



hepcatjk committed on GitHub Update Readme.md

Latest commit 8510560 8 days ago

..

SnakesAndLadders	Added BaseCode for Part 2 - Snakes and Ladders. Changed Readme to ref...	26 days ago
images	queue image	22 days ago
queues	todo wraps	22 days ago
Readme.md	Update Readme.md	8 days ago

## Readme.md

# Homework 2 : Discrete-Event Simulation Assignment

## IDS6938-Simulation Techniques - [University of Central Florida](#)

[University of Central Florida](#) This is the framework for homework #2.

The assignment is due: **Tuesday, March 28 at 11:59PM (EST)**

## Introduction

A Discrete-event Model simulates a complex system as an ordered sequence of well-defined events. Mathematically Discrete-event models use Markov Processes, Queuing systems, events, probability / statistics, and random variables. The purpose of this assignment is to learn the mathematical foundations, how to program these models, and how to simulate them. The assignment is due Tuesday, March 28, 2017 at 11:59 P.M.

Major parts for the Assignment You can think of the assignment broken up to 4 major parts:

- Empirical Tests of Randomness
- Snakes and Ladders (Discrete Event Markov Chains and Monte Carlo Simulations)
- Discrete Event Simulation - Queue Simulation
- Composing a final report

The goal of this assignment is to become familiar with the concepts in the second third of the class. You will be expected to compose a *final report* which demonstrates your understanding on the material in each section of the assignment. Be visual! - Pictures say a thousand words so you do not have to. Show off your different configurations and really explore the assignment.

## Assignment

### Part 0 - Getting Started

Read the assignment. Sync your fork with the [main IDS6938 repository](#). Use CMake to create project files for the Homework 2 assignment (*Hint: and Discrete Lecture folders*). Set your *startup project* to the correct project. Test building and executing the homework 2 project. Look over and understand the framework and find the functions you need to edit for the assignment.

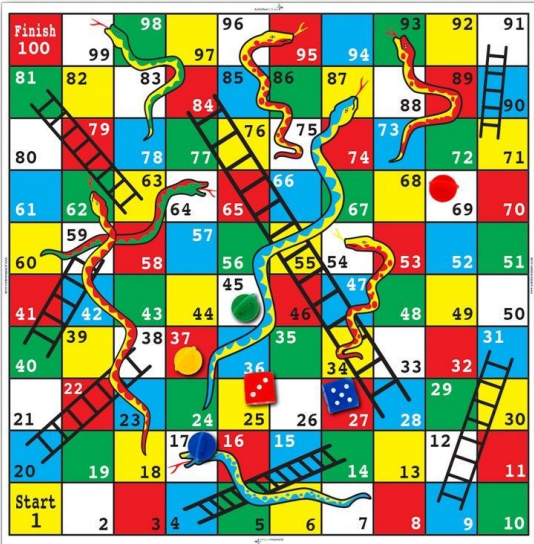
## Part 1: Empirical Tests of Randomness (20 pts).

We looked at different ways to generate [pseudo-random numbers](#) and [quasi random numbers](#). Generating random numbers are crucial to Discrete-Event simulations which rely on random variables and stochastic processes. This problem explores different random number generators, distributions, and statistics. Different [C++ pseudo-random numbers engines are instantiated](#) already for you. Also a wide variety of standard distributions are implemented. Two quasi random number generators are also provided. - One must be the Sobel - quasi random number generator.

- (a) - **3pts**: Output the results of five different random number engines, using a uniform distribution for values between [0-100]. Generate useful charts and statistics from the output to analyze how uniform these values truly are. You are expected to look at some advanced statistics and test, for example: tests like the Kolmogorov-Smirnov test, Chi-square test, Autocorrelation test, and Spearman's Rank Correlation Coefficient are a few examples of ones you could use.)
- (b) - **2pts**: Vary  $N$  (amount of samples). How do things change.
- (c) - **3pts**: Fix a random engine of your choice from part (a), and now vary five different [distributions](#) for just the pseudo-random numbers. Again, analyze your results with graphs and statistics of choice.
- (d) - **4pts**: Generate random numbers in two-dimensions for a unit square. Plot the results for the different random number engines. The vertical axis should vary  $N$  in increasing order. The horizontal axis should show of the random number engines.
- (e) - **4pts**: Generate random numbers in two-dimensions for a unit square. Plot the results for the different distributions. The vertical axis should vary  $N$  in increasing order. The horizontal axis should show of the random number engines. (See [Random Numbers Webcourse page](#) for a rough idea what you should produce.)
- (f) - **4pts**: Repeat parts (d) and (e) with a unit circle.

## Part 2 - Snakes and Ladders (Discrete Event Markov Chains and Monte Carlo Simulations) (30 pts)

We all love board games. A board game can be viewed mathematically as a Markov chain, where the probability of moving to the next position depends only on the position you are currently at and the chances provided by tossing a dice. For this part of the homework we will simulate the game "Snakes and Ladders" (This goes by other names: Chutes and Ladders, Moksha Patam but all essentially the same gameplay.)

Moksha Patam	Snakes and Ladders
	

### Background

The classic game has 100 positions on the board. You toss one die, and move squares based on the result of the die. If you land on a ladder you move up the ladder to a higher numbered square. If you land on a snake's mouth, you descend to a

lower numbered square. For purposes of simulation, we will add one extra square 0 (starting position). So there are 101 positions on the board.

The game is **memoryless** - your progression to the next position is independent of how you arrived there (opposed to Blackjack or Candyland where your progression is based on what cards have been drawn). A Markov Chain defines the probability of a move from state  $i$  to state  $j$  by a **Transition Matrix**,  $T$ . So in the case of *Snakes and Ladders* the dimensions of a transition matrix is 101x101.

- **(a) Null State Game transition matrix - 10pts:** The *null state game* is defined by a game with no snakes and no ladders. This simplifies the game to just the moves of the two dice rolls. Create the transition matrix for the null state game. The Transition Matrix would be decided by the roll of a fair, six-sided die, so it would start to look like:

$$\begin{pmatrix} 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 & \dots & 0 \\ \dots & & & & & & & & & & & \end{pmatrix}$$

From state 0 it is equally probable of landing on squares 1-6. From state 1 it is equally probable of landing on squares 2-7, and so on. Create this transition matrix. The end is trickier, we will consider any roll past 100 a win case. (Opposed to rolling exactly onto square 100.) Confirm you have a well formed stochastic matrix (Write checks for confirming each row of  $T$  sums to one and all elements are non-negative). The Transition Matrix methods can be found in the TransitionMatrix.h file.

- **(b) Simulate and analyze the results of Null State Game - 10pts:** What is the modal number of moves required by a single player to finish the game? We will be simulating the game two different ways. **(1) Markov Chain:** The game can be analyzed with a row vector,  $v$  with 101 components, representing the probabilities that the player is on each of the positions.  $V(0)$  is  $(1,0,0,\dots,0)$  since we know we start at square 0.  $v$  evolves by:

$$v^{(k+1)} = v^{(k)}T.$$

For this part (1) use the *Markov project* in the Snake and Ladders starter code.

**(2) Monte Carlo:** he will use a monte carlo process to solve our Discrete Time Markov Chains. Here (2) use the DTMC project, and utilize the DTMC method similar to what we did in class.

Produce graphs to analyze the results and show how the game evolves over time for both methods. Plot useful statistics of the results such as percentage chance of finishing the game in  $n$ -moves, cumulative probability of finishing the game in  $n$ -moves, and other ways to convey useful information of the results.

- **(c) Simulate and analyze the results of Snakes and Ladders -10pts:** Construct a new transition matrix based on the table:

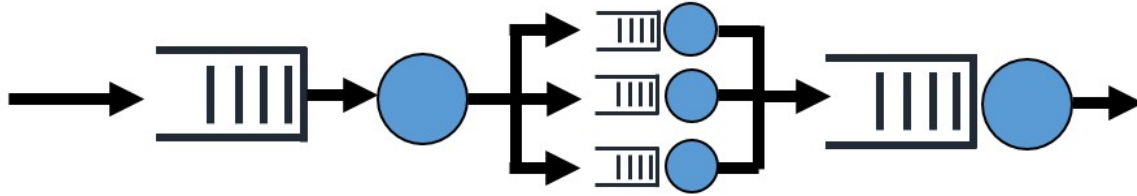
Ladders From	Ladders To		Snakes From	Snakes To
3	19		11	7
15	37		18	13
22	42		28	12
25	64		36	34
41	73		77	16
53	74		47	26
63	86		83	39
76	91		92	75
84	98		99	70

Run the same simulation and analyze your results similar to part (b) for the proper game of *Snakes and Ladders* for both methods. How often are the snakes and ladders used, how do the probability of finishing change, etc? What is the maximum and expected amount of moves for the game? Use charts and graphs to illustrate these points.

- **(d) Think - 0pts:** If these games are built entirely on chance, do they require any strategy? Is it really a *game*, would you rather play games of chance or games of strategy?

## Part 3 - Discrete Event Simulation - Queue Simulation (30 pts)

This problem will look at queues and commonly used performance measures. For this problem we will look to design a simple airport security check. We will make the following assumptions: (1) there is only one airline - Southwest; (2) passengers' interarrival times are independent and identically distributed (IID) with an exponential distribution with mean  $1 / \lambda$ . The service times are also assumed to be IID and exponentially distributed random variables with mean  $1 / \mu$ .



When a passenger arrives they have to wait in a queue to present their ID and ticket to the gate agent with all the other passengers. Once approved by the agent they will have to pass through a security check. Since this is Orlando, there are only 3 open metal/screening devices open and again passengers have to wait in a queue. After passing through security you again have to wait in a queue to board your plane.

- (a) - 4pts: To start create the scenario in the figure above in `main.cpp`. Checkin will have a  $\mu$  of 53 and accept new arrivals, the security gates will have a  $\mu$  of 20, and will not accept new arrivals, boarding will have a  $\mu$  of 80. You will have to set up the appropriate `MM1_Queue` objects to capture the functionality above.
- (b) - 4pts: You want to add a check that your process is within an error range `is_within_error_range(float)` where the error range will be 0.002. You also want to process the next event, and add an external arrival where marked.
- (c) - 3pts: in `mm1_queue.cpp` : add code to calculate the expected results for:
  - expected\_server\_utilization
  - expected idle prob
  - expected queue length
  - expected number customers
  - expected waiting time
  - expected response time
- (d) - 4pts: Write code to call the functions to output and generate data from the airport scenario. Plot and analyze the useful statistics/results in the program of your choice. (Hint - basically call `.output()`; on the `MM1_Queue` objects you create. Hint2 - two other use functions are `get_current_time()` and `plot_results_output()` call initially on your initial `MM1_Queue` object.)
- (e) - 15pts: Download the personal edition of [Anylogic](#), read through the [documentation](#) as needed, and set up the same type of simulation discussed above.

## Part 4 - Implementing Extra Features (10 pts)

Implementing 2 features on the extra features list. Pick any feature on the "extra features" list below to customize your assignment to fit your interests. Please document this in your writeup. (Note: These should total 10pts. You could successfully implement a feature worth 10pts or greater. This also fulfills this requirement. The features are assigned points based on difficulty. The 5pt features are more straightforward.)

## Part 5 - Final Report (10 pts)

Write up the results to the previous sections in the main `readme.md` in your forked repository. Turn in the URL for your fork in webcourses. Be visual. The report should contain the graphs and analysis requested. I have high expectations for the documentation here and you should allot the proper time to compose the writeup.

## Extra Features (Extra Credit - 25pts)

You have to implement two features from this list for Part 4. You may choose any two features you wish from this list. (Please explicitly note them in your `Readme.md`)

If you feel like going beyond the scope of the assignment, you should consider implementing more of the following extra features. *Get the assignment working without them first.* You can get a maximum of 25 points in extra credit. Simply implementing these things doesn't guarantee you a 25; you really need to go above and beyond to get the full amount. *(The instructor reserves the right to hand out extra credit as his he sees fit.)*

- **(5 Points)** - Implement and compare Halton, Hammersley, (Or another quasi method) Quasi sequences and add them to your analysis for appropriate subparts of Part 1.
- **(5 Points)** - Implement different distributions inside the Quasi random sequences and add them to your analysis for appropriate subparts of Part 1.
- **(5 Points)** - Implement and compare another (advanced) pseudo random sequences and add them to your analysis for all subparts of Part 1 (Examples include: [PCG](#), or [Random123](#)).
- **(20 Points)** - Complete Part 2 with a different board game. Construct the game's transition matrix, simulate the game, and analyze the results (Run the game past the instructor).
- **(20 Points)** - Complete Part 2 with a 3D version of Snakes and Ladders. Construct the game's transition matrix, simulate the game, and analyze the results.
- **(10 Points)** - Provide code in (Python, R, Matlab...) that demonstrates an animation of the board itself evolving overtime for Snakes in Ladder for part 2.
- **(10 Points)** - Add a 2D visualization to AnyLogic for Part 3.
- **(10 Points)** - Add a 3D visualization to AnyLogic for Part 3.
- **(5 Points)** - Pick a research problem (from your dissertation, Energy Microgrids, Missile Defence...). Compare and contrast different software packages (AnyLogic, Simio, Simulu8, Arena etc). Convince me as your "manager/advisor" what advantages, weaknesses, and costs each program has. Come to a conclusion. Roughly sketch out how you would formulate the problem in the framework.
- **(10 Points)** - Create an Anylogic simulation for Part 2 - Snakes and Ladders.
- **(10 Points)** - Setup another resource allocation / queueing problem related to your research in Anylogic with a basic visualization
- **(10 Points)** - Setup up SmartGrid or a Microgrid System that creates resources of power from wind, solar, and electric grid. Set up various devices in your house to draw power. [Reference 1](#), [Reference 2](#).
- **(25 Points)** - Set up a discrete simulation of your choice that uses an approved real-time data source and visualizes the results. (See instructor for approval).
- **(N Points)** - You are welcome to make suggestions for a feature of your own choosing, but they must be approved by instructor before implementing.

