Name: Joe
Date: 26/10/22

We need to calculate the structure factor of the system. This can calculated by finding the Fourier Transform of the order parameter ϕ and then finding $\langle\phi(\boldsymbol{k})\phi(-\boldsymbol{k})\rangle$, or $\langle\phi(\boldsymbol{k})\phi(\boldsymbol{k}+\boldsymbol{k_0})\rangle$ for some Fourier space element k_0 (basically the Fourier analogue of calculating the Correlation Function $\langle\phi(\boldsymbol{x})\phi(\boldsymbol{x}+\boldsymbol{r})\rangle$, where $\boldsymbol{k_0}$ and $\boldsymbol{r}$ are arbitrary vectors in Fourier and real space). I'm not clear on which equation to use.

Once we have the structure factor S, we can plot it against k for various times t, and we'll find that they produce different decay curves. For larger t, we'll find that the decay curve is steeper. If we scale the k axis by $t^{1/2}$ or $t^{-1/2}$ (I'm not sure which yet), all the curves will fall onto each other.

From this, we should be able to extract the dynamic scaling exponent, which has a value of $z = 2$.

$\nabla\phi$ can discretised along a square grid with separation Δx in each direction.

$$\nabla^2\phi \approx \frac{\Delta\phi}{\Delta x^2} = \frac{\phi_{i+1,j} + \phi i - 1, j + \phi i, j + 1 + \phi i, j - 1 - 4\phi}{\Delta x^2} \tag{1}$$

Also check out "Granular Media.pdf", which is the "PRL 3" article on the Moodle page. Also look at "Growth Laws" in Bray's paper (not sure if section 2.5 or section 7, try both).

Name: Joe
Date: 27/10/22

I've spent some time coding the basic diffusion equation:

$$\frac{\partial\phi}{\partial t} = \nabla^2\phi \tag{2}$$

Using equation 1 to discretize the Laplacian operator. It was vectorised using the following logic.

We can write the order parameter as some vector $\boldsymbol{\phi}$. Suppose we look at a 3×3 lattice:

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_{00} & \phi_{01} & \phi_{02} \\ \phi_{10} & \phi_{11} & \phi_{12} \\ \phi_{20} & \phi_{21} & \phi_{22} \end{pmatrix} \tag{3}$$

We argue that for periodic boundary conditions, when considering a term for the Laplacian such as $\phi_{i+1,j}$, when $i = 2$, $i + 1 = 0$. Conversely, for $\phi_{i-1,j}$, when $i = 0$, $i - 1 = 2$.

The `Numpy` function `numpy.roll` enables us to move elements in a matrix with ease. To move each row one row down, including moving the bottom row to the top, we type:

```
np.roll(phi, N)
```

Where `N` is the length of a row or column (in this case, `N=3`). To move one column over, we type:

```
np.roll(phi, 1, axis=1)
```

This is given a little more detail in the Github file `RollingMatrices.py`.

When we discretize equation 2, we can solve it using `Scipy`'s `odeint` function from its `Integrate` module. Since `odeint` only accepts one-dimensional inputs, we can reshape the $\boldsymbol{\phi}$ lattice into an N^2 long array and give it to `odeint`. Then, we can use `numpy.roll` on the lattice and perform all the discretized Laplacian calculations in one matrix. We then reshape the lattice again to make it more palatable for `odeint` and it repeats this process.

After `odeint` has given us the solution, we can retrieve the lattice by reshaping each row of the solution array. Currently, this is done using a for loop, which isn't really ideal, but I'm not sure how to vectorise it yet.
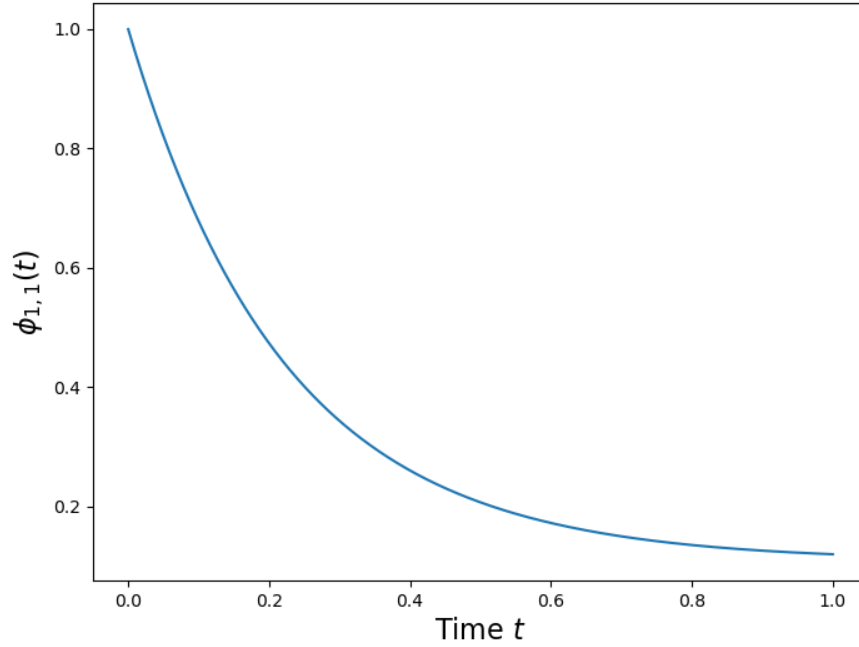
Figure 1: Plotting $\phi_{1,1}$, which was initialised to have the value $\phi = 1$, against time t. It displays exponential decay behaviour, which is what we expect.

Then, again using for loops because vectorising things is difficult, we can retrieve ϕ for a given lattice site for every point in time. This gives the plot in figure 1.

Since ϕ decays exponentially here, we could use some `SciPy` curve fitting to find decay constants, although I don't think this will be as important when we include a potential, as we're required to.

We'll need to animate this to make what's happening a lot clearer. This graph works great for a single lattice site, but it's not any good for the whole lattice. I remember Moriarty had some things written on animations.