Computational Physics Weekly Assessment

Week 7: random walk

Weekly assessment task & hints

You will produce simulation of 'Chemotaxis', a modified random walk used by some bacteria to find food sources. To briefly recap this week's lecture notes, Chemotaxis is a 'run and tumble' process in which a bacteria is either moving on a constant bearing and speed, or is tumbling. A tumble lasts for a fixed time, and no motion occurs during a tumble. After a tumble the bacteria travels at its constant speed along a new, random bearing uncorrelated with the previous one.

You will be simulating this walk on a 2-dimensional surface where some energy source such as sugar exists at a density specified by the function *f(x,y)* where:
$f(x,y) = 4000 - \left(x^2 + y^2\right)$ (the units for x and y are microns)

The bacteria modifies its random walk to make it more likely that it moves towards a food source by relating the probability of a tumble occurring to the rate of change of the energy field, such that it is less likely to tumble when travelling in a direction that increases the energy density, and more likely to tumble when it is decreasing. Model this by making the half-life (in seconds) of a run event proportional to the temporal energy gradient seen by the bacteria: $t_{1/2} = 1 + k\frac{df}{dt}$ where k is a constant defining the sensitivity of the bacteria. You should ensure that large negative gradients do not produce a negative half-life, but that the half-life never becomes smaller that 0.1s. ← abs?

You cannot derive the rate of change of the energy field with time analytically, as this is a quantity experience by the bacterium due to its random walk. The bacterium constantly samples the energy density as it moves and uses a biological filter to determine the rate of change with time. Simulate this system by sampling the current energy density f(x,y) at every time point and build a record of f(x,y) against time. The gradient is then calculated by comparing the current value with the appropriate historic value from 1 second ago. An efficient implementation of this would use a Python list as a shift-register as described in the lecture – thereby only keep a short record of 1 sec duration.

$$\frac{df}{dt} \approx f(x,y)\big|_{t=t} - f(x,y)\big|_{t=t-1}$$

Assume that the bacterium moves at a constant speed of 2 microns/sec and that the sensitivity, k, is 0.2. Assume that a tumble event lasts for 0.1 seconds.
Produce a function that simulates the motion of bacteria from a specified position (the origin) over a series of equally spaced timesteps covering 100 seconds. The function should return the trajectory of the bacteria.

Outputs:

Your code should illustrate the process of Chemotaxis on a matplotlib figure. Launch many bacteria from the position (x=20,y=40), which is some distance from the location of maximum energy density (0,0). Produce a figure with 3 subplots as shown in the lecture slides

1. The top left graph should show the energy field as a 2D greyscale plot with 20 different bacteria tracks overlaid.
2. The top right graph should show a simplified trajectory with markers and only the first and last points
3. The bottom graph should show the mean square displacement of the bacteria against time. Plot two lines – one for the MSD from the bacteria's origin and one showing the MSD from the location of maximum energy.

You should also use the variable ANSWER1 to answer the question "What effect does the sensitivity, *k*, have on the bulk behavior of the bacteria? Specifically consider the cases where *k* is too big and where *k* is too small."

Maths recap – the probability calculations you will need are the same as those for radioactive decay. The "half life" of a run event is given by $t_{1/2} = 1 + k\frac{df}{dt}$; this is defined by the behavior of our bacterium. The mean lifetime of a run event is given by $\tau = t_{1/2}/\ln(2)$. Given the mean lifetime, you can calculate the probability of not tumbling, P_{nt}, in an interval dt as $P_{nt} = e^{-dt/\tau}$

Random numbers recap - there are many ways to get a random number as covered in previous weeks; for example "random.random()" gives a number from a uniform distribution between 0 and 1. This may be multiplied by a constant to re-scale it into another range such as the range of angles that may occour after a tumble event.

Shift registers – a shift register is a storage element that contains a fixed number of entries in a sequence. When a new entry is added, the oldest one is discarded. This provides a convenient way to remember the energy levels experienced by the bacteria over a second. One example is given below:

```
shift = [0, 0, 0, 0, 0, 0, …, 0]  # Initial "pump priming" values; consider
if there is a better value than an energy of 0
for each timestep:
       r = … # Calculate new position
       eNew = f(r) # New energy level
       shift.append(eNew) # add to the python list
       shift = shift[-10:] # keep only the 10 most recent entries
       de = shift[-1] - shift[0] # [-1] is the newest entry, [0] is the
oldest
       t_half = … # some function of de/dt
```

Hints:

1. Think about what would be a suitable, and simple, timestep to implement.

2. Think carefully about how the half-life of a run event relates to the probability of tumbling (i.e. ending a run event) within one timestep. Consider the parallels to radioactive decay.
3. **DEVELOP YOUR SOLUTION IN INCREMENTAL STAGES**. It is suggested that you start in the absence of an energy field and with a constant probability of tumbling. Once this displays the expected behavior you can add energy sensitivity. Likewise, start with simulating and plotting a single trajectory, then add a for loop to generate and plot multiple simulations, before finally collecting and analyzing the trajectories to determine the MSD.
4. See hint 3.
5. Think about how many bacteria to launch – you want to see the bulk statistical behavior in the MSD, not individual effects. However, you don't want the simulation to take to long to run

General comments:

- Place two variables at the start of the code,
  o USER="your name"
  o USER_ID = "your CIS login"
- Your module must run in order to be awarded any marks
- Your solution should contain no more than about 95 lines of code, comments and whitespace. Excessively long submissions may loose marks.