Particle Models

November 3, 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):
        dist = np.abs(x_curr - position)
        interaction = phi(np.minimum(L-dist, dist))
        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.

2.1 Interaction Checks

The test functions are contained in `test_sanity.py`. All tests take as input a random vector of length 1000 as simulated data. When $\varphi \equiv 0$, we simply check every entry is zero. For $phi \equiv 1$, we expect the interaction to be the mean of all velocities. The functions `test_zero()` and `test_ones()` assert this. The final hand check of $\varphi = I_{[0,1]}$ is in `test_hand()`.

2.2 Convergence Checks

To assess the convergence, we can use firstly a visual check using a histogram, then the Kullback-Leibler divergence for normal and uniform distributions and the centred L_2 discrepancy for the spatial distribution.

Aside: Calculating KL and CL2

The Kullback-Leibler divergence already exists in Python as `scipy.stats.entropy()`. The centred L2 discrepancy however does not. It is quite slow to calculate as it involves a double sum, not sure if there's any way around that. Trying to be Pythonic and using `np.meshgrid` actually ended up being 4 times slower. This may be because of the size of the meshgrid, but I'm not sure.

```
def CL2_mesh(x, L=(2*np.pi)):
    n  = len(x)
    term3 = 0
    term2 = np.sum(2. + np.abs(x/L - 0.5) - np.abs(x/L - 0.5)**2)
    x_i,x_j = np.meshgrid(x,x)
    term3 = 0.5*np.sum(2+ np.abs(x_i/L - 0.5) + np.abs(x_j/L - 0.5)
         - np.abs(x_i/L - x_j/L) )

    CL2 = (13/12) - (term2 - term3/n)/n
    return CL2
```

One of the for loops has been removed here, this implementation seems to suffice.

```
def CL2(x, L=(2*np.pi)):
    n  = len(x)
    term3 = 0
    term2 = np.sum(2. + np.abs(x/L - 0.5) - np.abs(x/L - 0.5)**2)
    for i in range(n):
    term3 += np.sum(1. + np.abs(x[i]/L - 0.5)/2 +
        np.abs(x/L - 0.5)/2 - np.abs(x[i]/L - x/L)/2)
    CL2 = (13/12) - (term2 - term3/n)/n

    return CL2
```

Again these calculations need checking, I've done that using `test_CL2()` which simulates uniform random numbers and compares their mean and variance against the theoretical CL2 values given in Garnier's paper. Running the function plots the discrepancy and mean as well as the predicted mean.

We know that if $\varphi \equiv 0$ the system should converge to a Gaussian velocity with mean 0 and a uniform spatial distribution. We can compare the results here with the earlier space homogeneous system. This won't be very illuminating, as the checks on the interaction calculation seem water tight. The diffusion parameter was chosen so that $\sigma \neq \sigma^2$ and $\sigma \neq \sigma^2/2$, basically so that the difference between Garnier's method can be seen.
KL divergence was 0.000643480399. CL2 discrepancy was 2.9874356×10^{-5}, theoretical mean is 8.333×10^{-5}

```
particles = 2000
diffusion = 4
t,x,v = run_full_particle_system(interaction_function="Zero",
particles=particles,
D=diffusion,
initial_dist_v=np.random.normal(loc=0.5, scale=np.sqrt(2), size=particles),
T_end=100
)
```

Set up for normal convergence both positive and negative with `step_G` and Garnier G. Plots of histograms.

```
particles = 2000
```

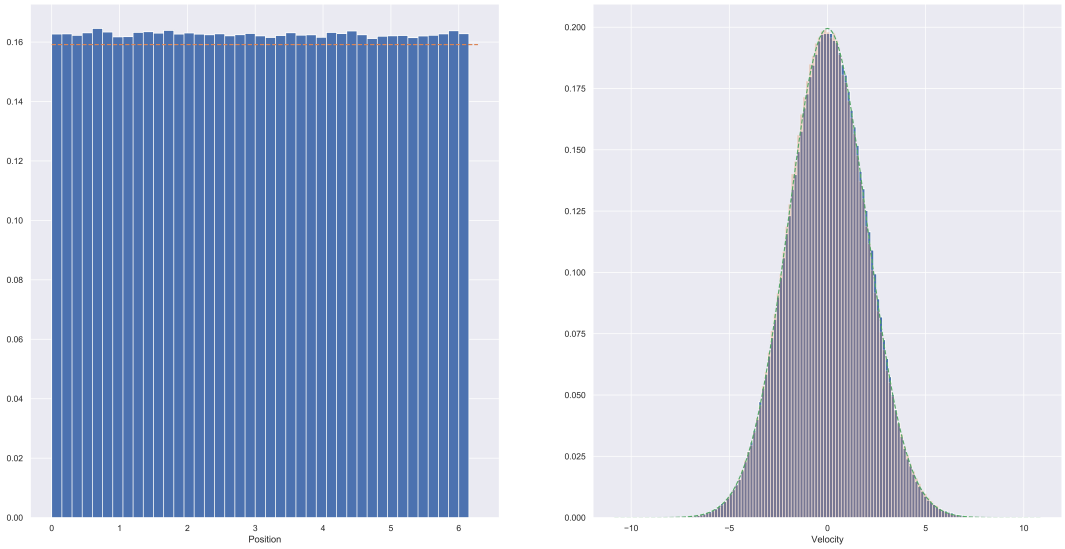

Figure 1: Histograms of position and velocity for no interaction. Orange shows same setup for homogeneous system. Dashed lines are theoretical stationary measure (i.e. $\mathcal{N}(0, \sigma)$ in velocity, uniform in space)

```
diffusion = 4
startTime = datetime.now()
t,x,v = run_full_particle_system(interaction_function="Uniform",
herding_function="Step"
particles=particles,
D=diffusion,
initial_dist_v=np.random.normal(loc=0.5, scale=np.sqrt(2), size=particles),
T_end=100
)
```

Finally just to check the Garnier interaction works as expected.

```
particles = 2000
diffusion = 4
well_depth = 6
xi = 5*np.sqrt((well_depth-4)/well_depth)
startTime = datetime.now()
t,x,v = run_full_particle_system(interaction_function="Garnier",
herding_function="Garneir"
particles=particles,
D=diffusion,
initial_dist_v=np.random.normal(loc=0.5, scale=np.sqrt(2), size=particles),
T_end=100
well_depth=well_depth,
L=10
)
```

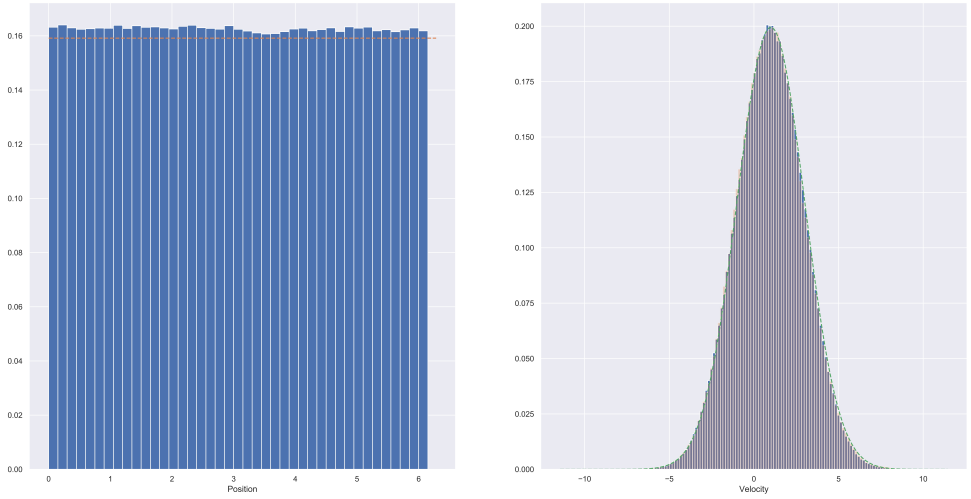

Figure 2: Histograms of position and velocity for uniform interaction. Orange shows same setup for homogeneous system. Dashed lines are theoretical stationary measure (i.e. $\mathcal{N}(1, \sigma)$ in velocity, uniform in space). KL Divergence was 0.0023419970369 and CL2 discrepancy was 5.3347×10^{-5} with expected value 8.33×10^{-5}. Convergence to -1 has also been checked with similar results.

3 Garnier Figure Reproduction

Satisfied that the interaction is correct and the heterogeneity is included properly. To reproduce Garnier's results we should be able to just plug their parameters into our model, remembering the difference in diffusion ($\frac{\sigma^2}{2}$) and the scaling of the interaction as mention in Section 1.1. I've a feeling the scaling affects the results. All axes are scaled to look exactly like Garnier's.

Figures 8, 9 and 10 are not particularly illuminating so I won't reproduce them.

4 Experiments to try

How can we get clustering in our model? Can we show that it is transient?

— Garnier parameters with proper weighted scaling like Buttá. Aim: show that lower avg velocity is artefact of scaling. (See below).

— Garnier parameters but with smooth/step G. Aim: convert Garnier's results to our model.

— Garnier parameters but smaller timestep/less particles/longer time. Aim: show clustering is transient.

— Implement interaction that gets stronger the further away particles are. Zero, Linear, Quadratic, etc. Aim: Force clustering, something Jakub said might work as similar things do for fractional Laplacian.

— Start with non-uniform distribution in space. Aim: show that clusters can be preserved.

— ???

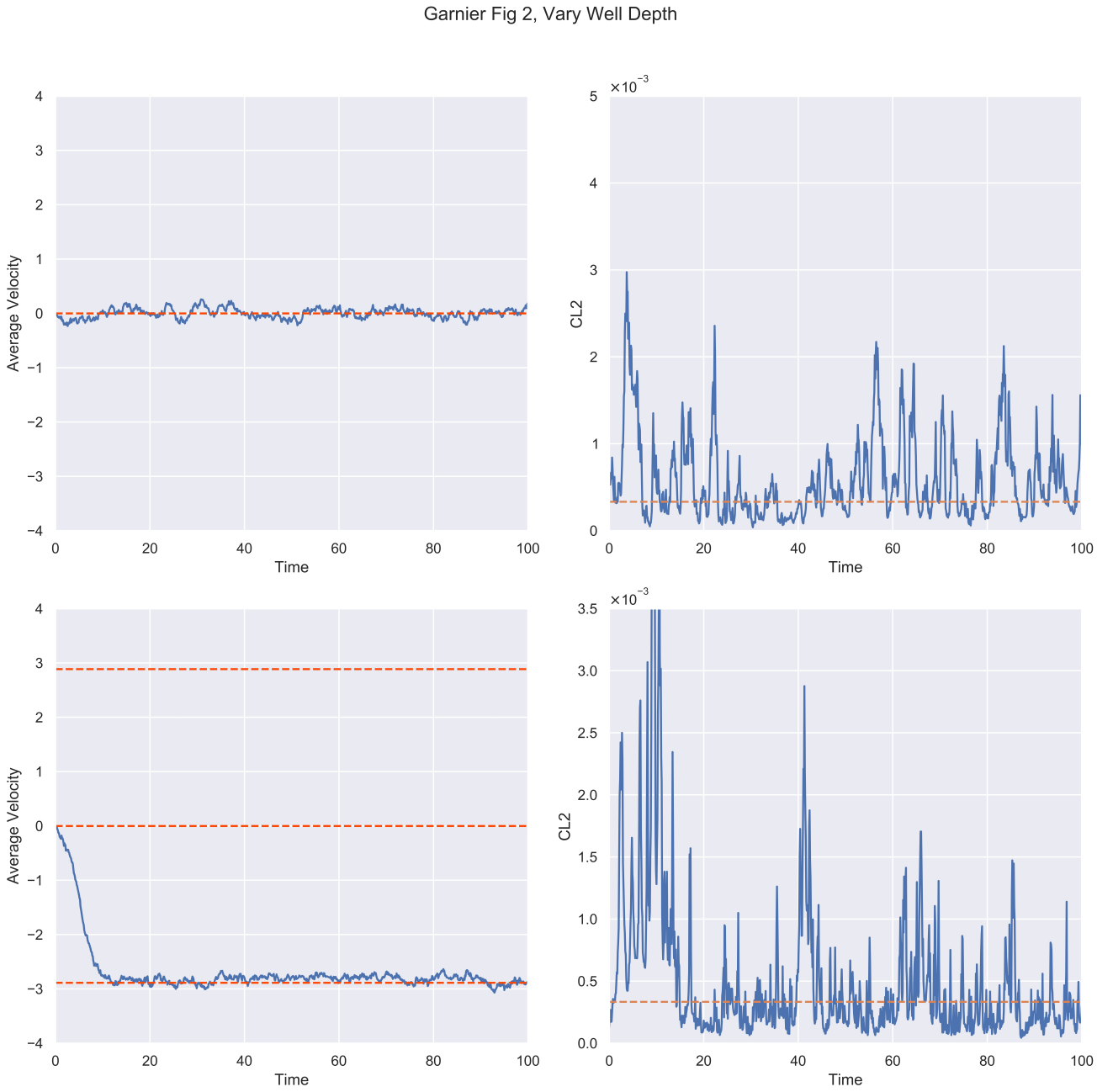

Figure 3: Garnier Figure 2: Average velocity and CL2 discrepancy for $h = 2$ (top) and $h = 4$ (bottom). This is pretty much identical to Garnier's.

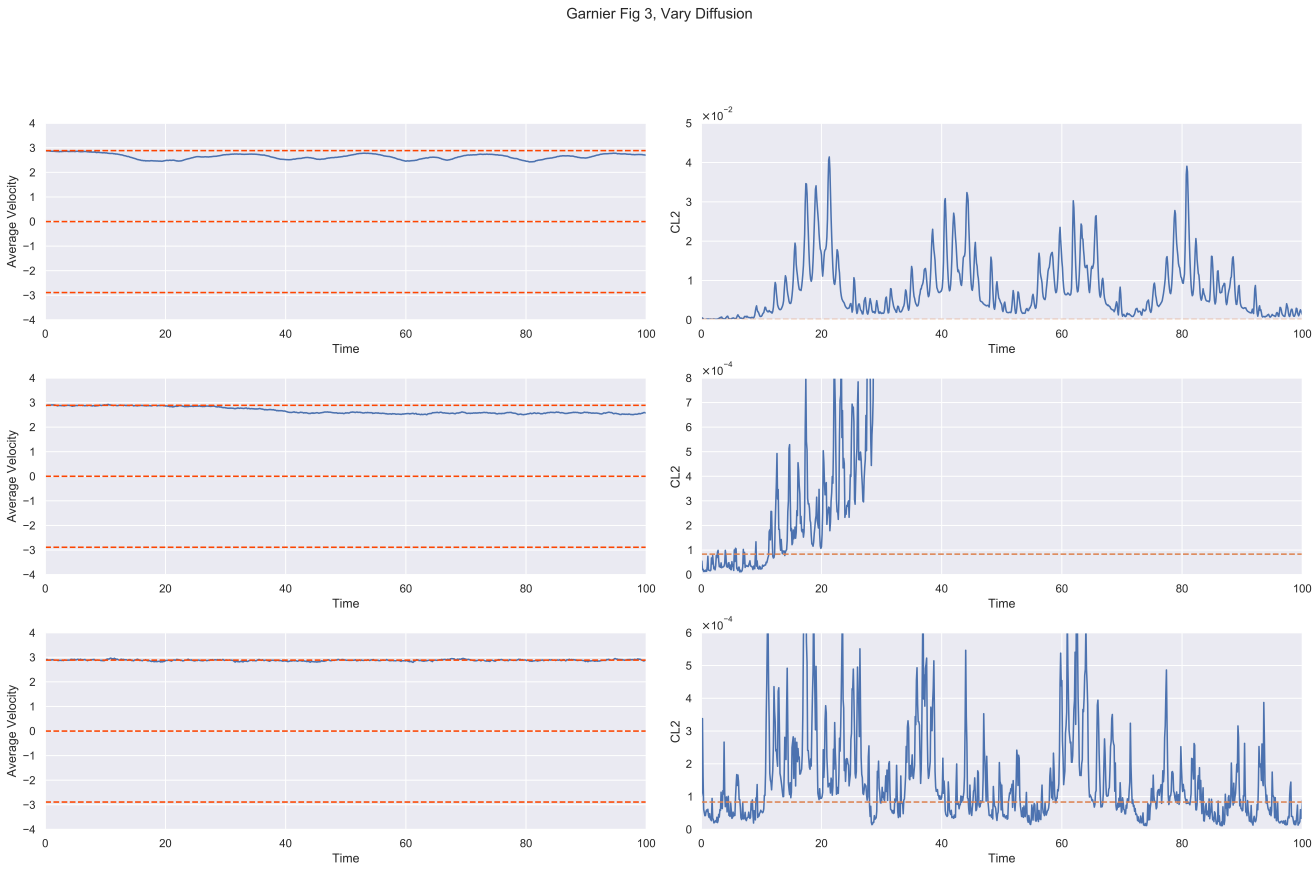

Figure 4: Garnier Figure 3: Average velocity and CL2 discrepancy for $\sigma = 0.5^2/2$ (top), $\sigma = 1^2/2$ (middle) and $\sigma = 1.5^2/2$ (bottom). A cluster forms when $\sigma = 1^2/2$? Proposition 3 just says *'σ large enough'*, no bound is given.

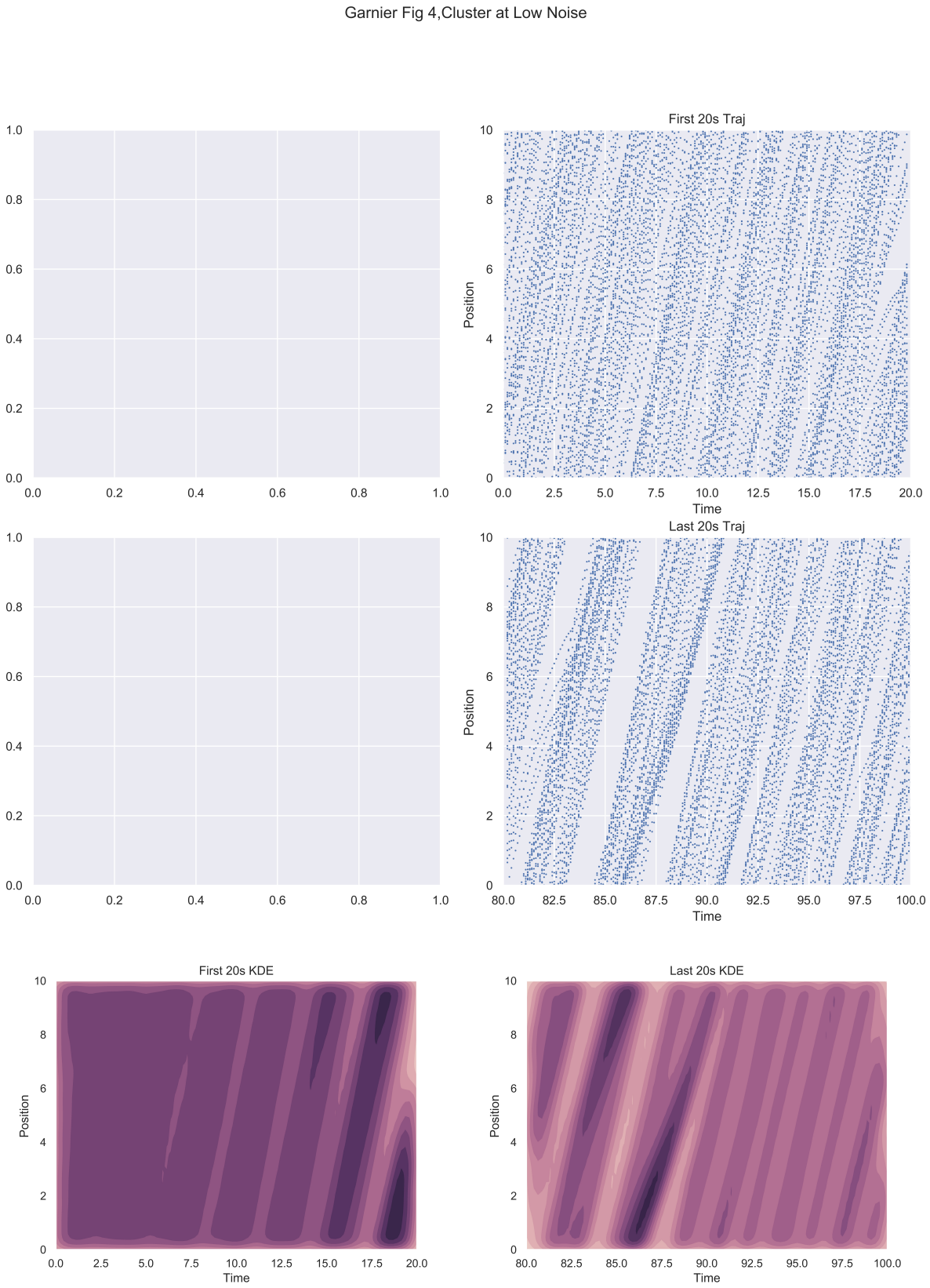

Figure 5: Garnier Figure 4: Clusters occurring at low noise from stationarity. KDE Plots are harder than you'd think to place on a grid. Agrees with Garnier's result, although looks like the cluster is dissipating towards the end. A few more runs might show some transience, or a KDE plot for the whole 100s. The cluster could be forming and dissipating.

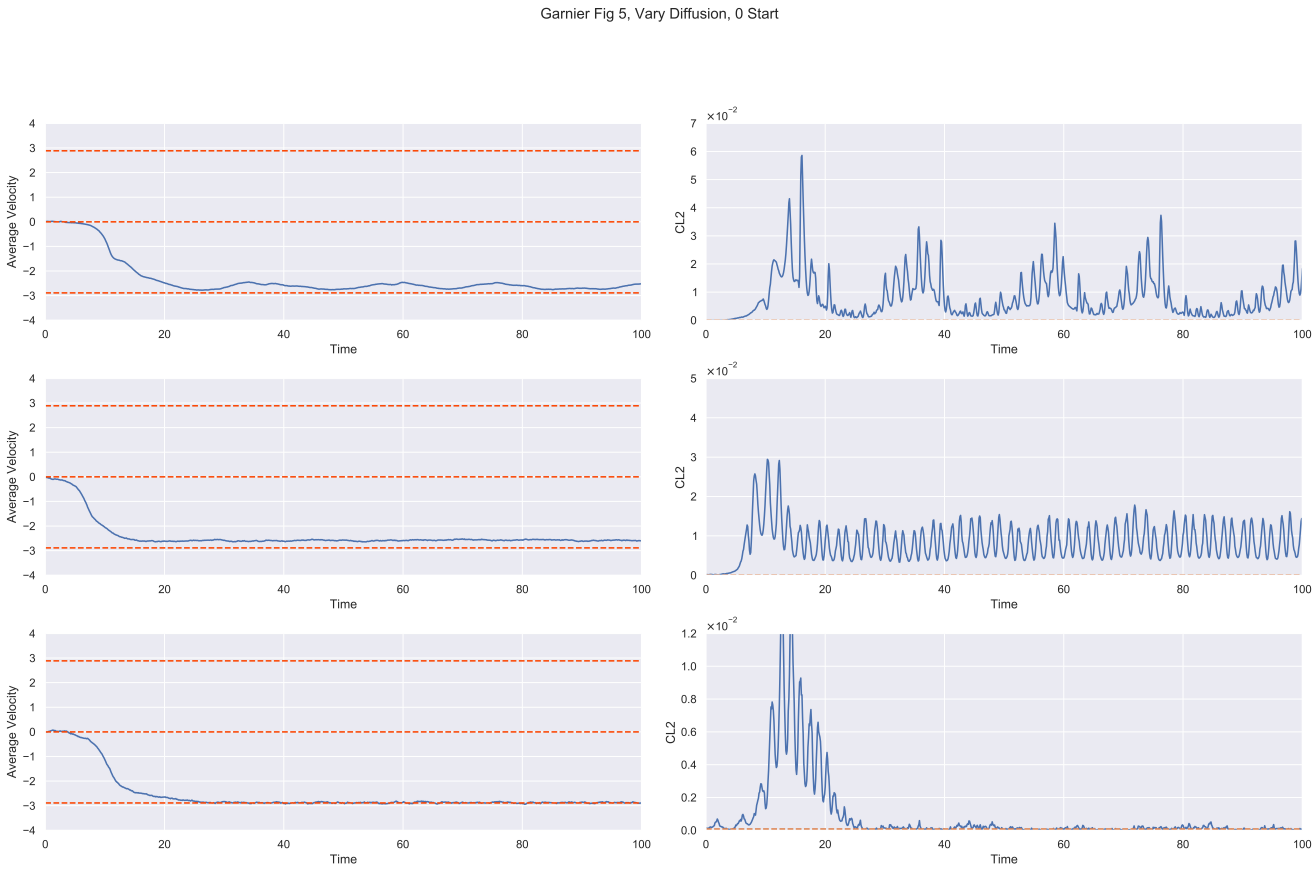

Figure 6: Garnier Figure 5: Clusters occurring at low noise. Setup is identical to GFig 3, but initial data for velocity has mean 0. Very similar to Garnier's result.

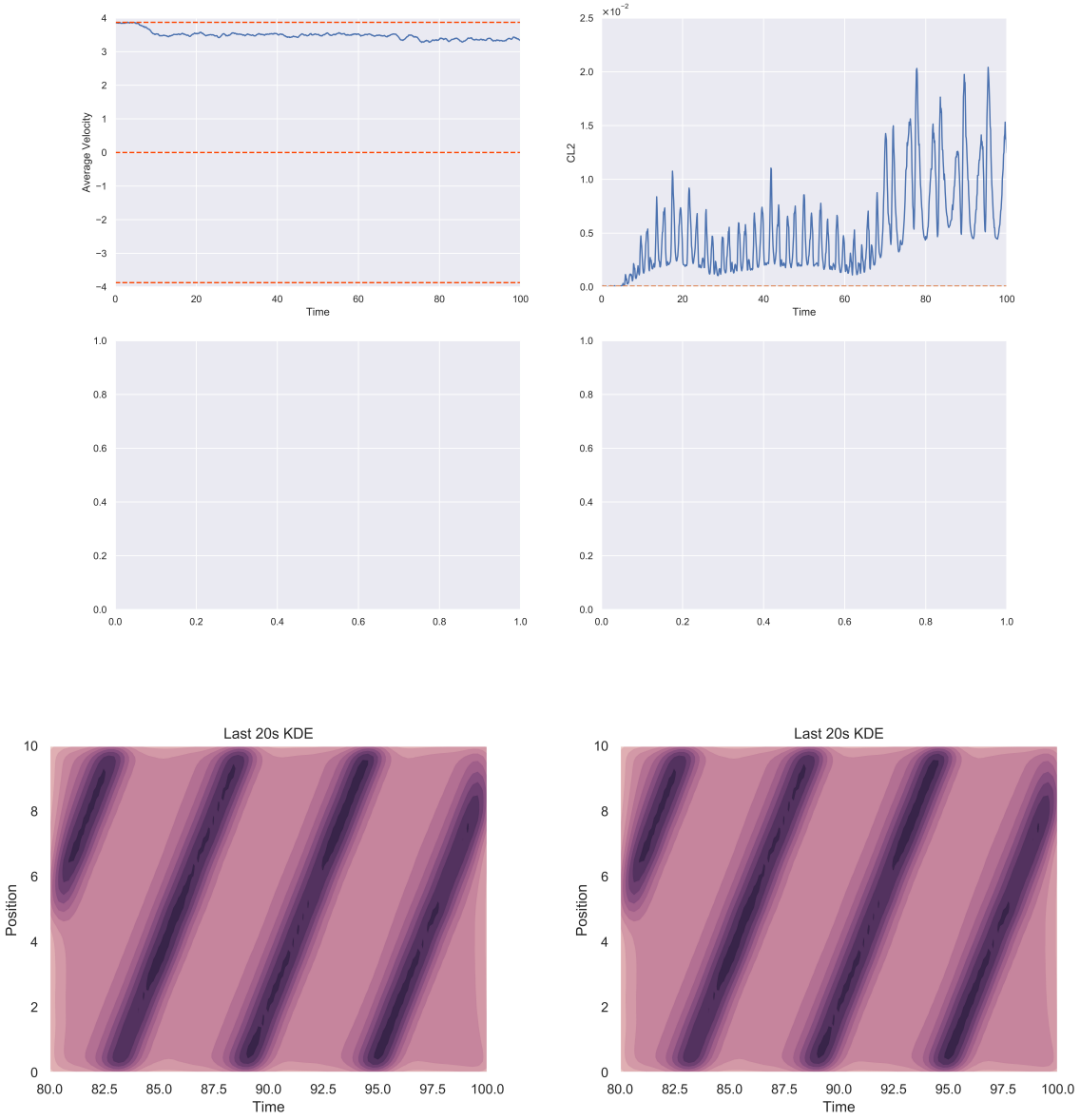

Figure 7: Garnier Figure 6: Cluster forming out of stationarity. Well depth here is 10 and $\sigma = 1^2/2$. It is unclear how Garnier got a cluster forming here. +++COMPARE WITH GARNIER.+++

4.1 Scaling Garnier with Number of Particles Interacting

Here we reproduce the Garnier figures but with the scaling of the interaction equal to the number of particles that the current particle interacts with, instead of the total number of particles. Garnier calls this N_i and mentions it in Section 8. I think the effect will be to speed up any changes as the interaction is stronger at every step (G is increasing, scaling by the number of particles reduces the change in velocity from interaction). Is this why clusters appear in Garnier? They randomly form and dissipate, but this happens too quickly when scaled properly?

4.2 Interesting Bits

Cluster forming when $\sigma = 1^2/2$ in Figure 4.

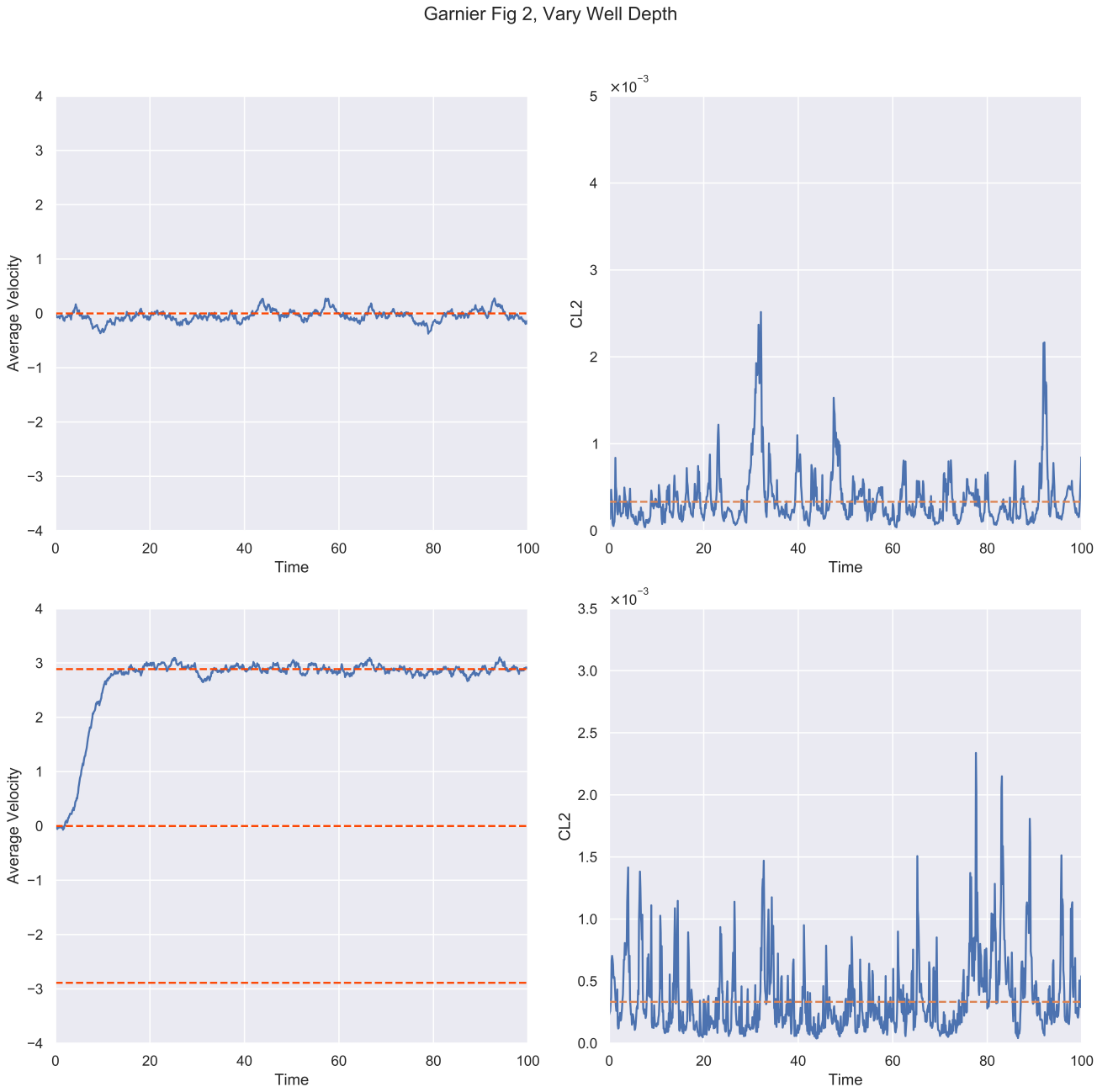

Figure 8: As in Garnier Figure 2 but scaled: Average velocity and CL2 discrepancy for $h = 2$ (top) and $h = 4$ (bottom). This is pretty much identical to Garnier's, nothing interesting here.

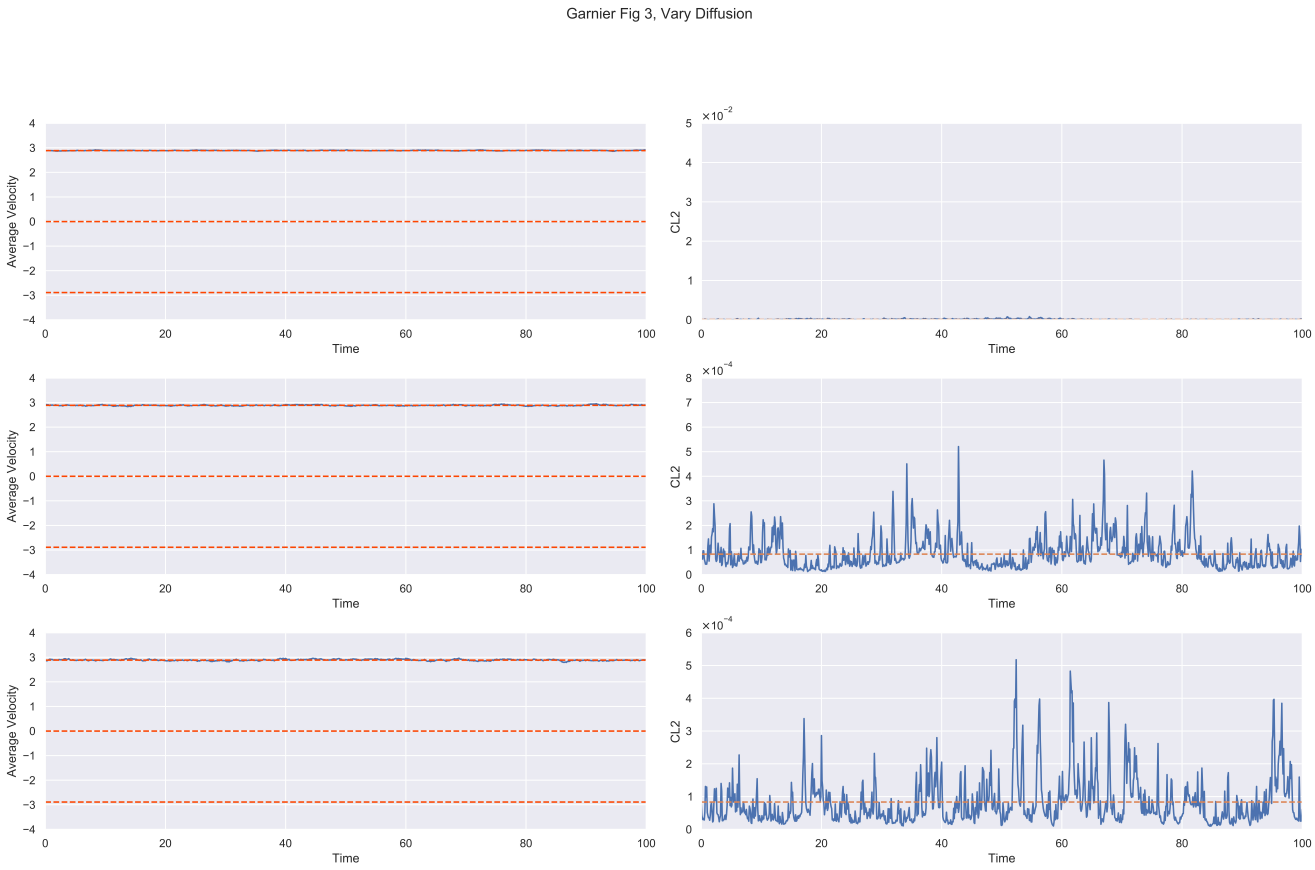

Figure 9: As in Garnier Figure 3 but scaled: Average velocity and CL2 discrepancy for $\sigma = 0.5^2/2$ (top), $\sigma = 1^2/2$ (middle) and $\sigma = 1.5^2/2$ (bottom). No clusters form! Perhaps check a noise even lower? Although $\sigma < 0.125$ is basically deterministic. Check deterministic?