Problem Set 2

Biological Physics

Due Thurs., Feb. 11, 2015

For this problem set, you will analyze data from a simulation of a particle diffusing through a model cell and find the diffusion coefficient.

The data provided is a simple 2-dimensional simulation of a small particle moving through a model E. coli cell, with a large number of obstacles in the way. Cells are, of course, 3-dimensional, but the data provided is two-dimensional to make simulation and analysis easier.

A) First Look at the Data

1. Make sure you have downloaded `PS2data.mat` into the same folder you are working in.

2. Use the `load` function to load the data into matlab.

3. You will now see three new arrays in your workspace: `locs1us`, `locs100us`, and `locs10ms`. These correspond to location data taken at $1\,\mu$s, $100\,\mu$s, and $10\,$ms intervals. They each have three axes: time, particle number, and coordinate (x or y). Coordinates are in microns (μm).

   Note: due to computational constraints, the three arrays are not exactly the same shape; in particular, `locs10ms` is significantly smaller.

4. **[Output]** Make a scatter plot of all the particle locations from all three arrays, using the `plot(x, y, 'k.')` command.

   Note: The shape of these arrays is 3-dimensional, and to get certain values out of it, you will need to use indexing. For example, to get the x-coordinates of the first 10 particles at all times, you might use `xs = locs(1:end, 1:10, 1);` that's `1:end` for all times (the first axis), `1:10` for the first ten particles (the second axis), and just the x-coordinate (the third axis).

5. **[Output]** Plot (in the same figure) the trajectory of the *first* particle, with the $10\,$ms data in green (`'g-'`), the $100\,\mu$s data in blue (`'b-'`), and the $1\,\mu$s data in red (`'r-'`), in that order so that the $1\,$ms data does not cover the shorter-time data.

6. You should be able to see both the shape of the cell containing the particles, as well as the "voids" where obstacles blocked the paths of the particles.

B) Calculating the MSD

The MSD is the Mean Squared Displacement of the particles, plotted as a function of Δt:

$$\text{MSD}\left(\Delta t\right) = \left\langle \left|\vec{r}_i\left(t+\Delta t\right) - \vec{r}_i\left(t\right)\right|^2 \right\rangle_{t,i}$$

Where $\vec{r}_i\left(t\right)$ is the position of particle i at time t, and we average over all particles (i) and over all times we have information for (t).

1. Calculate approximately 200 Δt values for which we can calculate the MSD.

   (a) First, we need to find the *indices* for which we can calculate Δt values. The minimum *index* would be 1, and the *maximum* would be $N-1$, where N is the length of the array in the time dimension. Find $N-1$ for the two arrays we have. You may want the `size` function.

   (b) The MSD is normally plotted on a logarithmic timescale, so we want our indices to be approximately logarithmically spaced. Use functions like `logspace`, `round`, and `unique` to create an array of indices `ixs` that goes from 1 to $N-1$ approximately logarithmically.

   (c) Now calculate the Δt values, in seconds, for these arrays.

2. Calculate the MSD for these two arrays.

   (a) For this, we are going to want a `for` loop. In this case, we want to loop over the array of indices (`ixs`) that we created before, so we will want something like this:

   ```
   MSD = zeros(size(ixs))
   for j = 1:length(ixs)
       ix = ixs(j);
       drs = arr(1:end-ix, :, :) - arr(1+ix:end, :, :);
       ...
       MSD(j) = ...;
   end
   ```

   Filling in the missing parts (`...`) above, calculate the MSD for each index. Do this for both arrays.

3. **[Output]** Plot the MSD for all three arrays.

   (a) Use `loglog` to plot it logarithmically. Label your axes appropriately, **including units**.

C) Calculate The Diffusion Coefficient and the Scaling Exponent

1. At short times ($\Delta t < 10\,\mu\text{s}$), the MSD is approximately diffusive, and approximately follows $\text{MSD}\left(\Delta t\right) = 4D\Delta t$[1]. Calculate this short-time diffusion coefficient.

   (a) First use array *indexing* to take only the data from each array that is not covered by the following arrays, i.e. $t < 100\,\mu\text{s}$ for the first array, $t < 1\,\text{ms}$ for the second array, and all of the third array. If you have an array `dt` that represents the Δt for an array `arr` of MSD values, you will want code like this:

   ```
   maximum_limit = 100;
   cut_arr = arr(dt < maximum_limit);
   ```

   (b) Use array *concatenation* to combine all three arrays of Δt values into one array, and all the MSDs into one array. Note that you can simply use the code

   ```
   long_arr = [arr1, arr2, arr3];
   ```

   (c) **[Output]** Calculate the diffusion coefficient: using array indexing, find $\left\langle \frac{\text{MSD}(\Delta t)}{4\Delta t} \right\rangle_{\Delta t}$ for values of $\Delta t < 10\,\mu\text{s}$. **What is D?**

   (d) **[Output]** Plot the line $\text{MSD}\left(\Delta t\right) = 4D\Delta t$ on the same figure as the MSD.

2. The obstacles in this simulation are densely packed enough to be close to *percolation*. At the percolation transition, the density of randomly-placed particles is dense enough that the obstacles will not only overlap, but create clusters that may become arbitrarily large, reaching from one end of the box to the other. Close to this density, the behavior of a random-walk becomes *sub-diffusive*: $\text{MSD}\left(\Delta t\right) = c\left(\Delta t\right)^n$. In general, sub-diffusive behavior simply means $n < 1$; for a random walk through a system close to the continuum percolation transition, the sub-diffusion exponent is $n \approx 0.707$[2]. We can see this type of behavior at longer times, $50\,\mu\text{s} < \Delta t < 1\,\text{ms}$. Let's find that exponent.

   (a) First create two arrays that we can fit to: $x = \Delta t$ and $y = \text{MSD}$, but only in the range $50\,\mu\text{s} < \Delta t < 1\,\text{ms}$. Use the steps above to join the data from all three datasets (1a, 1b).

   (b) **[Output]** Now we fit to $\text{MSD}\left(\Delta t\right) = c\left(\Delta t\right)^n$. You can take the log of both sides to get $\log \text{MSD} = n \log \Delta t + \log c$, which looks a lot like $Y = nX + \log c$. Then you can use the `polyfit` function to fit to this, and extract the exponent n and log-of-prefactor $\log c$ from the result of `polyfit`. **What values do you get for n and c?**

   (c) **[Output]** Plot the line $\text{MSD}\left(\Delta t\right) = c\left(\Delta t\right)^n$ on the same figure as the MSD.

[1] Note that the full formula is $\text{MSD}\left(\Delta t\right) = 2dD\Delta t$, where d is the number of dimensions.

[2] This exponent was calculated and simulated in 3D in Höfling *et al.* (2006); using the same equations, $n = 0.717$ can be calculated for 2D.

D) Problem Set Submission

As before, make a zip (or tar) file with your code, all your graphs, and the answers to the questions. The MSD graph should be in `.eps` format, and the trajectories plot may be in `.png` if the `.eps` file is too large. Upload to the classesv2 server, *named with the format* ***`LASTNAME-FIRSTNAME-PS2.zip`*** (or `.tar` / `.tgz`).

References

F. Höfling, T. Franosch, and E. Frey, Phys. Rev. Lett. **96**, 165901 (2006).