Ants on a ring

You will investigate the application of Monte Carlo and PDE techniques on the same problem. On a 1D ring of circumference L (i.e., and interval of length L with periodic boundary conditions), consider two cases:

1. A density field P that satisfies the following partial differential equations:

$$\frac{\partial P}{\partial t} = D\frac{\partial^2 P}{\partial x^2}$$

   (the diffusion or heat equation), where t is time, x is the coordinate along the ring ($0 \leq x < L$) and D is the diffusion constant.

   The initial condition is such that all density is concentrated at $x = 0$, i.e.,

$$P(x, t = 0) = \delta(x - L/2).$$

2. Alternatively, divide the ring up into N segments of length Δx. Z walkers (they could be ants) are randomly hopping from segment to segments.

   At each time step, which is defined to take a time Δt, these walkers move to the next segments to the left or the right with probability p (thus, the probability of staying is $1 - 2p$).

   Initially, all walkers are on the middle segment $n = N/2$.

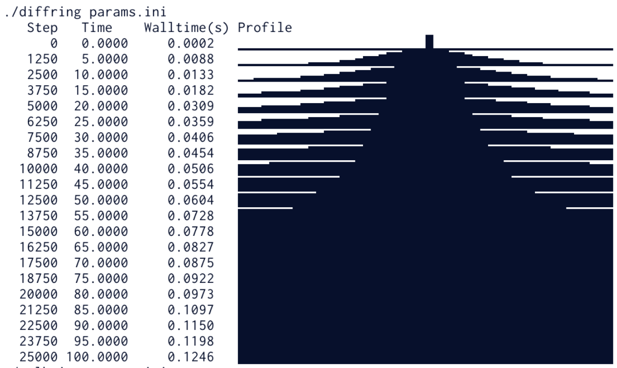
To solve case 1 numerically, we will discretize time using a time step of the same size Δt as in case 2, and we will discretize space using the same spacing Δx as well. We will use an explicit forward Euler for the time step, with the numerical second derivate compute as explained in the PDE lecture notes, and where we equate $D = \frac{p\Delta x^2}{\Delta t}$, for then it can be shown that these two cases become equivalent, in the sense that P is the probablity distribution for the random walkers. We therefore expect

$$P(n\Delta x) = \lim_{Z\to\infty} \frac{a_n}{Z\Delta x}$$

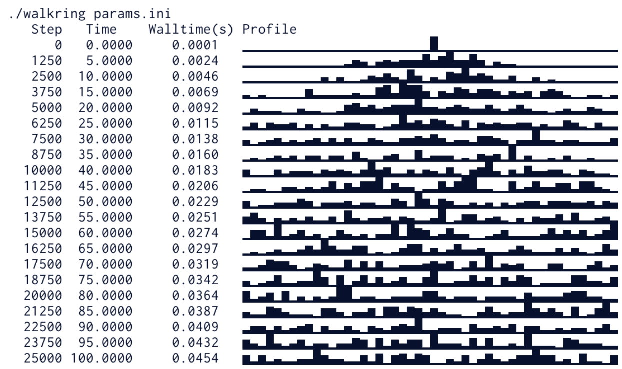
Where a_n is the number of walkers on segment n.

To get you started, you are given the codes for two applications, `diffring` and `walkring`, which provide a framework to solving the two cases above separately. The codes are already modularized into the following files:

`diffring.cc`	Driver for the diffusion code
`diffring.ini`	Parameters file read in by the code
`diffring_timestep.h`	Module to do one diffusion time step
`diffring_timestep.cc`	
`diffring_output.h`	Module to do output
`diffring_output.cc`	
`walkring.cc`	Driver for the random walk code
`walkring_timestep.cc`	Module to do one random walk time step
`walkring_timestep.h`	
`walkring_output.cc`	Module to do output
`walkring_output.h`	
`parameters.cc`	Module to read in the parameter file
`parameters.h`	
`params.ini`	Parameters file read in by the both applications
`ticktock.h`	Module for timing
`ticktock.cc`	
`sparkline.h`	Module to produce inline graphs (uses unicode tricks)
`sparkline.cc`	
`Makefile`	Combined makefile for the two applications



(a) Output of `diffring` (walltimes may vary).



(b) Output of `walkring` (walltimes may vary).

To get this code, clone it on the teach cluster with the command

```
git clone /scinet/course/phy1610/antsonring
```

Note that the code requires the following libraries:

- rarray
- boost
- openblas

To run this code, `cd antsonring && make run` would work and give the ouput shown in the figures above. However, there are a few pieces missing in `diffring_timestep.cc` and `walkring_timestep.cc`:

1. `diffring_timestep.cc` misses the initialization of the matrix F that is used to perform the time evolution (blas) matrix-vector multiplication in the `diffring_timestep` function. Note that most of the matrix is similar to the one shown on page 14–18 on the slides of lecture 12. To implement the periodic boundary conditions, the first and last row and the first and last column differ. In particular, if the number of grid points is N, one should make sure that one satifies:

$$P_{i+1}[N-1] = P_i[N-1] + \frac{D\Delta t}{\Delta x^2}\Big\{P_i[N-2] + P_i[0] - 2P_i[N-1]\Big\}$$

$$P_{i+1}[0] = P_i[0] + \frac{D\Delta t}{\Delta x^2}\Big\{P_i[N-1] + P_i[1] - 2P_i[0]\Big\}$$

   This should introduce non-zero elements `F[0][N-1]` and `F[N-1][0]`.

2. `diffring_timestep.cc` also misses the appropriate (blas) matrix-vector multiplication in the `diffring_timestep` function.

3. `walkring_timestep.cc` misses the random dynamics of walkers on the ring in the `walkring_timestep` function.

Your task is to

1. Provide the missing pieces (note that periodic boundary conditions imply the matrix for case 1 is not 'banded')

2. Determine qualititively roughly how many walkers you need to get close to the equivalence (you can do this visually for this assignment).

3. Write a report on what you did. You should explain the how and why of the implementation of the time stepping routines. Give details as to what each part of the time stepping functions does.

Due date: March 19, 2020. Use version control and comment your code as usual. Do not forget to add your report to the repo, and upload the git2zip-ed repo to the Assignment Dropbox on the course web site `https://courses.scinet.utoronto.ca/468`.