Phys 580 Final Project: Smoothed Particle Hydrodynamics

Kenny Van, ID:1255162

December 7, 2015

Abstract

Smoothed particle hydrodynamics (SPH) is a well established but still evolving method of simulation that was initially made for astrophysical simulations. Initially created in 1977[3, 4], the code has continued to be developed for more complicated systems [9]. Here, we will give a basic overview of the theory of SPH and how I applied it to the two dimensional toy models [8]. We will compare results found using this code to those we expect to see analytically. We will also compare two common smoothing kernels to see if they produce drastically different results in our simulations. Finally, we will attempt a collision model of two bodies. We see that in all cases the code appears to produce reliable results and with it's relatively straightforward theory, further expansion to this code would be viable.

1 Introduction

Smoothed particle hydrodynamics (SPH) is a meshless particle based method for simulating fluid dynamics. Originally created to model complex stellar models [3, 4], the SPH model has been greatly extended since its inception to cover a wide variety of physical models[6, 9]. As stated before SPH is a meshless particle method, this means that the computation works independently of any grid. As such, unlike finite-difference methods, interactions within the volume between elements is represented entirely by forces between particles. The main result of the meshless nature of SPH is the ability to calculate all values in a coordinate system centered on each element. This change in coordinate system allows for a purely Lagrangian method of calculation. But in order to have elements interact, a smoothing kernel must be introduced to the force calculations. A major advantage of SPH is its conservative properties. SPH conserves both linear and angular momentum regardless of geometry [9]. In comparison to mesh based methods, SPH can easily simulate a mixture of materials by dividing the particles used into the appropriate properties.

In the following sections, I will discuss the base theory of SPH and my choices in implementation in python. Using my implementation, I will then use it to try to recreate toy star models [8]. Initial conditions set for our system were influenced by work done by Monaghan & Price (2004, 2006)[7, 8] as well as Braune & Lewiner (2009)[1]. The results will be presented and comparisons will be made with those found by Monaghan & Price (2006). In addition to the simple models initially prescribed by Monaghan & Price(2006), I will simulate a collision between two toy stars. A discussion on current and future work done in regards to SPH will be presented after the analysis.

2 Underlying Theory

As mentioned in the Introduction, SPH is a Lagrangian method of simulating fluids. This results in the entirety of the fluid being approximated by a collection of particle elements, each with the appropriate physical quantities. In order to properly use SPH, the equations describing these physical quantities must be defined. The Euler equation for a non-dispersive ideal fluid is:

$$\rho \frac{dv}{dt} = -\nabla P + f \tag{1}$$

here ρ is the density, v is the velocity, t is time, P is the pressure and f is any additional external force. In this work we will attempt to use simple physical relations wherever possible. With that in mind, we assume that all our models can be sufficiently approximated with a polytropic fluid. This results in the following relation between pressure and density:

$$P = k\rho^{1+\frac{1}{n}} \tag{2}$$

where k is a multiplicative constant and n is the polytropic index. Again, the key idea in SPH is to approximate a fluid as a collection of particles. The above equations do not allow for this, instead we will reform the equations with a smoothed interpolated version. Any smooth interpolating function can be described as [5]:

$$A(r) = \int A(r')W(r - r', h)dr' \tag{3}$$

where $A(r)$ is some function, $W(r, h)$ is the smoothing kernel r is the particle's position and h is the smoothing length scale. The kernel must satisfy four conditions:

1. Compactness:
$$W(\frac{r}{h}) = 0$$
When $\frac{r}{h} \geq 1$. In other words, particles outside a certain range h have no effect on each other.

2. Delta Function Convergence:
$$\lim_{h \to 0} W(r, h) = \delta(r)$$

As the smoothing length approaches zero, our kernel becomes the delta dirac function.

3. Unity Convergence:
$$\int W(r)dr = 1$$

4. Smooth and continuous.

5. Positive and invariant under parity

Many different kernels satisfy these conditions, we will use an arbitrary d-dimensional Gaussian kernel:

$$W(r,h) = (\frac{1}{h\sqrt{\pi}})^d e^{-(\frac{||r||^2}{h})} \tag{4}$$

The Gaussian was chosen as it produces results most similar to those seen physically [3]. Another very common kernel is the cubic spline approximation in two dimensions [10]:

$$W(r,h) = \frac{10}{7\pi h^2} \begin{cases} \frac{1}{4}(2-\frac{r}{j})^3 - (1-\frac{r}{h})^3 & 0 \leq \frac{r}{h} < 1 \\ \frac{1}{4}(2-\frac{r}{j})^3 & 1 \leq \frac{r}{h} < 2 \\ 0 & \frac{r}{h} \geq 2 \end{cases} \tag{5}$$

I will compare the two splines to determine how large of a difference there is in choice of kernel for the toy stars.

Equation (3) must then be transformed into a discretized form for N particles. This numeric form will then be applied to the physical values in the system:

$$A(r) = \sum_i^N A(r_i)W(r_i,h)\Delta r \tag{6}$$

The derivative of this function is required to calculate the acceleration

$$\nabla A(r) = \sum_i^N A(r_i)\nabla W(r_i,h)\Delta r \tag{7}$$

We can now apply equation (7) to a rearranged equation (1) and (2):

$$\frac{dv}{dt} = -\frac{\nabla P}{\rho} + \frac{f}{\rho} \tag{8}$$

and

$$\frac{\nabla P}{\rho} = \nabla(\frac{P}{\rho}) + \frac{P}{\rho^2}\nabla\rho \tag{9}$$

Using equations (8), (9) and (10) we get the key equations for SPH, acceleration can be defined as:

$$\frac{dv_i}{dt} = -\sum_{i,j\neq i}^{N} m_i(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2})\nabla W(r_i - r_j, h) + f_i \tag{10}$$

with density as:

$$\rho_i = \sum_{i,j\neq i}^{N} m_i W(r_i - r_j, h) \tag{11}$$

The above equations are the main equations used to find the time evolution of the fluid in SPH. It is also important to note that the term f_i in equation (10) represents any external forces applied to the fluid. This allows for this general form of the equation to be expanded to incorporate more complex effects and cover more advanced physical systems. An important aspect of the two above equations is that the sum is over two indices i and j in two dimensions. The j index also only applies when $i \neq j$ to prevent the effects between particles being calculated twice.

2.1 Integration Method

Due to SPH approximating a continuous fluid as a collection of discrete elements it is possible for many time integration methods to be used. We will follow the method used by Monaghan (2005) [6] where he uses what is known as a Verlet second order integrator. Another way this integration method is defined is through the name "leapfrog integrator" which adequately describes what is done. This integration method is preferred over some more robust methods due to its simplicity, it is explicit, second order and conserves energy. This integration method is as follows:

$$v_{t+\Delta t/2} = v_{t-\Delta t/2} + \frac{1}{2}\Delta t a_t \tag{12}$$

$$r_{t+\Delta t/2} = r_t + \Delta t v_{t+\Delta t/2} \tag{13}$$

Where the values are denoted as t, $t+\Delta t/2$ and $t-\Delta t/2$ are at the timestep, half a step after and half a timestep later respectively. Δt represents the timestep size used in the simulations. We see that with this integration method to properly use this integration method the information at timestep $-1/2$ must be found. This can be done using the Euler equations to find $v_{t-1/2}$. Another approximation that we make with our code is the assumption that changes in velocity are small with each timestep. This is done so that we may decouple the velocity dependence in acceleration that would arise from equation (12). With this approximation, we can rewrite velocity as an equation that depends only on its previous and next point:

$$v_{t+\Delta t} = \frac{v_{t-\Delta t/2} + v_{t+\Delta t/2}}{2} \tag{14}$$

2.2 Smoothing Scale Length

The smoothing scale length is a key initial value that must be defined in SPH. The scale length h represents how far each element has an effect. If the value is too small, there may be no interaction between particles if the value is too large, the simulation may be unstable. A method of defining h was given in the initial creation of SPH [3]. In the paper it is recommended that h is chosen such that it is of the same order of magnitude as :

$$\sqrt{(\bar{r^2} - \bar{r}^2)} \tag{15}$$

Here $\bar{r}$ represents the average position of the particle. In our simulations we will be using a prescription of h as defined:

$$h = \frac{\Delta t}{\sqrt{\frac{N}{1000}}} \tag{16}$$

This is chosen to ensure that between timesteps Δt, the smoothing length h is large enough to cause interactions between elements. N represents the number of particles used in the simulation. So the division by $\sqrt{\frac{N}{1000}}$ is used to ensure only a fraction of the elements will be included. It is important to note that determining an efficient smoothing length such that the kernel includes enough elements to be accurate, but not so many that the computation is highly time consuming is still an area of ongoing work [2].

3 Implementation and Tests

The SPH code was written in python 2.7.6 and all simulations were run locally on my personal laptop. A minor issue with using python for SPH is that the language is not ideal for numerical simulations. This is evident in my current implementation running entirely on a single core. The code itself is also not highly optimized, but with low particle count and few timesteps the simulations finished in a reasonable amount of time. The code was created to reproduce the simple results from Monaghan & Price 2006 [8]. As such the code is written in 2D and an external force term must be applied to the acceleration calculations. From the toy star models described the acceleration is given as such:

$$\frac{dv_i}{dt} = -\varepsilon v_i - \sum_{i,j\neq i}^{N} m_i\left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right)\nabla W(r_i - r_j, h) - g r_i \tag{17}$$

We see that this is a slightly modified version of the general acceleration equations with the addition of two external force terms. The g term is a simplified gravity term that forces the system around the origin. Without this term, we would expect the system to disperse similar to an ideal gas in a large volume. We have an additional dampening term in ε that reduces the overall effects of the base acceleration calculation. Without this dampening term, the system will likely never reach equilibrium.

In order to test the validity of our SPH code, we need an analytic solution to compare our results with. We expect that once the system reaches equilibrium $v = 0$. Using this we can solve equation 1:

$$0 = -\frac{\nabla P}{\rho} - gr \tag{18}$$

$$0 = -\frac{k(1+\frac{1}{n})}{\rho}\rho^{1/n}\nabla\rho - gr \tag{19}$$

Which can solved analytically to give the relation for density:

$$\rho(r) = (\frac{g}{4k(n+1)}(R^2 - r^2))^n \tag{20}$$

Where again, g is a simplified gravity constant, k is a multiplicative value used in finding pressure, n is the polytropic index, R is the radius of the star while r is the radial position. Using the fact that the density equates to M/V, or mass over volume. In our case as the equation is two dimensional we get the equation:

$$M = \int_0^R d^2 r \rho(r) \tag{21}$$

Using this with equation (20) we can solve for g:

$$g = 2k\pi^{-1/n}\frac{(\frac{M(1+n)}{R^2})^{1+\frac{1}{n}}}{M} \tag{22}$$

Using this we will run multiple simulations and if properly implemented the solution should match with that found in equation (20) after sufficient timesteps.

4 Results

I performed 7 simulations [1] of the toy star models in arbitrary dimensionless units. To test the claims made in previous sections, simulations were made with appropriate initial conditions. All of the simulations parameters are listed in table 1 with plots of the results shown below. All of the initial positions of the star are generated using a random seed from python. The particles are randomly distributed within a radius which is initially defined as shown in table 1.

In simulations 1 to 3, I test the effects of the g and ε terms in our acceleration equation. We see that the gravity and damping terms act the way we expect. The next step in our simulations was to determine if there is a noticeable difference between cubic spline fits and a Gaussian fit. With a simple model such

[1] Videos of all results are available for download at `www.ualberta.ca/~kvan/Gifs.tar.gz`

Table 1: Table of values for simulations, simulation 6 and 7 are collisions where the values in brackets represent the particles per star. In the gravity column the g value corresponds to equation (22)

	Particle Count	Radius	Timesteps	Damping	Gravity	Kernel
1	100	0.75	200	0	0	Gaussian
2	100	0.75	200	1	0	Gaussian
3	100	0.75	200	0	g	Gaussian
4	100	0.75	200	1	g	Gaussian
5	100	0.75	200	1	g	Cubic
6	800, (400, 400)	0.25	600	1	g	Gaussian
7	500, (400, 100)	0.75	600	6	g	Gaussian

as the one used here, it would be expected that the choice in kernel would not cause large variations. In figure 3, we see that this prediction is true as the two kernels produce similar final density profiles.

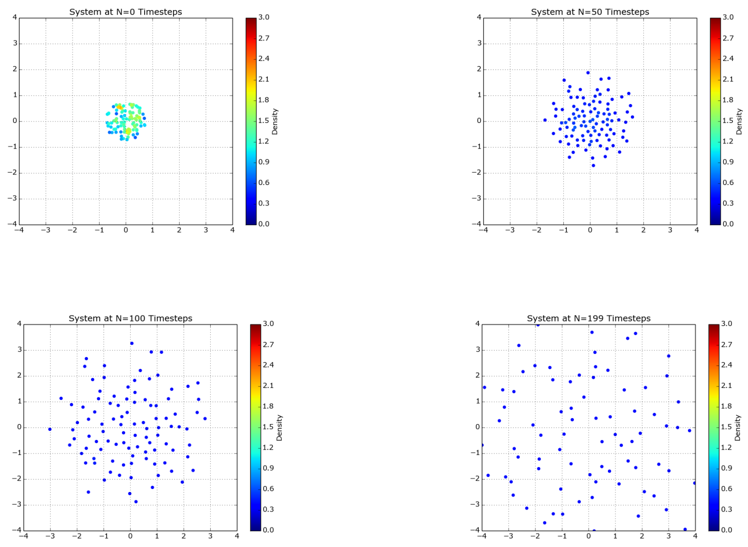

Figure 1: Simulation 1 at various iterations, with no gravity and no damping we see the particles disperse.

Seeing that the two kernels produce similar final results, one would expect the choice of kernel here to be arbitrary. This is not true as the cubic spline has a distinct advantage over the Gaussian. With the way the cubic spline is defined, each particle only affects elements within $0 \leq r \leq 2h$. This can significantly reduce computation time over using the Gaussian kernel where all elements affect one another. However, the trade off for longer computation time using the Gaussian is that it generally reproduces results more similar to

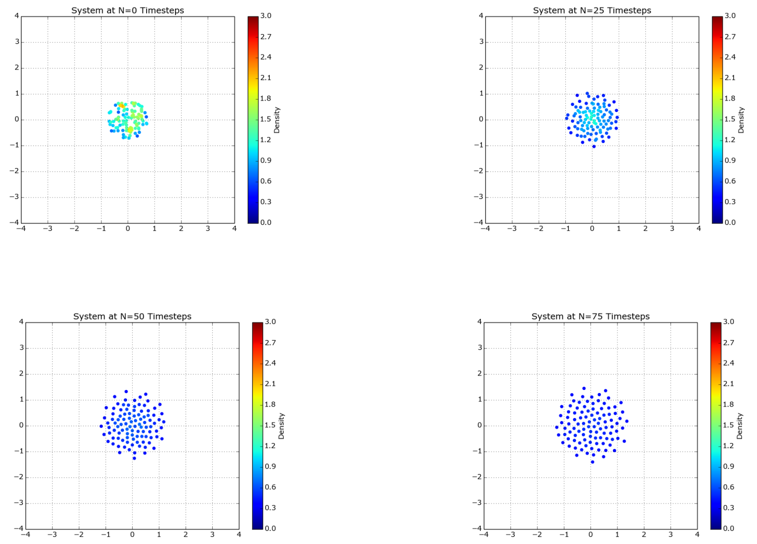

Figure 2: Simulation 2 at various iterations, with damping but no gravity the particles move until they no longer interact.

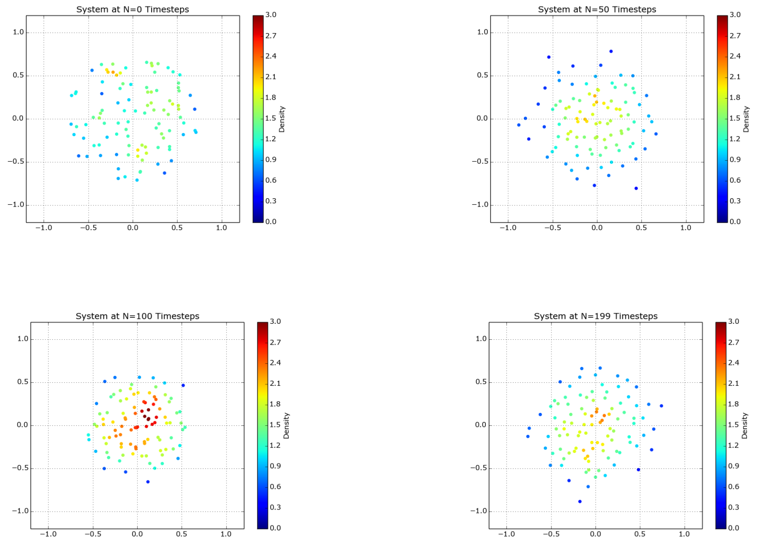

Figure 3: Simulation 3 at various iterations, with gravity but no damping, the particles do not appear to be closer to equilibrium after 200 steps.

physical results. So while the cubic spline would be more time efficient, all further simulations will be done with the Gaussian.

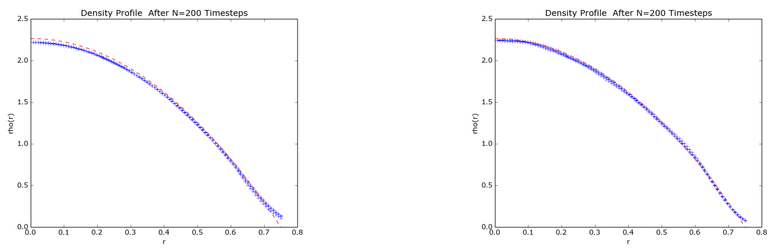

Figure 4: The basic toy star model using two different kernels. The left figure is using the Gaussian kernel while the right figure uses the cubic spline. In both plots the blue is the results produced by the code while the red line is the analytic answer.

As expected from figure 4, the overall simulations of our toy star model regardless of spline produces the same general result. This shows that our SPH produces similar results that correctly predict those found analytically using the toy star model. One major point of difference seen between our solution and the analytic solution is the values found near the edge of our toy star. This is likely caused by the simulation not having run for a sufficient amount of timesteps. Looking at figures 5 and 6 it appears as though our models become a perfect circle, but looking at figure 4 this is likely not the case. As a result of our approximation of fluids as a collection of elements, it is entirely possible that we have elements that are pushed outside the prescribed radius. This is the most probable reason to why the density at the edge of the model is not that seen in the analytic solution.

Additional simulations were run that fell outside the scope of those described in the toy star paper [8]. I ran two additional cases to test the effects of collision in this SPH code. Surprisingly, in simulation 6 we see what appears to be mixing between the two initial stars. This is likely caused by the large acceleration the stars experience, tearing them apart before being recombined. As this tearing and recombination occurs multiple times there may be an artificial mixing that occurs. To ensure there is no unknown mixing hidden in the code model 7 was run. In this case we have a relatively small number of particles fall onto our larger star. Here I color code the particles to clearly show any mixing effect. As expected there is no observed mixing between the two particles, instead the particles that were initially in the right remained on that portion of the larger star.

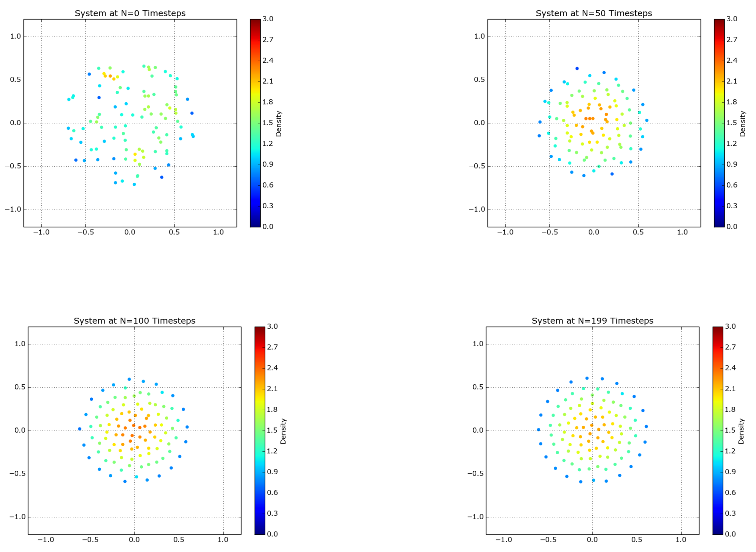

Figure 5: Simulation 4 at various iterations, this simulation is using the Gaussian kernel.

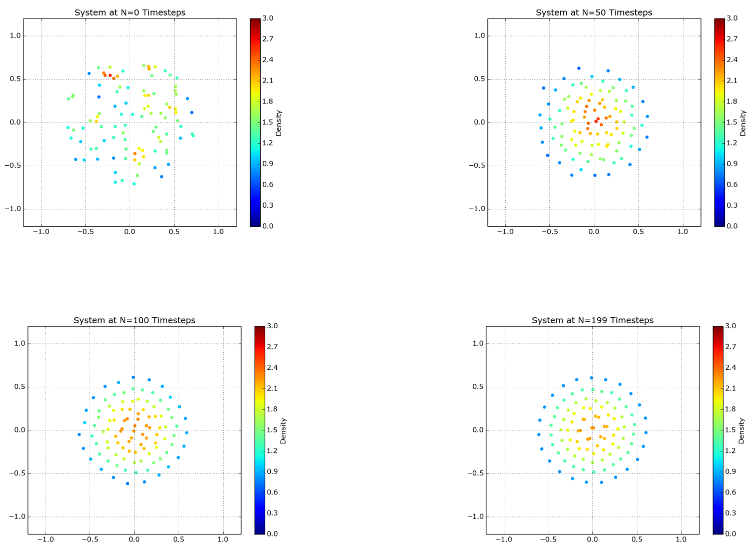

Figure 6: Simulation 5 at various iterations, this simulation uses the cubic spline kernel.

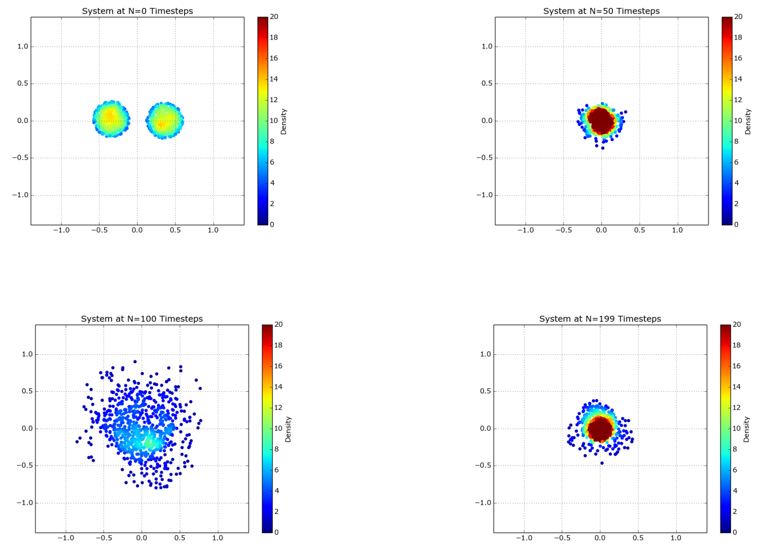

Figure 7: Simulation 6 at various iterations, the two stars are of equal particles and size and are given a small initial kick towards each other.

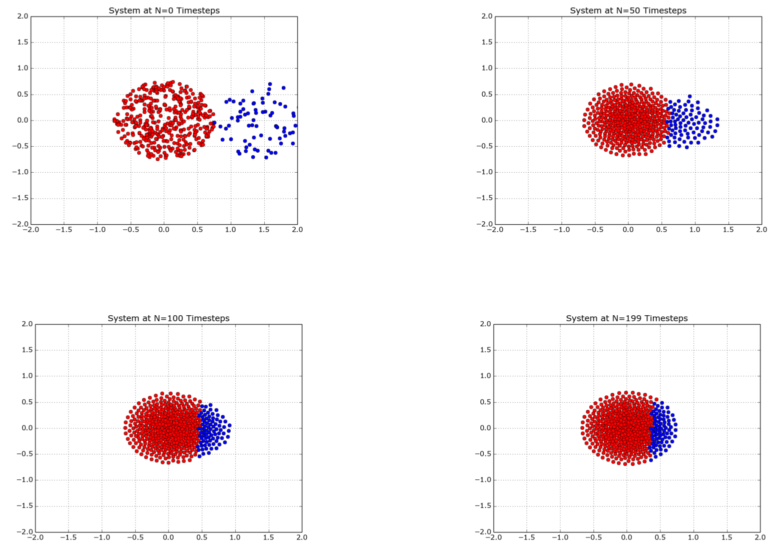

Figure 8: Simulation 7 at various iterations, this simulation clearly shows that there is no internal mixing.

5 Discussion

In this project we implemented our own simple two dimensional SPH code. In order to test its validity, we compare our results to toy star models[8]. The results of our code reliably match analytic solutions found given by these toy stars. In addition to these results, there were no unexpected physical phenomena arising from our code except when collisions were being tested where the possibility of mixing may have occurred, however this was a result of our toy models breaking into smaller pieces and being reformed with each other. Shown explicitly in a subsequent model, we see that there is no internal mixing that may have arisen from our code.

5.1 Extensions

Due to the relatively simple theory behind SPH, further extension of our code is entirely possible. What may be the most obvious improvement to the code that could be made is the expansion of the models to a third dimension. With how general the code was written, this expansion should be relatively simple to do. Beyond this, most logical additions to the code would require considerable more time and theory. As explained previously, the choice of smoothing length is a key choice to be made in SPH. Having too small of a h value results in inaccurate results, while having too large of an h value causes extended computation. What has been done to alleviate this problem is the concept of adaptive smoothing lengths. This change in h is such that the value is no longer initially defined as a length but instead, as minimum number of points that must be encompassed by our smoothing length. As a result of this, more densely populated portions can have small smoothing lengths reducing the calculations required. An inverse of this is that sparsely populated areas have large smoothing lengths to accommodate.

Outside of these two large changes in the code, moving from python to a code more suited for numerical simulations would be necessary. With our current implementation of SPH, simulation 6 took hours to complete. The scaling of computational resources required in SPH scales as N^2 where N is the number of particles. Doubling the amount of particles used would increase the amount of time required by a factor of four. To even approach more complex physical systems with this code, parallelization and further optimization must be done for this to be viable.

Beyond possible improvements I can make to my current implementation of SPH, research is more established SPH code continues. Recently SPH, code has been used to attempt to probe complex systems ranging from protostellar outflows to black hole accretion disks. Clearly as more complex systems arise and require analysis computational models will expand in an attempt to explain these systems.

Acknowledgements

I would like to thank Philip Mocz for his help through personal emails in this topic.

References

[1] L. Braune and T. Lewiner. An initiation to SPH. 2009.

[2] P. J. Cossins. Smoothed Particle Hydrodynamics. page 50, jul 2010.

[3] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[4] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013, dec 1977.

[5] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574, sep 1992.

[6] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, aug 2005.

[7] J. J. Monaghan and D. J. Price. Toy stars in one dimension. *Monthly Notices of the Royal Astronomical Society*, 350(4):1449–1456, jun 2004.

[8] J. J. Monaghan and D. J. Price. Toy Stars in two dimensions. *Monthly Notices of the Royal Astronomical Society*, 365(3):991–1006, jan 2006.

[9] D. J. Price. Smoothed Particle Hydrodynamics: Things I wish my mother taught me. page 10, nov 2011.

[10] D. J. Price. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794, feb 2012.