EGR-101 Intro Computing Engineers

Due: 8 November 2021 at the start of class

Question 1 (10 Points)

Write a well-documented MATLAB script $hmwk8Q1.m$ to perform a Monte-Carlo simulation of 1,000,000 trials of an experiment where a duel is simulated between two people. Create a MATLAB function $randomDuel()$ to simulate a duel where each contestant comes at a random moment between NOON and 1 P.M. on an appointed day and leaves exactly five minutes later. A duel occurs only if opponents are together within the time interval. The function returns a one (1) if a death occurs or else returns zero (0). The program $hmwk9Q1.m$ calls the function $randomDuel()$ and tallies how many deaths occur and estimates the probability of death by dividing the number of deaths divided by the number of simulations.

Record the estimate of the probability of death as a comment in $hmwk8Q1.m$.

Grading: 5 Points for $hmwk8Q1.m$, 5 Points for $randomDuel.m$, the file containing the function.

Question 2 (10 Points)

Write a well-documented MATLAB script *hmwk8Q2.m* that performs a Monte-Carlo simulation of 1,000,000 trials of two gamblers playing a game of chance. Gambler A has $2, Gambler B has $1. Each time the gamblers play, a $1 is exchanged to the winner. We seek to determine the likelihood Gambler A does not go bankrupt over 1,000,000 trials. Create a MATLAB function $gamblersRuin(odds)$ to simulate a single trial of two gamblers playing. The MATLAB function returns 1 if Gambler A does not go bankrupt, and 0 if Gambler A does go bankrupt, again in a single trial. The input parameter $odds$ affect the simulation. When $odds = 0$, Gambler A never wins. Gambler A always wins when $odds = 1$. Similarly, when $odds = 0.5$ Gambler A when fifty percent of the time, and so forth.

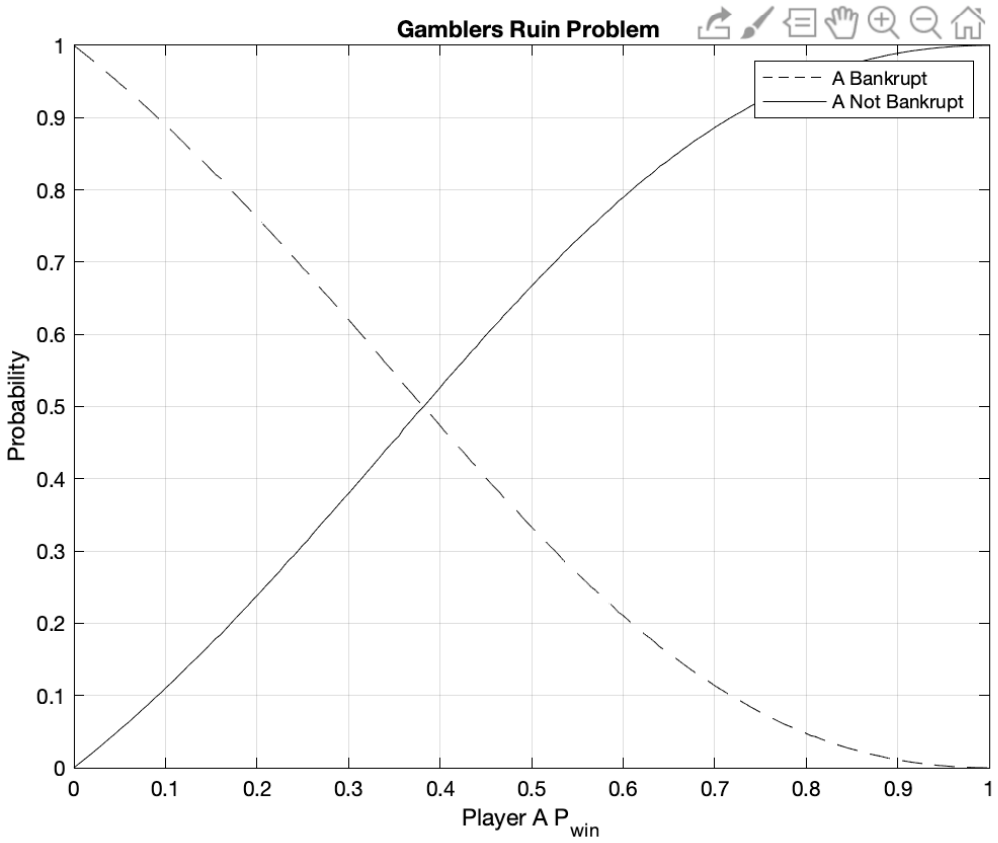
One way for the MATLAB function $gamblersRuin(odds)$ to work is to call the random number generator as follows:

```
play = rand();
if play < odds
    playerOneMoney = playerOneMoney + 1;
    playerTwoMoney = playerTwoMoney - 1;
else
    playerOneMoney = playerOneMoney - 1;
    playerTwoMoney = playerTwoMoney + 1;
end
```

where the variables $playerOneMoney$ and $playerTwoMoney$ keep track of each Gambler's bankroll.

The MATLAB script $hmwk8Q2.m$ contains two (2) for-loops. The outer for-loop might vary the odds from zero to one. The inner for-loop might repeat the number of trials 1,000,000 and compute the average number of trials where Gambler A does NOT go bankrupt. Your program should produce a figure similar to the one shown below.

s

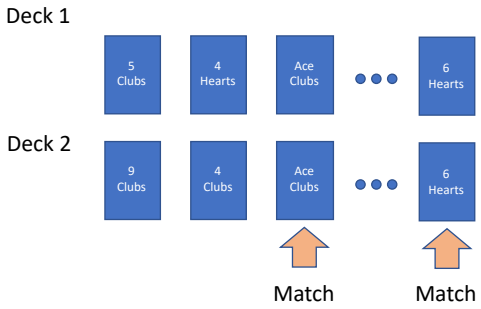


hmwk8Q2.fig

Grading: 5 Points for $hmwk8Q2.m$ and the figure $hmwk8Q2.fig$, 5 Points for $gamblersRuin(odds).m$, the file containing the function.

Question 3 (10 Points)

Write a well-documented MATLAB script hmwk8Q3.m to perform a Monte-Carlo simulation of 100,000 trials of an experiment where two (2) decks of playing cards are laid out on a table. A MATLAB Function myDeckExperiment() returns the number of matches when the randomly shuffled decks are laid out from left to right as shown below. For the case shown below, the number of matches returned to the program in hmwk8Q3.m is TWO (2) as the playing cards in positions three (3) and fifty-two (52) are identical in both number and suit. Use the MATLAB function *randperm()* in your function. Also, use logical indexing to help identify the matching conditions.



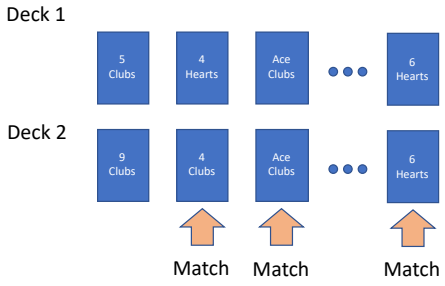
2 MATCHES FOR THE TRIAL

The program *hmwk8Q3.m* tallies the average number of matches, calling *myDeckExperiment()*. Record the average number of matches as a comment in *hmwk8Q3.m*. You can use a for-loop in running the experiment through repetition.

Grading: 5 Points for *hmwk8Q3.m*, 5 Points for *myDeckExperiment.m*, the file containing the function.

Question 4 (10 Points)

Write a well-documented MATLAB script hmwk8Q4.m to perform a Monte-Carlo simulation of 100,000 trials of an experiment where two (2) decks of playing cards are laid out on a table. A MATLAB Function myDeckExperimentTwo() returns the number of matches when the randomly shuffled decks are laid out from left to right as shown below. For the case shown below, the number of matches returned to the program in hmwk8Q4.m is THREE (3) as the playing cards in positions two (2), three (3), and fifty-two (52) are in number alone. Use the MATLAB function *randperm()* in your function. Also, use logical indexing to help to identify the matching conditions.



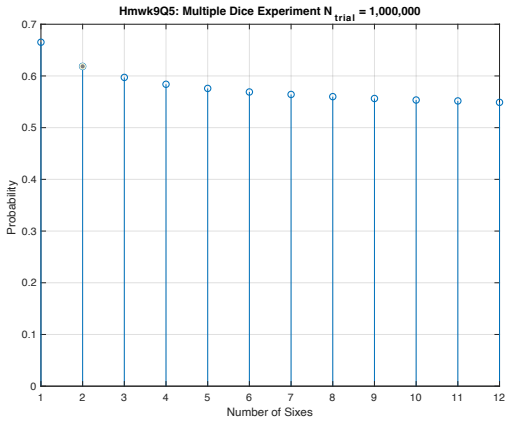
3 MATCHES FOR THE TRIAL

The program *hmwk8Q4.m* tallies the average number of matches, calling *myDeckExperimentTwo()*. Record the average number of matches as a comment in *hmwk8Q4.m*. You can use a for-loop in running the experiment through repetition.

Grading: 5 Points for *hmwk8Q4.m*, 5 Points for *myDeckExperimentTwo.m*, the file containing the function. REUSE *hmwk8Q3.m and myDeckExperiment.m in your solution.*

Question 5 (10 Points)

Write a well-documented MATLAB script hmwk8Q5.m to perform a Monte-Carlo simulation of 1,000,000 trials to determine which of the twelve (12) events is more likely: that a person gets (a) at least 1 six when 6 dice are rolled, (b) at least 2 sixes when 12 dices are rolled, (c) at least 3 sixes when 18 dices are rolled, … (l) at least 12 sixes when 72 dice are rolled. Consider a solution where two (2) for-loops are employed. The outer for-loop performs experiment (a), (b) … (l), while the inner for-loop performs 1,000,000 trials. Use the MATLAB function stem to produce the figure below.



Hmwk8Q5b.fig

Grading: 5 Points for *hmwk8Q5.m* and 5 Points for *hmwk8Q5b.fig*.