# PROGRAMMING ASSIGNMENT 6

**Due:** Sunday, November 22 by 11:59pm. Upload *one* Jupyter notebook to Gradescope.

## 1. A 2D RANDOM WALK

Simulate a 2D random walk that is *not* on a grid. Assume a step in an arbitrary direction, with all angles being equally probable. Assume that the length of each step is also a random variable distributed uniformly between 0 and some user-defined variable $a$, so that the average step length is $a/2$.

Make plots of two sample trajectories, both with an average step length of 1, and also plots of mean-squared displacement vs. time for a few different average step lengths. Compute the mean-squared distance by averaging over $10^4$ trajectories of up to 200 steps each. Is the mean-squared displacement vs. time graph linear? How does the slope of the mean-squared displacement vs. time graph depend on $a$? Is the slope proportional to $a$, $a^2$, $\sqrt{a}$, or something else?

*Hint*: *test* that your code is producing reasonable results with significantly fewer trajectories *first*. Once you crank the number of walkers up to $10^4$, your code will take a while to run.

*Grading Rubric*: Program has required features described in prompt (25%), graphs show correct behavior and are easy to interpret (25%), written response is factually correct (25%) and uses proper grammar and spelling (15%), code compiles without errors and reproduces all output (10%)

## 2. RANDOM WALK WITH A PREFERRED DIRECTION

Do a 2D walk on a grid for the following cases:

- $P(\text{left}) = P(\text{right}) = 0.3$ and $P(\text{up}) = P(\text{down}) = 0.2$
- $P(\text{right}) = P(\text{up}) = 0.3$ and $P(\text{left}) = P(\text{down}) = 0.2$

Show $\langle x \rangle$, $\langle y \rangle$, $\langle x^2 \rangle$, $\langle y^2 \rangle$, $\langle x^2 \rangle - \langle x \rangle^2$, and $\langle y^2 \rangle - \langle y \rangle^2$ as functions of time.

Explain in a few grammatically correct sentences why some of these graphs fluctuate around zero, why some are linear, and why some of the linear graphs have larger slopes than others.

*Grading Rubric*: Program has required features described in prompt (25%), graphs show correct behavior and are easy to interpret (25%), written response is factually correct (25%) and uses proper grammar and spelling (15%), code compiles without errors and reproduces all output (10%)

## 3. THE HOUSE ALWAYS WINS

Let's consider a somewhat simplified casino slot machine, which works by randomly circling through five wheels containing eight different symbols. The player places a bet of fixed amount (let's say \$1) and then spins the wheels. If two of the same symbols appear consecutively at any place along the screen, the machine pays back the initial bet. If three or more of the same symbol appear consecutively, the machine pays back a multiplicative win: three in a row returns \$2, four in a row returns \$5, and five in a row returns \$20. If no numbers appear consecutively, the player loses their \$1 wager (the house wins).

A player starts with \$1,000. Run a Monte Carlo simulation of the player placing 10, 100, and 1000 bets on this machine, and track both the number of games won and the total amount of money the player has at the end of the night. Run many realizations of each case, and calculate (1) the percentage of games where the player wins (define "winning" as "no net loss of money"), (2) the typical (median) amount of money the player ends the evening with, and (3) the maximum and minimum amount of money the player ends with. Create several histograms of your results to support your conclusions, and answer the following questions:

- In which scenario is the player *more likely* to win money: after placing a few bets, or after placing many bets?
- What is the average amount lost *per bet* (as a fraction of the initial bet amount)? The \$1 slot machines at Las Vegas hotels have a typical "house edge" of about 8.1% - meaning this is, on average how much the casino gains (and the player loses) per bet. How does your simple simulation compare with a Las Vegas slot machine?
- Roughly how many bets could the player place before running out of money entirely?

*Grading Rubric*: Program has required features described in prompt (25%), graphs show correct behavior and are easy to interpret (25%), written response is factually correct (25%) and uses proper grammar and spelling (15%), code compiles without errors and reproduces all output (10%)