Assignment 0: MATLAB tutorial

Integrated Workshop

Due Fri., Sept. 15, 2023 at 11:59 PM

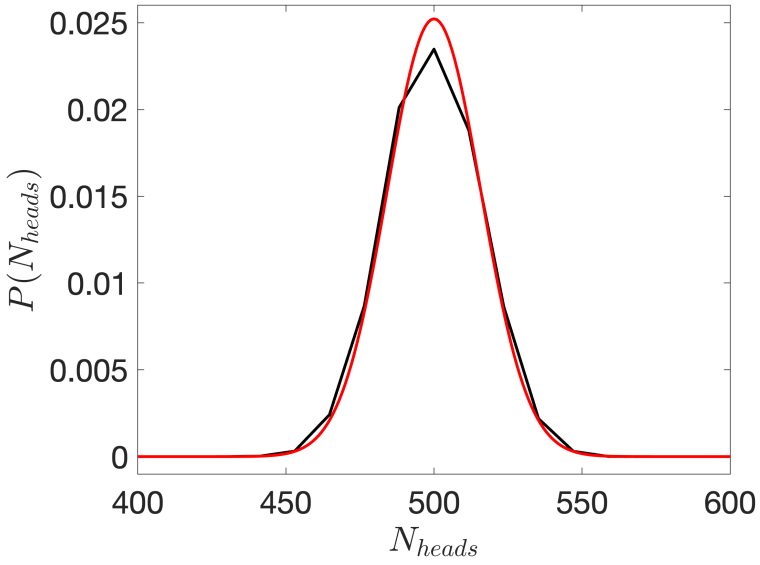
For this assignment, the code for each problem should be written in its own MATLAB script, and each sub problem should be separated into different sections (using the sectioning comments `%%`). Please name each script `LASTNAME_assignment0_problemX.m`.

Problem 1. Coin flipping experiment

To begin this problem, start from the template code `assignment0_problem1_template.m` found on Canvas in the Assignments folder.

In this problem, we will analyze the statistics of coin flips using MATLAB. Imagine we have a perfect coin, i.e. one that has equal probability of landing on either heads or tails when you flip it. We want to know what is the likelihood of observing a given number of heads or tails after a set of repeated flips. We will simulate flipping a coin `nFlips` times, over `nTrials` independent trials.

1. First, declare the variables `nFlips` and `nTrials` to store the number of coin flips and flipping trials.
2. Create two arrays `nHeads` and `nTails` that store the number of heads and tails found in each trial.
   (a) The `rand` function in MATLAB can create an array of random numbers distributed uniformly between 0 and 1. You can first create a large `nTrials` $\times$ `nFlips` matrix with all random numbers, and then the `round` function to obtain a matrix of all 0s and 1s. Each row will then correspond to an independent trial, and we can treat the 0s as tail flips and 1s as head flips. Then you can use the `sum` function to calculate the number of heads obtained. Using some simple arithmetic, you can do something similar to determine the number of tails obtained in each trial.
3. **[Output]** Use the MATLAB `histogram` function to calculate the normalized probability density function (pdf) of the number of heads and tails found per trial, and plot the distributions on the same figure. Be sure to label the axes, make the axes font 22pt, and add a legend so we can tell which curve is for the heads distribution, and which one is the tails distribution. Number this figure as Figure 1 [i.e. using `figure(1)`].



4. **[Output]** Repeat parts 2 and 3, now with `nFlips` fixed to 10^3, and carry out `nTrials` $= 10^2$, 10^3, and 10^4. Plot the distribution **only of the number of heads flipped** for each different experiment on the same figure window (Figure 2) with a legend, labeled axes, 22pt font, and choose 3 different plot styles for the curves so they are easily legible. In addition, you should also plot the binomial distribution with a different plot style (color/line type) and confirm that for a large number of trials, the binomial distribution will emerge (See Fig. 1).

Problem 2. Mean square-displacement of a 2D random walker

To begin this problem, start from the template code `assignment0_problem2_template.m` found on Canvas in the Assignments folder.

In this problem, we have provided a data file `randomWalker.dat`, found in the Assignments folder on Canvas, that contains the x and y positions of a random walker in two dimensions (see the trajectory in Fig. 2 of this assignment). The particle positions are in units of microns (μm) and times are in units of seconds in this assignment.

You will analyze the motion of this random walker using the mean square-displacement (MSD). The MSD describes the average squared-displacement a particle will travel during a given time window Δt. If the random walker particle has position $x(t)$ and $y(t)$ at time t, then the MSD is defined as

$$\mathrm{MSD}(\Delta t) = \left\langle [x(t_0 + \Delta t) - x(t_0)]^2 + [y(t_0 + \Delta t) - y(t_0)]^2 \right\rangle_{t_0}, \tag{1}$$

where the angle brackets denote an average over *time origins* t_0. That is, the MSD over a time window Δt is measured with respect to all differences in time (called a *time window*) of length Δt. For example, if you had a random walker trajectory with 5 time points (say $t = 1$ second, 2 seconds , ..., 5 seconds), we would have 4 possible sizes for the time window Δt: $\Delta t = 1$ sec. between adjacent time points, up to $\Delta t = 4$ seconds between the time point

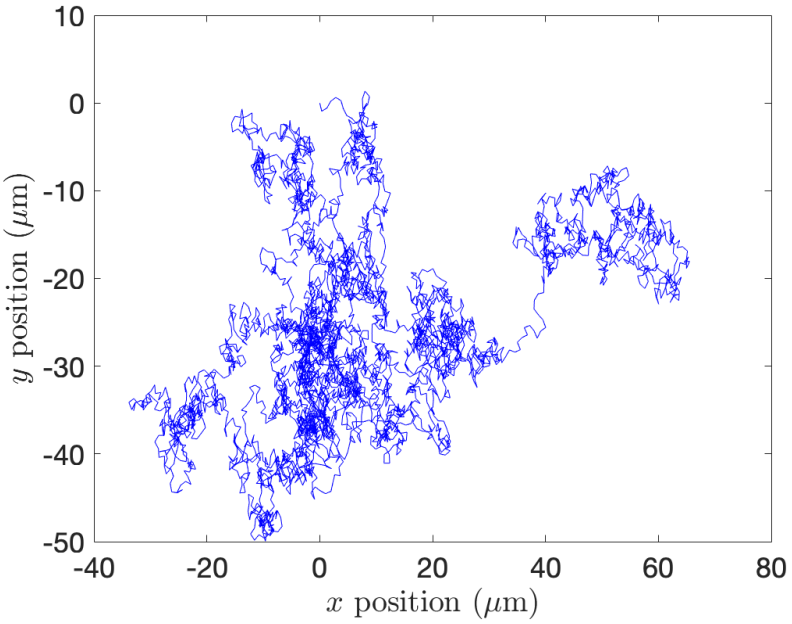


Fig. 2: 5000 steps of a random-walking particle, which takes a step of unit length (i.e. 1 μm) in a random direction, independent of all previous steps.

at $t = 5$ sec. and $t = 1$ sec. The MSD is averaged over all possible time windows; so the value of the MSD at $\Delta t = 1$ sec. would be

$$\begin{aligned}\text{MSD}(\Delta t = 1 \text{ sec.}) = \frac{1}{4}\big\{&[x(2 \text{ sec.}) - x(1 \text{ sec.})]^2 + [y(2 \text{ sec.}) - y(1 \text{ sec.})]^2 \\ &+[x(3 \text{ sec.}) - x(2 \text{ sec.})]^2 + [y(3 \text{ sec.}) - y(2 \text{ sec.})]^2 \\ &+[x(4 \text{ sec.}) - x(3 \text{ sec.})]^2 + [y(4 \text{ sec.}) - y(3 \text{ sec.})]^2 \\ &+[x(5 \text{ sec.}) - x(4 \text{ sec.})]^2 + [y(5 \text{ sec.}) - y(4 \text{ sec.})]^2\big\}\,.\end{aligned} \tag{2}$$

We calculate the squared displacement between every displacement separated by 1 second, and then we average over the total number of displacements of that size. **Note:** for evenly-spaced time points in a trajectory, if there are N_t time points, then there are $N_t - 1$ adjacent time points, $N_t - 2$ instances separated by 2 time points, etc. This implies that, for a trajectory with N_t time values, there will be $N_t - 1$ values for the MSD.

1. Read in the data from the file `randomWalker.dat` using `textscan`. The file has three columns, which correspond to the the times t and the positions x and y, respectively. Read this data into the vectors `t`, `x` and `y`.

2. Calculate the MSD for the random walking particle.

   (a) Create an array of zeroes called `MSD` to store the calculated MSD values. Note that, if there are N_t values of time in the trajectory, the MSD will have $N_t - 1$ values.

(b) Using a `for` loop, loop over the different displacements in time, and take their averages and store the results in the `MSD` array.

i. To do a `for` loop, use the syntax

```
for ii = 1:(NT - 1)
    ...
end
```

where `NT` is the number of time points in the given trajectory. Matlab will iterate over the code indicated by the `...`, where `ii` takes on the values in the vector `1:(NT-1)`.

ii. In the body of the for loop, you will need to take averages over many displacements. One quick way to this is to use vectorization: consider the following line of code:

```
dx = x(2:end) - x(1:end-1)
```

This line creates a new vector `dx` that takes the 2nd to final entry in the vector `x` and subtracts off the first to second-to-last entry in the same vector. Thus, if `NT` is the number of time points and therefore number of entries in the vector `x`, `dx` is a vector with `NT - 1` entries. It is exactly what we want for the first entry in the MSD, i.e. all of the displacements for the particle that are separated in time by 1 time point. Using a similar expression for `dy`, you can add every displacement together and use the `mean` function to get the average x and y displacement for a given time window. Looping over time windows can then give all entries in the `MSD` array.

(c) Create an array of zeroes called `deltaT` to hold the possible values of the time windows Δt. This array will also have $N_t - 1$ values. You can do this part either using a for loop or using vectorization.

3. **[Output]** Plot the MSD vs the lag time Δt on a $\boldsymbol{log_{10} - log_{10}}$ scale and label the axes with correct units. **Be careful:** What are the correct units for the MSD? To make your axes have a $\boldsymbol{log_{10} - log_{10}}$ scale, use the following trick; after your figure window has been created, use `gca` to create an axes object that contains information about the most recent figure window you have edited. You can then edit that object to edit the axes scale like so:

```
% create axes object, edit scales
ax = gca;
ax.XScale = 'log';
ax.YScale = 'log';
```

4. **[Output]** If part 3 was done correctly, the MSD curve should be a straight line on the $\boldsymbol{log_{10} - log_{10}}$ for smaller values of Δt (statistical error starts to become a problem

for larger Δt). This implies that the curve follows a *power law*, i.e. the MSD can be described by the following form

$$\text{MSD}(\Delta t) \sim D\Delta t^{\alpha}. \tag{3}$$

Random-walk-type motion is also referred to as *diffusion*, and therefore the D is called the *diffusion coefficient*. For this part, extract the value of the diffusion coefficient and the power-law exponent α from the curve. To do this, note that when you are viewing a curve on a $\boldsymbol{log_{10} - log_{10}}$ scale, you can think about the axes being the logarithms of the values being plotted, i.e. $\log_{10}(\text{MSD})$ and $\log_{10}(\Delta t)$. For (3), taking the $\log_{10}$ of both sides we obtain

$$\log_{10}(\text{MSD}) \sim \alpha \log_{10}(\Delta t) + \log_{10} D, \tag{4}$$

which is the equation of a line with slope α and y-intercept $\log_{10} D$ (which is why power laws are straight lines on $\boldsymbol{log_{10} - log_{10}}$ plots). You can then fit the log_{10} of the data using a straight line to obtain the best-fit slope and y-intercept. To fit the data to a line in MATLAB, use the function called `polyfit` (see the documentation or help page for this function using `doc polyfit` or `help polyfit`). Assign the values of α and D to the variables `alpha` and `diffusionCoefficient`, and the code provided in the template will print the values for you to the command window.

5. **[Output]** We also have a theoretical prediction for the diffusion coefficient: In two dimensions, we expect $D = L^2/\delta t$ where δt is the amount of time between steps, and L is the step length. Determine and store these values in the variables `stepLength` and `stepDuration`, and the code provided in the template will print the expected value for D for you to the command window. How does your calculation from the trajectory compare?

Submission

To submit the assignment, upload your MATLAB scripts `LASTNAME_assignment0_problem1.m` and `LASTNAME_assignment0_problem2.m` to Canvas in the Assignments section.