UNIVERSITY OF LIVERPOOL

Experiment 22 - Monte Carlo Simulation

ELEC273*

October 30, 2019

Abstract

This document describes what is expected from an engineering laboratory report. While this is not a definitive guide to report-writing, it is intended as a guide to assist in documenting and presenting your experimental work. Appended to this guide are some real examples of common mistakes that should be avoided. It is expected that lab reports are prepared using the LaTeX typesetting system, adopting pne of the standard report templates provided on VITAL.

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own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web).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

Include a statement of the problem to be investigated, and why is addressing this problem worthwhile or important. This section should also introduce the history and theoretical background of the problem, a brief statement of the general procedure adopted and hint at expected results. This section is sometimes broken down into subsections: Objectives and Theoretical Background.

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.

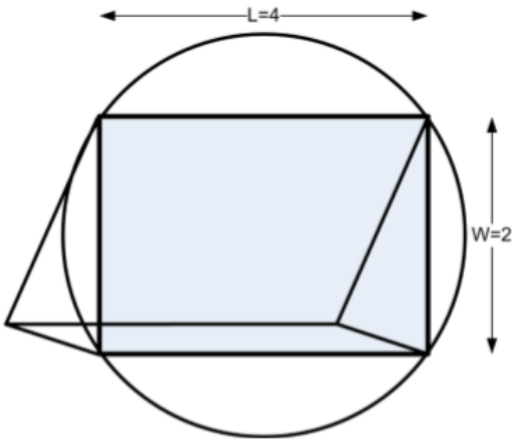

Figure 1: Goal Model (taken from [1])

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 5π, the possibility then can be computed, which is 51.0%.

3.1.2 Task 2: Matlab Simulation

This is where you describe the important qualitative and quantitative observations from the experiment. Data should be tabulated and/or graphed. Graphs must always be captioned (as with figures, above), and axes clearly annotated with units. Always follow your graphs with a brief description, demonstrating your understanding of what has been plotted (eg. "The graph in Fig. 4 illustrates the frequency response of this circuit, with a clear resonance peak at 15 kHz.") as tables and graphs are rarely self-explanatory. If the results differ from what was expected, explain both **how** (i.e. in what way) and **why** (i.e. possible reasons behind this behaviour) when referring to these results.

The summary of the experimental results is presented in Table 1

Table 1: Note that table captions are placed above the tables, not below as for figures

P (W)	V (V)	I (A)
0.0 ± 0.01	0.0 ± 0.01	0.0 ± 0.01
0.1	0.2	0.3
0.1	0.2	0.3
0.1	0.2	0.3

Table 2: Resistance and Temperature of the Filament

R (Ω)	T (K)	$1/T$ (K^{-1})	$\ln P$
151.00±3.92	828.35±23.46	1.2072×10^{-3}	-13.29
157.12±3.71	856.88±22.25	1.1671×10^{-3}	-12.64
162.53±3.49	881.99±21.02	1.1338×10^{-3}	-12.33
166.67±3.33	901.14±20.13	1.1097×10^{-3}	-11.90

4 Discussion

The Discussion section allows you to fully discuss and interpret the results and the results of the any analysis carried out. It is also important here to relate your findings to the experimental objectives, i.e. do your results support the theoretical background, and have the objectives of the experiment been met? Are the results reliable and/or significant?

Mathematical expressions can be included in LaTeXdocuments easily by surrounding them in `$` signs, such as f, δx, Ω_2, $y = mx + b$ or $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$. Numbered equations can be created just as easily using the `equation` environment:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2}$$

When making reference to the above equation, you simply need to mention the label, as in "`...as shown in Equation~\ref{eq:mean}.`", and the Equation number will be inserted automatically. If you need to show several lines of mathematical expressions, you can use the `eqnarray` environment as below:

$$\begin{aligned} P &= IV && (3)\\ &= I^2R && (4)\\ &= \frac{V^2}{R} && (5) \end{aligned}$$

Further guidance on writing and manipulating mathematical expressions, symbols and equations can be found online in references such as [2].

5 Conclusions

This final section contains a brief statement to summarise the outcome of the experiment, and a statement outlining to what extent the objectives of the experiment have been met, and what has been learnt as a result of this. In some ways the Conclusions section is a reflection of the Abstract, reiterating what the experiment was aiming to achieve, and summarising the outcomes. It can also be thought of as a summary of the Discussion section.

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.

[2] WikiBooks, "LaTeX advanced mathematics," http://en.wikibooks.org/wiki/LaTeX/Advanced_Mathematics, 2011.

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, 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_circl
    array_prs = [array_prs, prs];
end

disp(array_prs);

plot(N, array_prs);
xlabel('N random shots');
ylabel('Probability');
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_circl
    array_prs = [array_prs, prs];
end

disp(array_prs);

plot(R, array_prs);
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_

% 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_circl
    array_prs = [array_prs, prs];
end

disp(array_prs);

plot(N, array_prs);
xlabel('N random shots');
ylabel('Probability');
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_circl
    array_prs = [array_prs, prs];
end

disp(array_prs);

plot(R, array_prs);
```

```
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_

% 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);
```

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_

% 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);
```

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_

% 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_

% 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,
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, g
            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
```