Particle Models

October 24, 2019

1 Model Implementation

The main difficulty in moving from the space homogenous to the two dimensional case is calculating the heterogeneity for every particle.

The problem term is:

$$G\left(\frac{\frac{1}{N}\sum_{j=1}^{N} v_t^{j,N}\varphi(x_t^{j,N}-x_t^{i,N})}{\frac{1}{N}\sum_{j=1}^{N}\varphi(x_t^{j,N}-x_t^{i,N})}\right). \tag{1}$$

Let's unpack this piece by piece, and discuss their implementation. The herding function G, is identical to before, with a new option `G_Garnier`, corresponding to the variable well-depth function given in Garnier's paper:

$$G(u) = \frac{h+1}{5}u - \frac{h}{125}u^3$$

Calculating the interaction is more difficult computationally. For every particle we must calculate the distance to every other particle in the system. A first implementation is shown below.

```
x_curr = x[n] #Particles' current positions
    for particle, position in enumerate(x_curr):
        interaction = phi(np.abs(x_curr - position) % L)
        weighted_avg = np.sum(v[n, ] * interaction)
        scale = np.sum(interaction)
        interaction_vector[particle] = weighted_avg / scale
```

For each particle in the system, we calculate the difference between all positions and the particle position, modulo the length of the domain. This is equivalent to $\min(x_i - x_j, L - (x_i - x_j))$ but requires less operations. The implementation here should be quick, as it exploits NumPy's ability to broadcast arrays of different sizes. The state of the system is then updated as before.

```
x[n + 1,] = (x[n,] + v[n,] * dt) % (L)  # Restrict to torus
v[n + 1,] = (v[n,] - (v[n,] * dt) + G(interaction_vector) * dt
             + D*np.sqrt(dt) * normal(size=particles))
```

Should the interaction include the current particle? As it's written, the particle interacts with itself. Could be attributed to some sort of "inertia". In the case of one particle, (1) becomes

$$G\left(\frac{v_t^1\varphi(0)}{\varphi(0)}\right) = G(v_t^1).$$

The equation of motion for this particle is then

$$\mathrm{d}v_t^1 = -v_t^1\,\mathrm{d}t + G(v_t^1)\,\mathrm{d}t + \sqrt{2\sigma}\,\mathrm{d}W_t^1.$$

This is very different from a free particle moving in a potential well. It will move between velocity ± 1, not 0 as would be expected for a free particle.

Obviously this effect decreases very quickly as $N \gg 1$. Removing the particles self-interaction is not difficult.

1.1 Differences in Garnier's Model

Garnier's model differs in three ways: the scaling of the interaction, the size of the spatial domain and the diffusion coefficient. The diffusion coefficient here is $\frac{\sigma^2}{2}$ instead of just σ. No changes to the implementation are necessary here, other than minor plotting changes. The potential function is also easily implemented. Changing the length of the domain from 2π doesn't materially affect anything as it can just be scaled back to the unit circle. I can't see any reason for using a longer domain, other than it avoiding a call to NumPy. The only difference that may have an effect is the scaling of the interaction. Rather than counting the number of particles that contributed to the interaction, the total number of particles is used. This difference is mentioned in Section 8, where they conclude that it will have an effect on the critical diffusion value that makes the order states stable. As far as I can tell no simulations were done to confirm this.

Scaling by N will reduce the input to G, biasing the velocity towards 0. Is this why in low noise we see velocity away from the expected value? Try running for with each denominator – still get cluster, but avg vel is much closer to where it is expected to be.

2 Sanity Checks

To check this implementation is working as expected, we can exploit the analysis and the earlier code. Things to check:

i) If $\varphi \equiv 0$, does the interaction return zero?

   (a) Does the system converge to $\mathcal{N}(0, \sigma)$?

ii) If $\varphi \equiv 1$, does the `interaction_vector` return the average velocity?

   (a) Does the system converge to $\mathcal{N}(\pm\xi, \sigma)$, where $G(\xi) = \xi$?

iii) If $\varphi = I_{[0,1]}$ and we place say 5 particles on the torus deterministically, does the code return the same as we calculate by hand?

First we will verify the interaction calculations, before testing convergence.

+++ Run test_sanity for all phi calcs +++ +++ How can we check convergence? Could do KL div in x and v, CL2 disc? Everybody loves the eyeball or do error between old code and new?+++

3 Garnier Figure Reproduction