);
```

## A.3 Function Codes

### A.3.1 Draw

Listing 12: Matlab Source Code for Function drawBackGround

```
function drawBackGround(Radius, Width, Length)
% prepare figure
hold on
    % - draw circle
d_ang = 0 : pi/100 : 2*pi;
plot(Radius*cos(d_ang), Radius*sin(d_ang), '-');
    % - draw rect
line([
    -Length/2, -Length/2, Length/2, Length/2, -Length/2
], [
    -Width/2, Width/2, Width/2, -Width/2, -Width/2
]);

axis equal
end
```

### A.3.2 Math

Listing 13: Matlab Source Code for Function getDisProb

```
% compute distributation
function [score, totalTimes, prs, d_rx, d_bo] = getDisProb(R, N, Radius,...
    Length, Width, isPlot, disMethod, goalkeepMethod)
score = 0;
totalTimes = 0;
for m = 1 : R
    for n = 1 : N
        [x, y] = disMethod(Radius);
        totalTimes = totalTimes + 1;
        if isInRect(Length/2, Width/2, x, y) && (nargin<8 || ~isGoalKept(Length,...
                Width, x, y, goalkeepMethod))
            score = score + 1;
            if isPlot
                d_rx = plot(x, y, 'rx');
            end
        else
            if isPlot
                d_bo = plot(x, y, 'bo');
            end
        end
    end
    prs = score / totalTimes;
end
```

### A.3.3 Tools

Listing 14: Matlab Source Code for Function getDisProb

```
function res = getDivByPos(Length, Width, x, y)

    ux=Length/4;
    uy=Width/2;

    x = x + Length/2;
    y = y + Width/2;

    px = ceil(x/ux);
    py = floor(y/uy);

    res = px + py*4;
end
```

Listing 15: Matlab Source Code for Function isGoalKept

```
function res = isGoalKept(Length, Width, x, y, goalkeepMethod)
    div = getDivByPos(Length, Width, x, y);
    act = goalkeepMethod();

    pattern = [2, 3, 6, 7; 2, 5, 0, 0; 3, 8, 0, 0; 1, 2, 0, 0; 3, 4, 0, 0];

    if ismember(div, pattern(act, :))
        res = true;
    else
        res = false;
    end

end
```

Listing 16: Matlab Source Code for Function isInRect

```
% judge if a position in a rectangle
function res = isInRect(x, y, x_, y_)
    if abs(x_) < x && abs(y_) < y
        res = true;
    else
        res = false;
    end
end
```

### A.3.4 Distribution Methods

Listing 17: Matlab Source Code for Function normrnd_circle

```
% get random position in circle
function [res_x, res_y] = normrnd_circle(r)

    while true
        ang = 2*pi*rand(1);
        t_r = normrnd(0, r);
        res_x = t_r*cos(ang);
        res_y = t_r*sin(ang);
        if sqrt(res_x^2 + res_y^2) < r
            break;
    end

end
```

Listing 18: Matlab Source Code for Function uniform_5case

```
function res = uniform_5case()

    res = randi(5, 1, 1);

end
```

Listing 19: Matlab Source Code for Function uniform_5case_plus

```
function res = uniform_5case_plus()

    if rand(1) <= 0.9
        res = randi([4, 5], 1, 1);
    else
        res = randi(3, 1, 1);
    end

end
```

Listing 20: Matlab Source Code for Function unifrnd_circle

```
% get random position in circle
function [res_x, res_y] = unifrnd_circle(r)

    while true
        res_x = unifrnd(-r, r, 1, 1);
        res_y = unifrnd(-r, r, 1, 1);
        ans_r = sqrt(res_x^2 + res_y^2);
```

```
        if ans_r < r
            break
        end
    end

end
```