Hitting probabilities in 3-player betting games

Angel Yong He, supervised by Prof. Mark Holmes

angelh1@student.unimelb.edu.au | GitHub repo: https://github.com/7angel4/3-person-betting-game

2023/2024 Mathematics and Statistics Vacation Scholarships Program, The University of Melbourne

Introduction

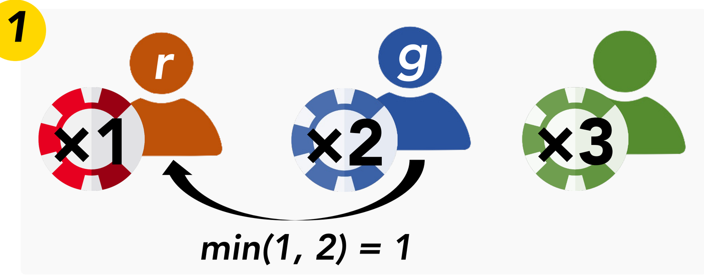
We study two variants of the models examined by Prof. Persi Diaconis [1]. Denote the players' fortunes at each round as $(X_n, Y_n, Z_n) \in \mathbb{N}^3$, which evolves as a Markov chain. Write $(X_0, Y_0, Z_0) = (x, y, z)$.

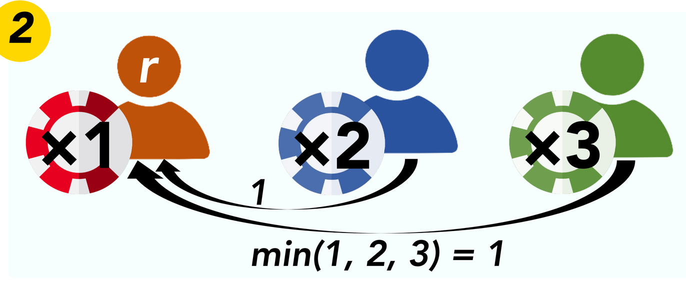
- *Game 1*: At each round, a giver and receiver are chosen at random. The giver transfers the minimum of their fortunes to the receiver.
- *Game 2*: Only a receiver is chosen each round to receive $\min(x, y, z)$.

Define the following terminology / representations:

- **Loser**: First player to reach 0 (if tied in Game 2, pick one randomly).
- **Winner**: First player to have all the money.
- $L_{(x,y,z)}$: Probability that player 1 loses, given initial state (x, y, z).

The research problem addressed is the *difficulty of attaining $L_{(x,y,z)}$ by hand*. This poster presents an overview of the program I produced in response, and some further analysis on the first-hand data.

1 

2 

Problem Specification

While $P(\text{winner} = \text{player 1})$ is simply $\frac{x}{x+y+z}$ [2], determining their losing probability is much more difficult. *E.g. Consider the initial state (1, 2, 3):*

1

$$\begin{aligned} L_{(1,2,3)} &= \frac{1}{6}\big(L_{(0,3,3)} + L_{(0,2,4)} + L_{(2,1,3)} \\ &\quad + L_{(1,0,5)} + L_{(2,2,2)} + L_{(1,4,1)}\big) \\ &= \frac{1}{6}\left(1 + 1 + L_{(2,1,3)} + 0 + \frac{1}{3} + L_{(1,4,1)}\right) \\ &= \frac{7}{18} + \frac{1}{6}\big(L_{(2,1,3)} + L_{(1,4,1)}\big) \end{aligned}$$

One step of the analysis has already introduced 6 intermediate states (but some can be 'pruned' early).

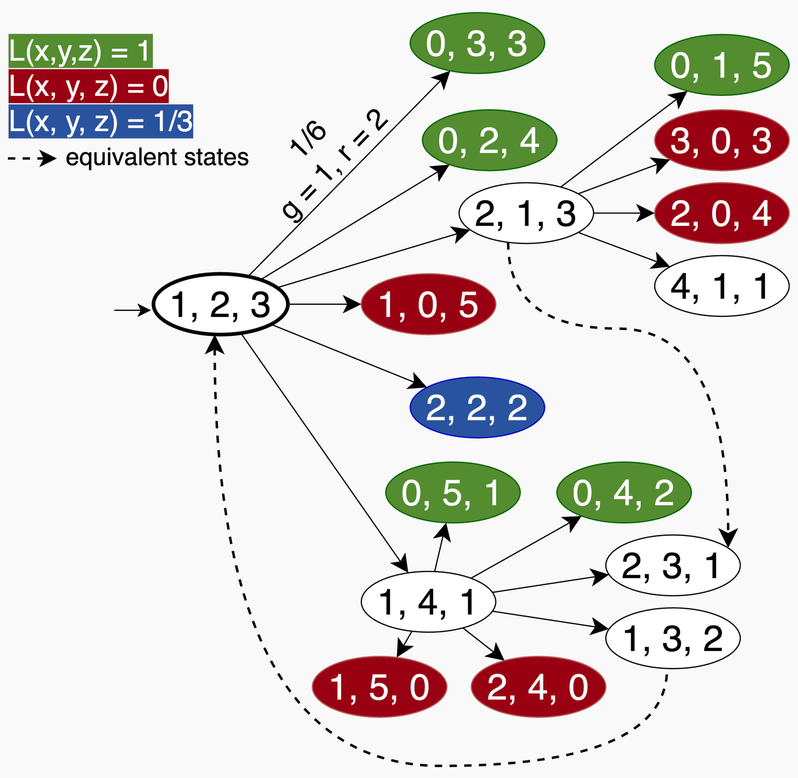


Figure 1.1: Modified transition diagram for Game 1, illustrating how the program analyses state (1, 2, 3).

2

$$\begin{aligned} L_{(1,2,3)} &= \frac{1}{3}\big(L_{(3,1,2)} + L_{(0,4,2)} + L_{(0,1,5)}\big) \\ &= \frac{1}{3}\big(L_{(3,1,2)} + 1 + 1\big) \\ &= \frac{1}{3}L_{(3,1,2)} + \frac{2}{3} \end{aligned}$$

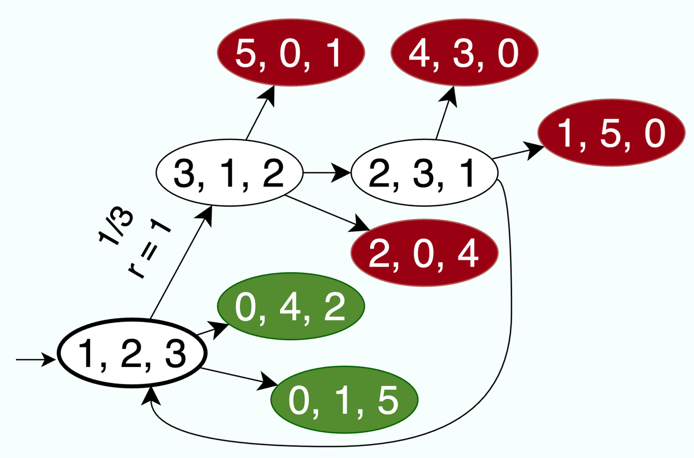


Figure 1.2: Modified transition diagram for Game 2.

Due to the 'depth' of the analysis, as the states become larger, it quickly becomes intractable to calculate by hand. Thus, I wrote a Python program to automate the analysis (with guaranteed termination) and produce the exact $L_{(x,y,z)}$'s in fraction form.

Program Functionalities

Given the initial state as input, the program can:

- Generate and solve the first step analysis equations;
- Generate $L_{(x,y,z)}$ for the initial and intermediate states.
- Export the equations and probabilities to a text file.

These are combined into the `LoserAnalysis` class.

Analysis in the "Selected Results" section is also automated.

Implementation Details

- The program uses the **sympy** library to solve equations.
- $L_{(x,y,z)} = L_{(x,z,y)}$. Our convention is to store the smaller stack at front.
- To accommodate for slow execution large inputs, the program offers:
  i. **Fallback option**: Approximates $L_{(x,y,z)}$, by *enumerating* the game up to a fixed number of rounds (t), using **memoisation** to optimise efficiency. The result is accurate to 2^{-t}.
  ii. **Exception-handling**: If the maximum recursion depth or time limit is hit, the program throws and catches an exception, then skips the current state / uses the fallback method.

Selected Results

The probabilities generated are stored as CSV files in the project's GitHub repository. We only present some remarkable results here.

1. If player 1's objective is not to lose, when are they better off giving away \$1?

1 The smallest* such state is (11, 19, 48) vs. (10, 19, 48):

$L_{(11,19,48)} =$ 9505533830767552681325332460536243296492851849399342033764290735641529081784363354971001546676377024313257419331316917 1878567165664636090433413549346556542318405224485002394984381187095762857031875372514851648744689876606583868833130789124944281795228465 4738673214410277893782426223384701540290921143078273287839698880724012451250170055782308083321973435371340847387859216007087622235865621 6355262548657371387855499893220300767863500018989926419644556351560167468150557905546740168393234581650793331708269728862569517012615864 811807101687363225609355415793126060185989197699764147347376451436008818937161865893721781396257019 41/1781977679341565653879062316696873 8432477569983271536712319868523149518433213441869248413988212864124407801562834940404112578706727560878477149731918101076780906408169254 2704158384981828417284705119427008399746104598790307294201748170143987149291135520733374002287217775101669536582256671886738481160720021 5463380494510770187945490845928143445232732949053965144678241474218200604350499913285920102311013763668735307552502191738215421222671791 4936368792270340460363363600205029257226017910414902297593134822746299465920848726683501707045280321742201920572234233639669931523977995 36216014235688392527706247314395749592323268734337 $\approx 0.5334260883828699$

$L_{(10,19,48)} =$ 384927437348092865826787154522036966929331477671257388795406213322077025027329708904002230160514488758925861008529937 2641865076307695712428629862510743966379144258747669183765648308949640633266498014775160083551908074040886977133195582955400146709753989 3736472089107670746944940138298692043311547964288456867952310747031559711923401123354938125594417587908704419279778529647788014939856406 6044668357513633952588377925634818352400474258882404223657377314751413083100101954283316910145455750718339182097013355811579429030093387 81392941800/7217263194771645633240126555255838476299049886246841503769225103997039449052747771980511648002392515987375937925303682666479 7405000819352675795558218205074946859195078363825316854111978561845163168833699489271135683392883961132105083231492809220415333658006561 83 4637915106023376987127445506182008039880270393859346595996143509931738007423070998446360400532405859776577658805815263294367324760378016 5125261773026254548974238037597139908155084767333005841357760140215127815456879580085371803405945673770446916448716439745218669565734873 6787 $\approx 0.5333426632230114$

2 The smallest* such state is (4, 5, 11) vs. (3, 5, 11):

$$L_{(4,5,11)} = \frac{164}{243} \approx 0.6748971193415638$$

$$L_{(3,5,11)} = \frac{2}{3} \approx 0.6666666666666667$$

* by lexicographical ordering

2. Similarly, when is player 1 better off giving away \$1 to another player?

The smallest such state is (2, 4, 5) vs. (1, 5, 5):

$$L_{(2,4,5)} = \frac{3529997}{7036351} \approx 0.5016800611566989$$

$$L_{(1,5,5)} = \frac{3522116}{7036351} \approx 0.5005600203855664$$

The smallest such state is (2, 9, 10) vs. (1, 10, 10):

$$L_{(2,9,10)} = \frac{327195}{368089} \approx 0.888901868841503$$

$$L_{(1,10,10)} = \frac{8}{9} \approx 0.8888888888888889$$

3. Visualisation of $L_{(x,y,z)}$ for varying x, fixed y and z

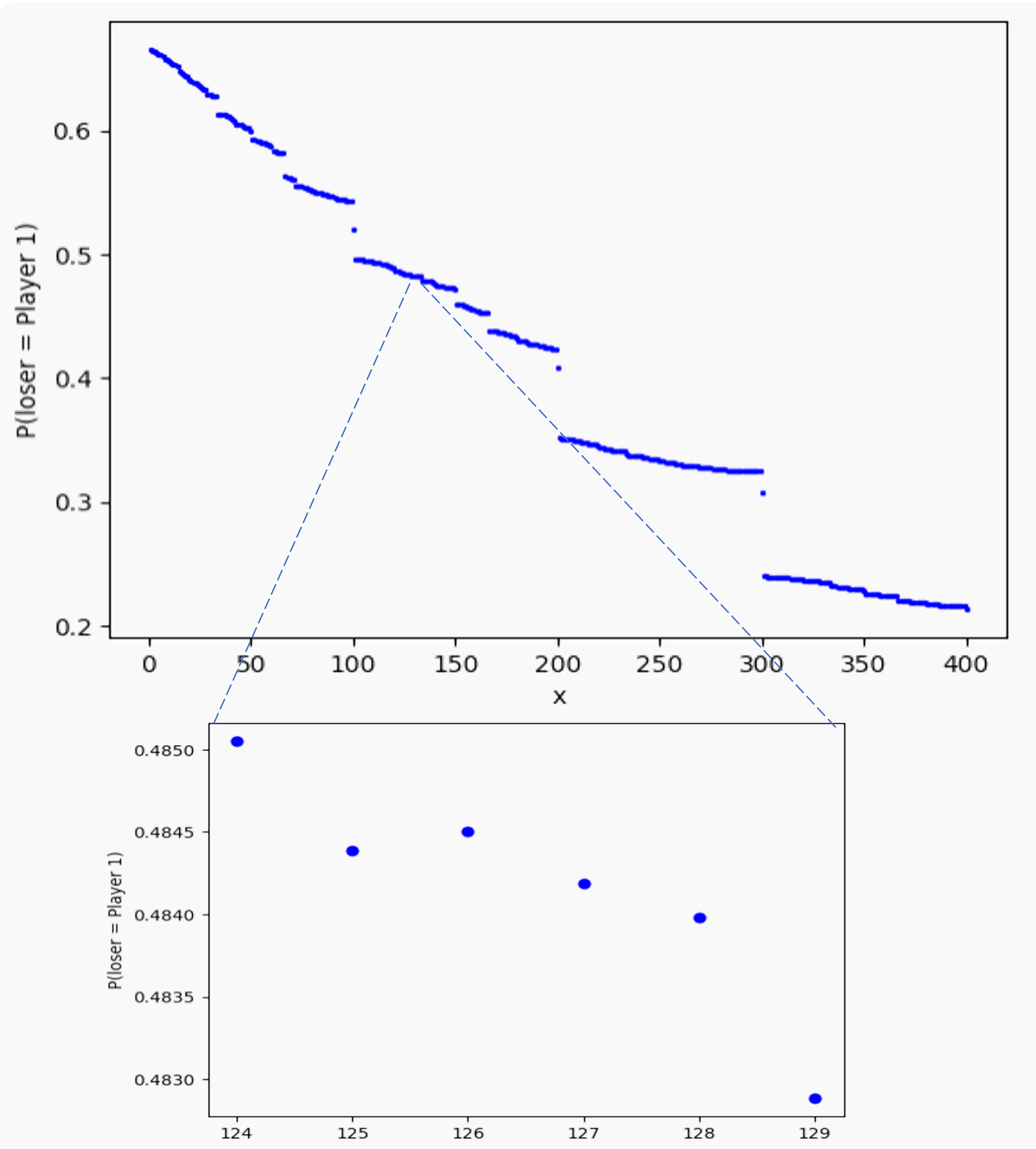


Figure 2.1: $L_{(x,y,z)}$ for $x \in [1, 400]$, $y = 200$, $z = 300$, zoomed into the range $x \in [124, 129]$.

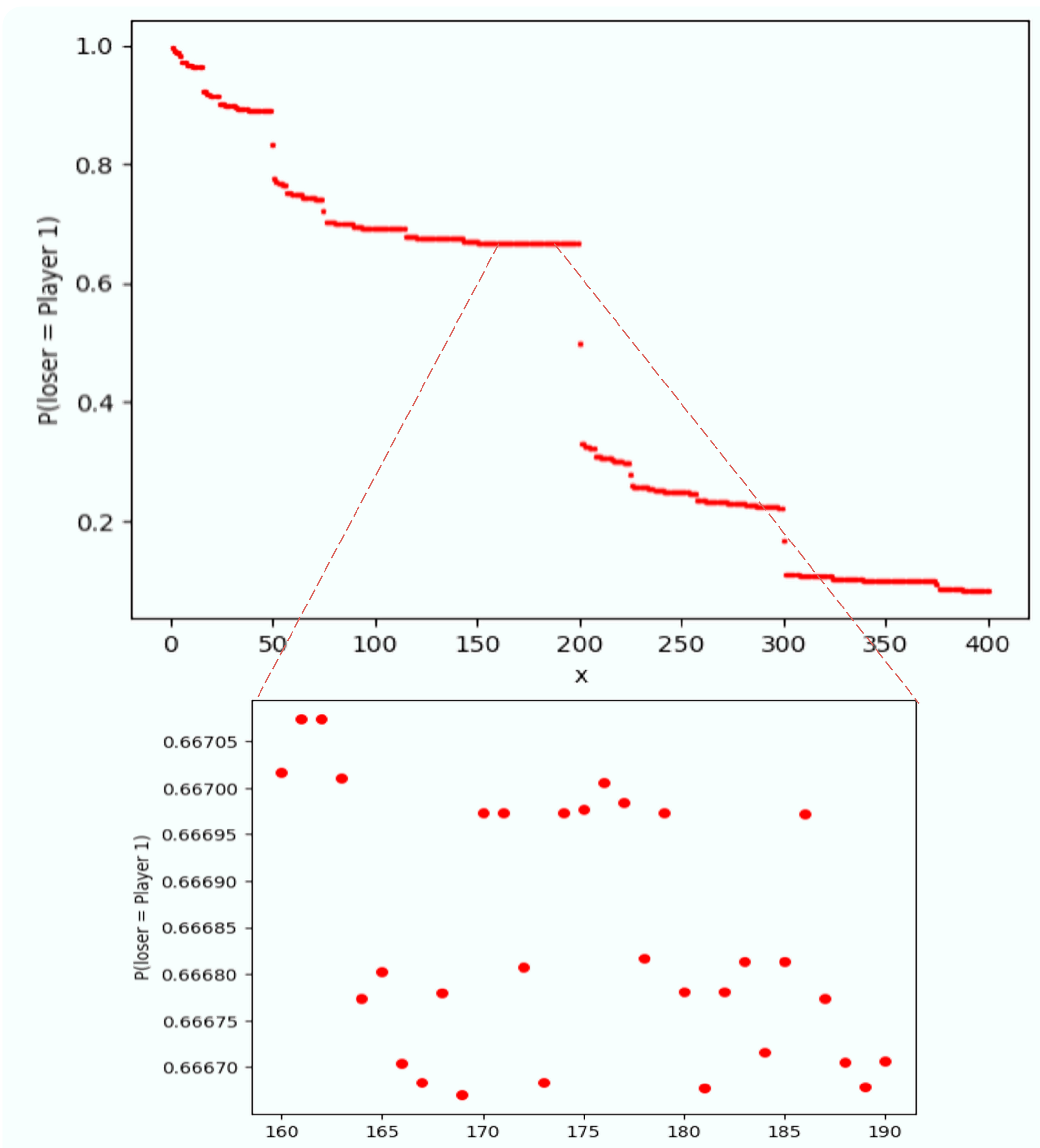


Figure 2.2: $L_{(x,y,z)}$ for $x \in [1, 400]$, $y = 200$, $z = 300$, zoomed into the range $x \in [160, 190]$.

4. Visualisation of $L_{(x,y,z)}$ for fixed sum, $x + y + z$

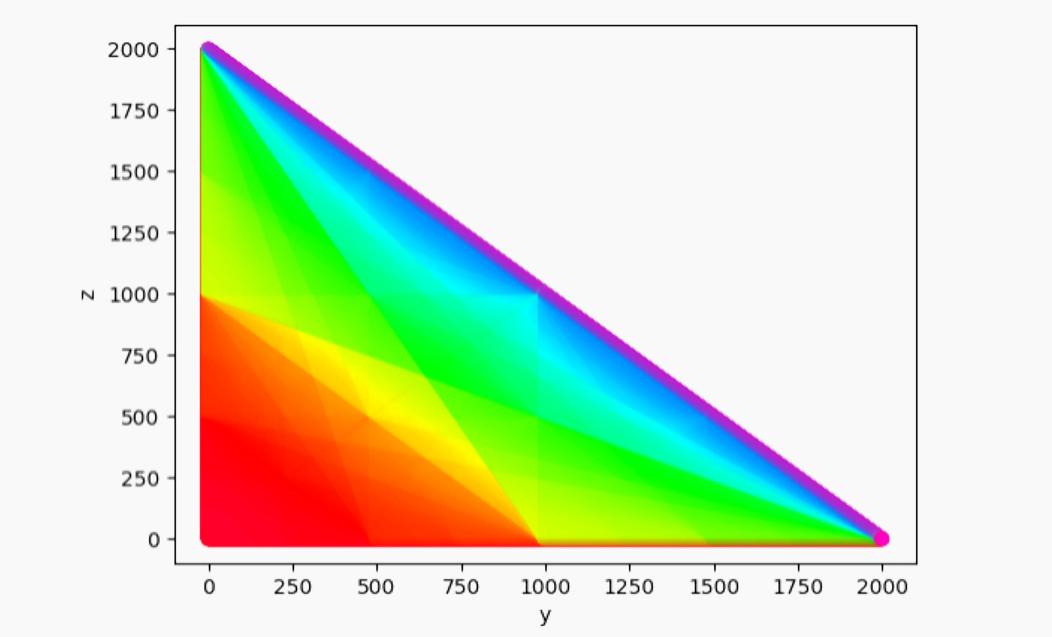


Figure 3.1: $L_{(x,y,z)}$ for fixed $x + y + z = 2000$ in Game 1.

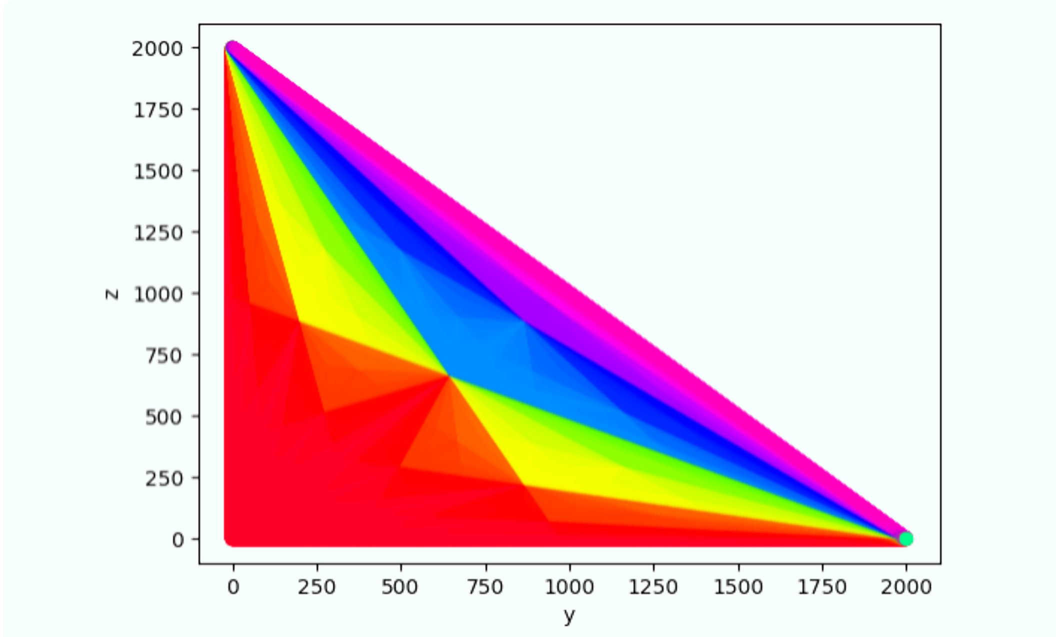


Figure 3.2: $L_{(x,y,z)}$ for fixed $x + y + z = 2000$ in Game 2.

Key Observations

- The big **jumps** happen when player 1's fortune is tied with another player's fortune.
- The function $L_{(x,y,z)}$ is **not monotonic decreasing in x** (visible when we zoom into a flat segment).
- Given the same (x, y, z), the program executes **faster on Game 2** than Game 1 (no. of intermediate states grows exponentially with base 6 in Game 1, but with base 3 in Game 2).
- **Game 2** has **more states where player 1 is not better off with an extra \$1** (more flat segments).
- For both models, $L_{(x,y,z)}$ is **fractal-like**!

Future extensions

- Variants of the model:
  i. Players choose at random to give $\min(x, y)$ or $\min(x, y, z)$.
  ii. More than 3 players.
- Optimise program's efficiency for large inputs.

Acknowledgement

This project has really taught me how to effectively integrate theory with practice, computing techniques with mathematical analysis, to conduct research. I am glad that my computing skills can be incorporated to produce some concrete results. Many thanks to my supervisor, **Prof. Mark Holmes**, for his insightful guidance and support throughout this project.

References

[1] Diaconis, P. & Ethier, S. (2020). Gambler's Ruin and the ICM. *Statistical Science, Statist. Sci. 37*(3), 289-305. https://doi.org/10.1214/21-STS826

[2] Grinstead, C. M. & Snell, J. L. (2006). 12.2: Gambler's Ruin. In Doyle, P. G (Ed.), *Introductory Probability* (pp. 487-490). American Mathematical Society.