Smoothed particle hydrodynamics for hydrodynamic fluctuations

Michael Heinz

The Ohio State University

Abstract

These notes summarize my results during my JETSCAPE REU project at Wayne State University during the summer of 2019. I implement the smoothed particle hydrodynamics (SPH) method for solving relativistic hydrodynamic equations. This project provides a useful tool to study the hydrodynamic medium response to small local fluctuations in relativistic heavy-ion collisions. This open source code package is written in C++ with C++11 standard and has unit tests for each individual components.

1 The project

The general idea of this project was looking at modeling the evolution of hydrodynamic fluctuations. The first two weeks included a lot of reading papers on hydrodynamic fluctuations and understanding the kinetic equations that were derived in these papers. I learned a lot about four vectors and tensors, for example how $g^{\mu\nu}$ changes based on coordinates chosen and how this affects equations. In these papers [1, 2], the equations that we wanted to solve for the two-point correlation functions were partial differential equations. Partial differential equations can be solved numerically through a grid-method, but Professor Chun Shen wanted to test the smoothed particle hydrodynamics (SPH) method (commonly used in industry and astrophysics) to solving partial differential equations due to it's potential to be easily generalizable to different situations since SPH is a mesh-free method to solving the PDE. Thus, the main focus of my project became understanding and implementing SPH.

Over the next 8 weeks, I started from the ground up, programming an implementation of SPH in C++ with C++ 11 standard. This included learning a lot about C++ structures and how to manipulate them. I had to make a heirarchy of classes for greater convenience, such as a class for a single particle which held data about the particle and a class for the driver that controls all the particles. I also created a few base classes, such as a class for four vectors which overloaded operators for easier use as well as a few classes to implement a mesh that would help ease computation in finding neighboring particles for each particle. Additionally, I learned to use cmake to compile all of these classes together with the main program, allowing the code to be separated into many smaller parts to make it more readable and understandable.

As the code began to develop, I also learned from Prof. Shen many of the key strategies to developing "good" code, that is code that was clear, concise when possible, and well-documented. I also learned about unit tests, which check that basic methods within each class work as expected, allowing the user to make sure everything is running as expected still even after perhaps making big changes to code for computational efficiency. Finally, I learned about using git to commit changes to a repository such that there was a constant log that kept track of your changes to the code and would allow us to go back to an older version of the code if necessary. All of the code and unit tests are committed to the following repository which is public: `https://bitbucket.org/wayne_state_nuclear_theory/sph_solver/src/master/`. It was extremely rewarding to be able to write such code from the ground up with clear documentation that would hopefully make it easy for someone else to pick up when my time at the REU was finished.

On the other hand, I was also constantly working outside of the code to understand the methodology and implementation of SPH better. This included reading many papers [4, 5, 7–9] by some of the "godfathers" of SPH, including Professor Joseph Monaghan from Monash University. Through many discussions with Prof. Shen, we learned how to deal with various operators and covariant derivatives to write our equations in a form that would be solved correctly using the SPH method. A lot of what we found is discussed in the following sections about test cases I ran for the C++ code which I wrote.

In this report, we take $c = \hbar = 1$.

2 SPH basics

The basic idea of SPH is to discretize your problem domain into particles which carry mass m_i, density ρ_i, velocity v_i, and whichever other field values are necessary to a problem. These particles move according to their velocities which match the flow. These particles are essentially smoothed over a finite volume using a smoothing kernel W, and continuous field values are calculated by a weighted average of the field values of the particles. One advantage of this method is that spatial gradients don't have to be calculated using a finite difference method, but rather are calculated using the known gradient of the smoothing kernel W. This turns partial differential equations into a set of coupled ordinary differential equations, which is much easier to solve [5, 8].

2.1 Approximating a continuous field

The SPH approximations start from the identity

$$f(\boldsymbol{x}) = \int_V f(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}', \tag{2.1}$$

where f is any function defined on $V \subseteq \mathbb{R}^3$, $\delta(\boldsymbol{x})$ is the Dirac delta function, and $\boldsymbol{x}'$ is a dummy variable ranging over V [5, 7]. The delta function can be generalized to a smoothing kernel W with a smoothing length h, which has the following two properties [5, 7, 8]:

$$\int_V W(\boldsymbol{x}, h)d\boldsymbol{x}' = 1 \tag{2.2}$$

and

$$\lim_{h\to 0} W(\boldsymbol{x}, h) = \delta(\boldsymbol{x}). \tag{2.3}$$

If this smoothing kernel W is even, one can show that [5, 7]

$$f(\boldsymbol{x}) = \int_V f(\boldsymbol{x}')W(\boldsymbol{x} - \boldsymbol{x}', h)d\boldsymbol{x}' + \mathcal{O}(h^2). \tag{2.4}$$

To discretize the problem domain, one defines a series of particles in the problem domain with (potentially varying) mass $m_j = \rho(\boldsymbol{x}_j)d\boldsymbol{x}_j$. This replaces the infinitesimal volume $d\boldsymbol{x}_j$ by $\Delta V_j = m_j/\rho_j$, leading to the approximation

$$f(\boldsymbol{x}) \approx \sum_j \frac{m_j}{\rho_j} f_j \; W(\boldsymbol{x} - \boldsymbol{x}_j, h), \tag{2.5}$$

where $\boldsymbol{x}_j$, m_j, $\rho_j = \rho(\boldsymbol{x}_j)$, and $f_j = f(\boldsymbol{x}_j)$ are the position, mass, density, and value of function f of particle j respectively, and j runs over all particles [5, 7, 8]. We can apply this approximation at $\boldsymbol{x}_i$, the position of particle i, to get

$$f_i \approx \sum_j \frac{m_j}{\rho_j} f_j W_{ij}, \tag{2.6}$$

where $W_{ij} = W(\boldsymbol{x}_i - \boldsymbol{x}_j, h)$. One of the most important times this approximation is used is when updating the densities of the particles [5, 7, 8], as we have

$$\rho_i \approx \sum_j m_j W_{ij}. \tag{2.7}$$

In our case, we took the smoothing kernel W to be the Gaussian,

$$W(\boldsymbol{x}, h) = \begin{cases} \frac{1}{(\sqrt{2\pi}h)^d} e^{-|\boldsymbol{x}|^2/(2h^2)} & |\boldsymbol{x}| \leq 5h \\ 0 & |\boldsymbol{x}| > 5h, \end{cases} \tag{2.8}$$

where $d = 1, 2$, or 3 is the dimension of the problem. We cut the Gaussian off at $5h$ away from the center to increase numerical efficiency. This eases computation as if $|\boldsymbol{x} - \boldsymbol{x}_j| > 5h$, particle j doesn't contribute to the sum in equation (2.5).

2.2 Approximating spatial derivatives

One can similarly get an approximation for the spatial derivative [5, 8],

$$\nabla f(\boldsymbol{x}) \approx \sum_j \frac{m_j}{\rho_j} f_j \ \nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h), \tag{2.9}$$

as the gradient with respect to $\boldsymbol{x}$ passes through to the smoothing kernel W. Applied this approximation to particle i at position $\boldsymbol{x}_i$, we get

$$\nabla_i f_i \approx \sum_j \frac{m_j}{\rho_j} f_j \nabla_i W_{ij}, \tag{2.10}$$

where

$$\nabla_i W_{ij} = \frac{\partial W_{ij}}{\partial x_i}\hat{x} + \frac{\partial W_{ij}}{\partial y_i}\hat{y} + \frac{\partial W_{ij}}{\partial z_i}\hat{z}. \tag{2.11}$$

However, one can construct other estimators for the spatial derivative that are preferred for various reasons. For example, since $\nabla 1 = 0$, one can use

$$\nabla f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) - f(\boldsymbol{x})\nabla 1 \tag{2.12}$$

to get the approximation

$$\nabla f(\boldsymbol{x}) \approx \sum_j \frac{m_j}{\rho_j} f_j \nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h) - f(\boldsymbol{x}) \sum_j \frac{m_j}{\rho_j} \nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h) \tag{2.13}$$

$$= \sum \frac{m_j}{\rho_j}(f_j - f(\boldsymbol{x}))\nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h), \tag{2.14}$$

which has the advantage that it vanishes identically for a constant function f [5].

Similarly, the following two identities are often used to improve the gradient approximation,

$$\nabla f = \frac{1}{\rho}\Big[\nabla(f\rho) - f\nabla\rho\Big] \tag{2.15}$$

$$\nabla f = \rho\Big[\nabla\Big(\frac{f}{\rho}\Big) + \frac{f}{\rho^2}\nabla\rho\Big]. \tag{2.16}$$

From these, we get the following approximations,

$$\nabla f(\boldsymbol{x}) \approx \frac{1}{\rho(\boldsymbol{x})} \sum m_j (f_j - f(\boldsymbol{x}))\nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h) \tag{2.17}$$

$$\nabla f(\boldsymbol{x}) \approx \rho(\boldsymbol{x}) \sum m_j \left(\frac{f_j}{\rho_j^2} + \frac{f(\boldsymbol{x})}{\rho(\boldsymbol{x})^2}\right)\nabla W(\boldsymbol{x} - \boldsymbol{x}_j, h). \tag{2.18}$$

Again, the first identically vanishes for constant functions while the second is pairwise symmetric and is commonly used to ensure conservation of momentum when used with the pressure gradient [5, 7, 8]. From here on out, I will drop the approximation and write everything as equalities.

3 Relaxation equation test

The first equation of motion [10] that we are analyzing has the general form

$$u^\mu \partial_\mu \phi_a = -\Gamma(\phi_a - \phi_{eq}). \tag{3.1}$$

Here, Γ is a constant matrix, and ϕ_a is a vector of field values, which have an equilibrium value of ϕ_{eq}. For the SPH approximation, we write the equation as follows:

$$\gamma \frac{d\phi_a}{dt} = -\Gamma(\phi_a - \phi_{eq}), \tag{3.2}$$

where $\gamma = u^t$ and $d/dt = \partial_t + v^i\partial_i$ where i runs over x, y, and z. The reason for this is that the SPH method requires the equations to be written in terms of the full Lagrangian time derivative and spacial derivatives. As we will see, these equations will determine the change in the field value of a certain particle, and the $u^i\partial_i\phi_a$ terms are dealt with by the movement of the particles. Then, for particle i, we have the equation

$$\gamma\frac{d\phi_{a_i}}{dt} = -\Gamma(\phi_{a_i} - \phi_{eq}(\boldsymbol{x}_i)). \tag{3.3}$$

Until otherwise noted, ϕ_a is a unit-less scalar field (a vector with only one component), and hence we have that Γ is simply a constant in units fm^{-1}. Also, we restricted the problem to one dimension, having all particles on the x-axis with velocities only in the x-direction. Here, we are using the Euler's forward time difference to evolve the given ODEs. We discuss higher order numerical schemes in section 7.

3.1 Stationary fluid test

In our first test, we let the fluid be at rest and stay at rest, that is $u^x = u^y = u^z = 0$, so $\gamma = u^t = 1$. Hence, the equation of motion becomes

$$\gamma\partial_t\phi_a = -\Gamma(\phi_a - \phi_{eq}), \tag{3.4}$$

which has the analytic solution

$$\phi_a(t) = \phi_{eq} + (\phi_{a,0} - \phi_{eq})e^{-\Gamma t/\gamma}, \tag{3.5}$$

since γ is constant. We initialized 121 particles of mass $m_i = 1$ GeV uniformly in a box of width 6 fm, so $\rho = 20$ GeV/fm. Each particle is initialized to have $\phi_{a_i} = 2$ and $h = 0.1$ fm. We also let $\phi_{eq} = 0$ and $\Gamma = 1$ fm^{-1}.

Our numerical solution matched the expected analytic solution in the center of the box. However, there is a noticeable "bunny ear" edge effect when one calculates the field ϕ_a at the edge, as seen in figure 1. This is due to a drop in density at the edge. We have

$$\rho_i = \sum m_j W_{ij} \tag{3.6}$$

where $W_{ij} = W(\boldsymbol{x}_i - \boldsymbol{x}_j, h)$. At the edge, there are fewer neighboring particles (that is, particles within the support of W), so although mass is distributed uniformly, there are less significant terms in the sum when at the edge, leading to a drop-off in the density. This propagates through to when calculating ϕ_a near the edge. We have

$$\phi_a(\boldsymbol{x}) = \sum \frac{m_j}{\rho_j}\phi_{a_j}W(\boldsymbol{x} - \boldsymbol{x}_j, h). \tag{3.7}$$

Although each particle is initialized to have $\phi_{a_i} = 2$, at the very edge one suffers again from not having many neighboring particles, so the value of ϕ_a is lower. However, as one moves further in from the edge, the field value actually overshoots, becoming larger than 2 before finally settling back to 2 around $5h$ from the edge. This overshooting is because the particles near the edge have a lower value of ρ, leading them to contribute more in the sum due to the $1/\rho_j$ factor. One might ask why this isn't offset by the lower field values near the edge, and that is because the actual value of ϕ_{a_i} for particles near the edge isn't any lower: at all times, it is the same as for the particles in the middle since the evolution of ϕ_{a_i} has no interaction with other particles with this equation of motion as there are no spatial gradients. For a discussion of a way to deal with the "bunny ear" edge effect, see section 5.

3.2 Test with constant velocity

We did a second test where the fluid has a constant velocity in the x-direction, that is $u^x = $ const and $u^y = u^z = 0$. Thus, the equation becomes

$$\gamma\partial_t\phi_a + u^x\partial_x\phi_a = -\Gamma(\phi_a - \phi_{eq}). \tag{3.8}$$

Here, the flux term $u^x\partial_x\phi_a = 0$ since the initial field is constant and the fluid flows with constant velocity. This still has the analytic solution

$$\phi_a(t) = \phi_{eq} + (\phi_{a,0} - \phi_{eq})e^{-\Gamma t/\gamma}, \tag{3.9}$$

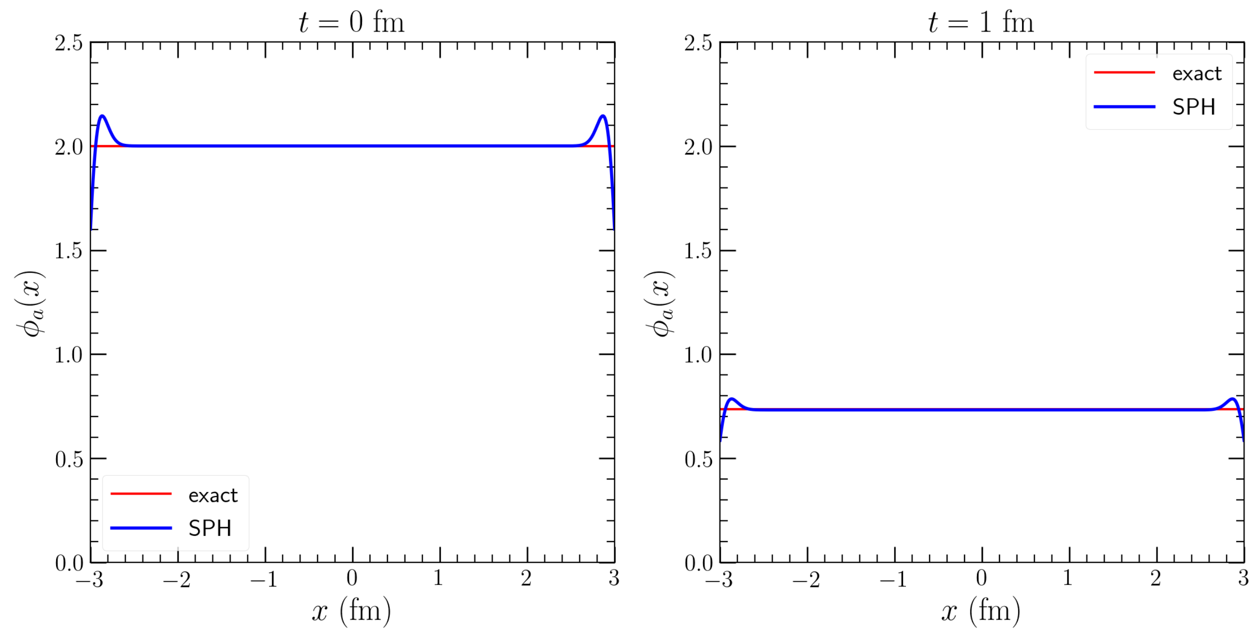

Figure 1: Plots of the SPH vs. exact solution to the relaxation equation with a stationary fluid at times $t = 0$ fm and $t = 1$ fm.

since γ is constant as the velocity is constant, so we can see that indeed $u^x \partial_x \phi_a = 0$. When the particles left the box on the right, we imposed periodic boundary conditions, resetting their position to re-enter on the left. Here, we let $v^x = \sqrt{3}/2$, so we have $\gamma = 2$, changing the decay rate in the analytic solution. As expected, we saw a factor of two decrease in the decay rate in the numerical solution.

4 Investigating size of h for Gaussian smoothing function W

A large part of the smoothed particle hydrodynamics is the smoothing function W. What this function does is that it takes the discretization of the fluid into finitely many particles and smooths it out to reproduce the properties of the field. This smoothing function has a given smoothing length h, and it is important to investigate this h thoroughly. To do this, we initialized a constant field in a one dimensional box of size 6 fm. We uniformly distributed 121 particles in the box, each with $m_i = 1$ GeV and $\phi_{a_i} = 2$, so $\rho = 20$ GeV/fm. We then varied h and recorded the initial values of $\phi_a(x)$ (up to 6 significant digits) for x values from $x = -3$ to $x = 3$ with step size $dx = 0.01$ Notice, the spacing between the particles is

$$\Delta x = \frac{6 \text{ fm}}{120} = 0.05 \text{ fm}.$$

We began with $h = 0.1$ fm, so that

$$\frac{h}{\Delta x} = 2, \tag{4.1}$$

and the field value was constant, $\phi_a(x) = 2$ (up to 6 significant digits). This means that the smoothing length h is large enough for the spacing between particles, smoothing out the particle discretization enough to reproduce the constant field. Increasing the size of h won't do us any good anymore, as it will only increase the distance that the "bunny ear" edge effect that we have seen previously reaches.

We then decreased the value of h, trying to find a point at which the smoothing length became too small and one could start to see the discretization of the continuous field. We found that at $h = 0.045$ fm, the field value was still constant. However, at $h = 0.0425$ fm the field was not constant anymore, having small dips in $\phi_a(x)$ at the x-values between the particles, hitting $\phi_a(x) \approx 2 - 10^{-5}$ as seen in figure 2. Thus, we found

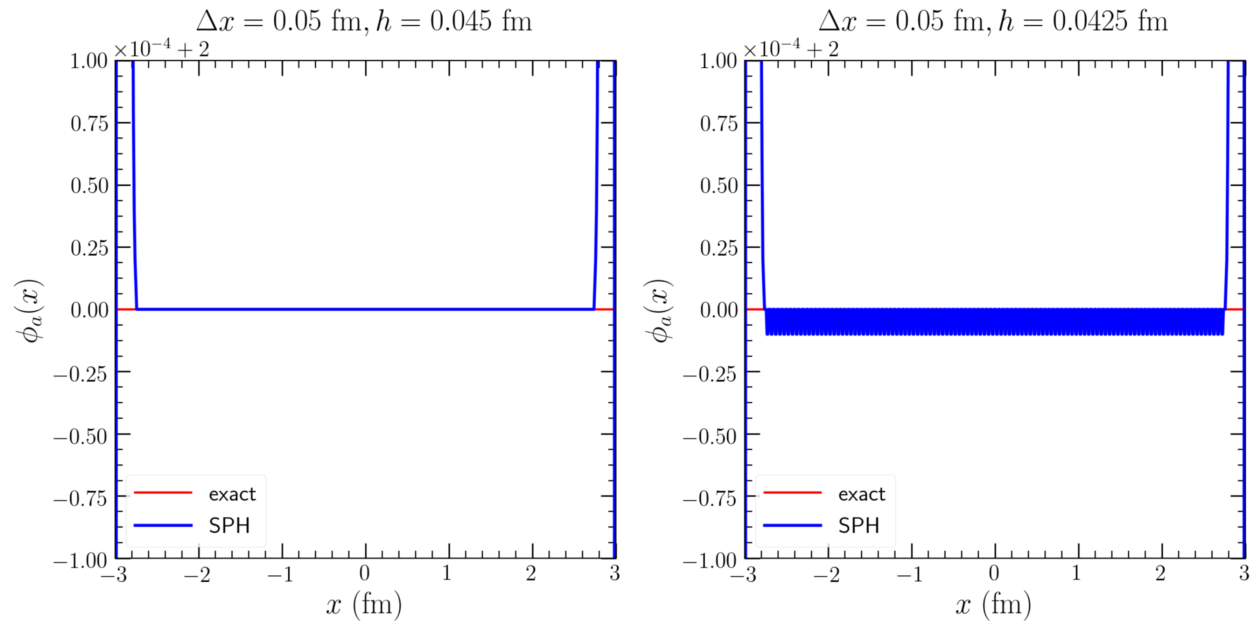

Figure 2: Plots of the initialized continuous constant field $\phi_a(x) = 2$ for 121 particles in a box of size 6 fm ($\Delta x = 0.05$ fm) at two different values of h: $h = 0.045$ fm and $h = 0.0425$ fm.

that to correctly represent a constant field in one dimension, one should have

$$\frac{h}{\Delta x} \geq \frac{0.045}{0.05} = 0.9, \tag{4.2}$$

as then the smoothing length is large enough for the distance between each particle. On the other hand, if

$$\frac{h}{\Delta x} \leq \frac{0.0425}{0.05} = 0.85, \tag{4.3}$$

the smoothing length is not large enough as one can see small variation in the value of $\phi_a(x)$.

To make sure that it was actually the ratio of h to Δx which was important, we ran the test again, this time initializing 61 particles in the box of width 6 fm, so that $\Delta x = 0.1$ fm. Once again, we found that for $h/\Delta x \geq 0.9$, the field was constant, while for $h/\Delta x \leq 0.85$, one could see some variation.

5 Constant shockwave test

Next, we did a shockwave test. This tests the equation

$$\gamma \frac{d\phi_a}{dt} = u^\mu \partial_\mu \phi_a = 0 \tag{5.1}$$

with the fluid moving at a constant velocity. We initialized 500 particles uniformly between $x = -4$ fm and $x = -3$ fm with $m_i = 1$ GeV and $h = 0.05$ fm, so $\rho = 500$ GeV/fm. The fluid had a constant velocity $v^x = \sqrt{3}/2$, so $\gamma = 2$. The particles were initialized with $\phi_{a_i} = 2$. Here, we let $\Gamma = 0$ to get rid of the source term. As we have seen before, the initial shape of ϕ_a had the"bunny ear" effect at the edges. However, the bunny ear shape stayed exact as the shockwave moved, as the values of ϕ_{a_i} for each particle didn't change since $\frac{d\phi_a}{dt} = 0$, as seen in figure 3. This seems to be an advantage to the usual grid numerical solutions of the shockwave, as they have to deal with a gradient of the steep discontinuity due to the flux term.

One attempt to fix the "bunny ear" issue is to use a different method for computing the density [7]:

$$\rho_i = \frac{\sum m_j W_{ij}}{\sum \frac{m_j}{\rho_j} W_{ij}}. \tag{5.2}$$

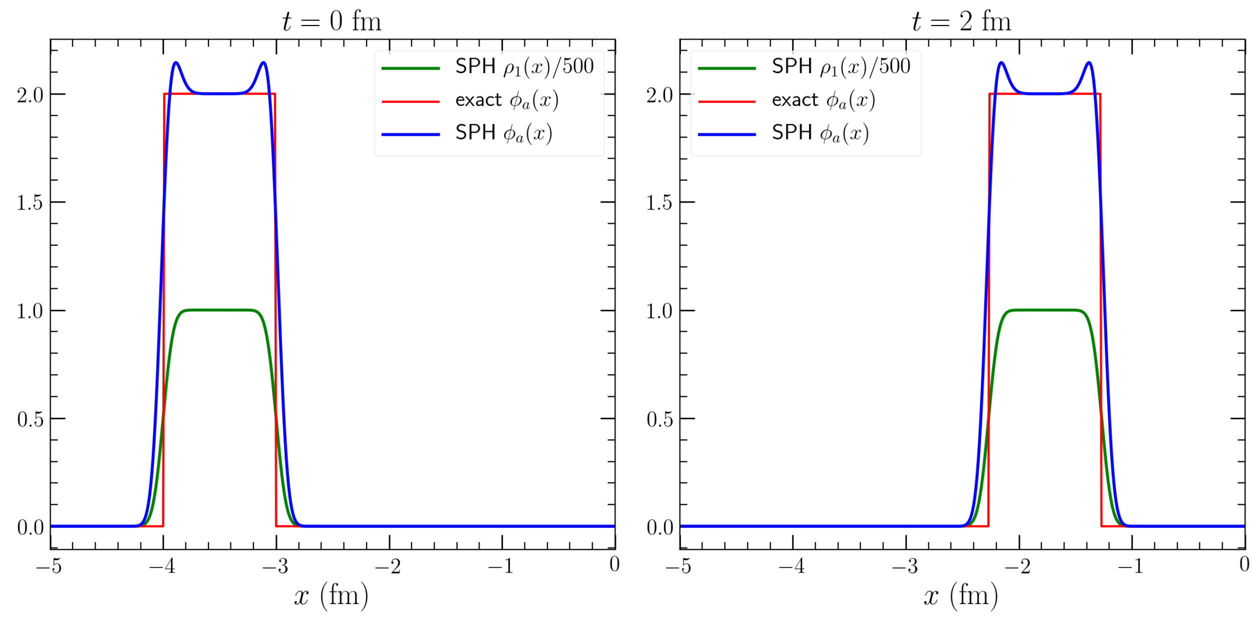

Figure 3: Plots of the SPH vs. exact solution of $\phi_a(x)$ for the shockwave equation at times $t = 0$ and $t = 2$. Also plotted is $\rho_1(x)/500$ solved according to equation (3.6). Notice the "bunny ear" edge effect in the SPH solution for $\phi_a(x)$.

This method has a normalization factor in the denominator. When making a fluid with constant density and evenly spaced particles, this avoids the drop-off in density along the edge, which gets rid of the"bunny ear" effect. However, it also creates a non-zero fantasy density when you compute the density where there are no particles, which is undesirable (see figure 4).

6 Neighboring particles list

Up until this point, to find the neighboring particles for a single particle i, we loop over all particles j and see if they are within a certain distance to particle i. To do this for each particle i is $O(N^2)$ time. We call this method the"all-pair search". This becomes really costly when we get to a large number of particles. Hence, we have implemented another method to finding the neighboring particles. This method utilizes a mesh that is placed overtop the problem domain with spacing κh, which is the radius of the support of the smoothing function W (it is the maximum distance between neighboring particles). First, we loop through the list of particles and add it to the corresponding cell which it is in (the cell has a list of particles belonging to it, which we call the linked list). Then, for each particle i, we simply have to loop over particles j in its cell and neighboring cells to find its neighboring particles list. This reduces the runtime to $O(Nlog(N))$ as h becomes small (so that the number of neighboring particles is small compared to the total number of particles). We call this method the "linked-list search" [7].

To demonstrate this, we ran a test in 2-dimensions. We initialize $101^2 = 10201$ particles distributed evenly in a 10 fm$\times$10 fm box. Each particle is initialized with ϕ_{a_i}. We run with a zero source term, so that their ϕ_a values don't change. Each particle is at rest, and thus should have the same number of neighboring particles at each time step. When using the linked-list search, the program ran 300 time steps in 74.65 seconds. When using the all-pair search, the program ran 300 time steps in 640.01 seconds. Here, the linked-list method was faster by a factor of 8.

Of course, this factor depends on the number of total particles, as well as the number of particles that are contained in each of the cells in the mesh (that is, the number of neighboring particles each particle should have). For example, if all the particles are contained in two cells of the mesh (so that each particle is neighbors with almost every other particle), the linked-list method will give no significant improvement.

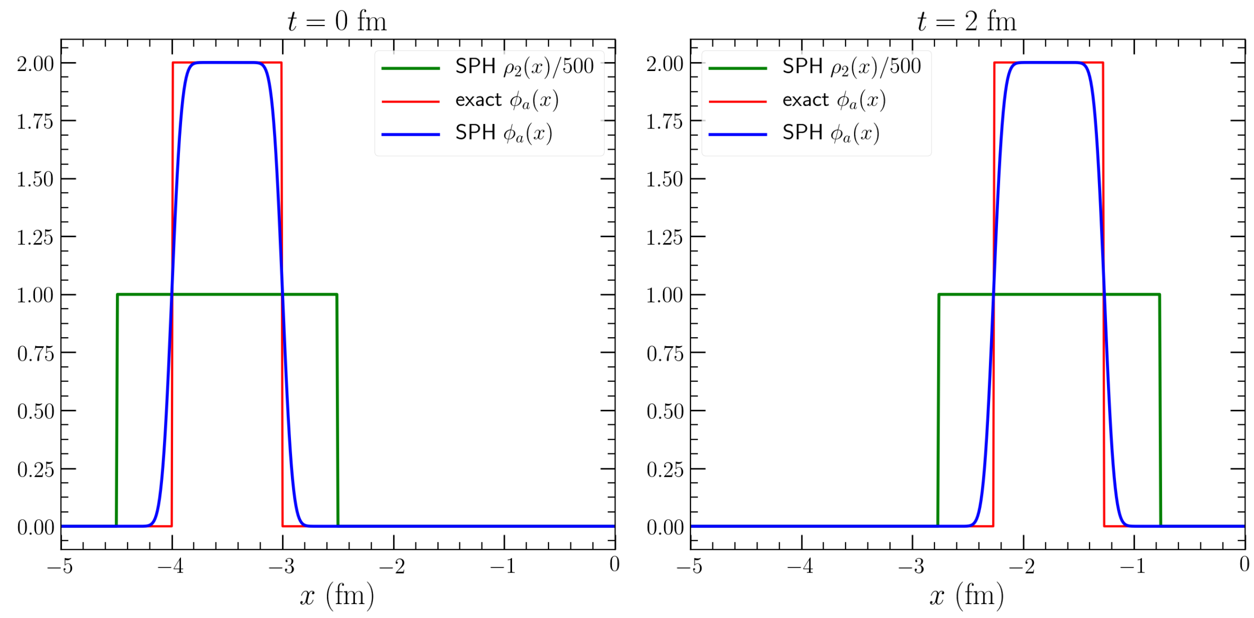

Figure 4: Plots of the SPH vs. exact solution of $\phi_a(x)$ for the shockwave equation at times $t = 0$ and $t = 2$. Also plotted is $\rho_2(x)/500$ solved according to equation (5.2). Notice that this solves the "bunny ear" edge effect in the SPH solution for $\phi_a(x)$, but also introduces a fantasy density where there are no particles.

7 Delta function diffusion test

The next test we ran was a delta function diffusion test. That is, we tested the equation

$$\gamma \frac{d\phi_a}{dt} = u^\mu \partial_\mu \phi_a = D\nabla^2 \phi_a \tag{7.1}$$

We ran it in both one dimension and two dimensions. We initialized ϕ_a to only be non-zero at the particle placed at the origin, to mimic a delta function. Of course, due to the smoothing function, the represented continuous ϕ_a field was a Gaussian,

$$\phi_a(\boldsymbol{x}) = \frac{\phi_{a_0}}{(\sqrt{2\pi}h)^d \rho} e^{-|\boldsymbol{x}|^2/(2h^2)}, \tag{7.2}$$

where ρ is the density of the initialized particles, d is the dimension of the problem, and ϕ_{a_0} is the value of ϕ_a for the particle at the origin. We let $\phi_{a_0} = 10$. This equation can also be solved exactly for this initial Gaussian.

We used three different SPH schemes to calculate the diffusion term. The three schemes were as follows:

$$D\nabla_i^2 \phi_{a_i} = D \sum \frac{m_j}{\rho_j} \phi_{a_j} \nabla_i^2 W_{ij}, \tag{7.3}$$

$$D\nabla^2 \phi_{a_i} = D \sum \frac{m_j}{\rho_j} (\phi_{a_j} - \phi_{a_i}) \nabla_i^2 W_{ij}, \tag{7.4}$$

and

$$D\nabla^2 \phi_{a_i} = D\rho_i \sum m_j (\frac{\phi_{a_j}}{\rho_j^2} + \frac{\phi_{a_i}}{\rho_i^2}) \nabla_i^2 W_{ij}, \tag{7.5}$$

where

$$\nabla_i^2 W_{ij} = \frac{\partial^2 W_{ij}}{\partial x_i^2} + \frac{\partial^2 W_{ij}}{\partial y_i^2} + \frac{\partial^2 W_{ij}}{\partial z_i^2} \tag{7.6}$$

in three dimensions, and was adjusted accordingly in one and two dimensions. We will refer to these three schemes as the basic, antisymmetrized, and symmetrized schemes respectively.

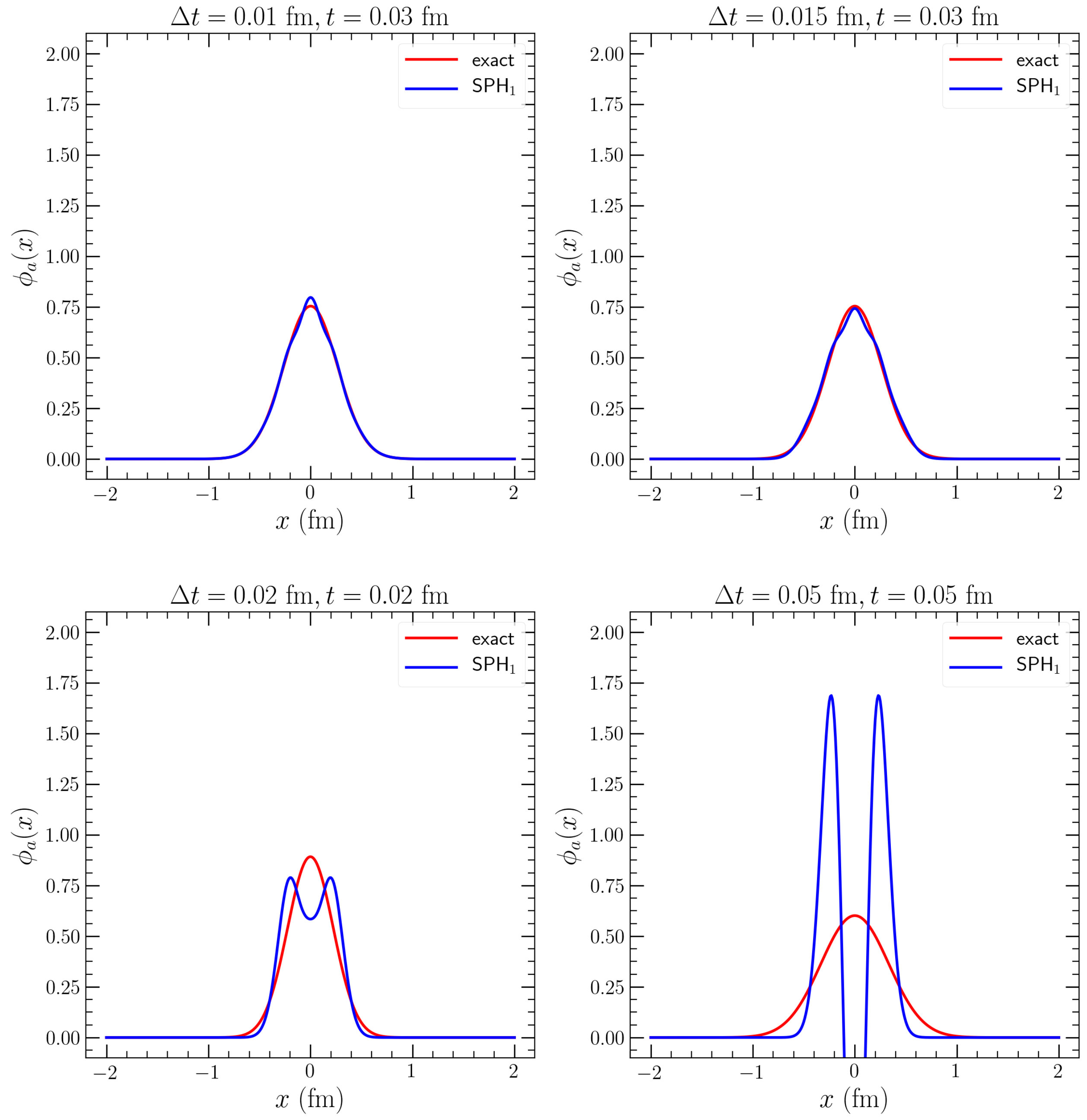

Figure 5: Plots of the SPH vs. exact solution of $\phi_a(x)$ for the diffusion equation in one dimension with diffusion constant $D = 1$ fm for an initialized Gaussian at small t for various values of Δt. Here, we used the basic SPH method for calculating the diffusion described in equation (7.3).

7.1 One dimension

In one dimension, we placed 121 particles with mass $m_i = 1$ GeV in a 6 fm width box, so that $\Delta x = 0.05$ fm and $\rho = 20$ GeV/fm. We also set $h = 0.1$ fm and $D = 1$ fm. We then varied Δt.

First, we used the basic scheme for calculating diffusion. At $\Delta t = 0.001$ fm, we found that in the first 100 time steps the SPH approximation trailed the exact solution, the SPH peak being larger than the exact peak as they both diffused. After around 0.2 fm, the SPH approximation caught up and was indistinguishable from the exact solution until the diffusion reached the edge of the box. At that point, the SPH approximation was actually lower than the exact solution, the exact solution rising at the edges as ϕ_a diffused, while the SPH solution trailed behind.

At $\Delta t = 0.01$ fm, the SPH solution again trailed the exact solution, as seen in figure 5. However, it described the exact solution much better than at $\Delta t = 0.001$ fm, matching to the exact solution pretty closely by 0.1 fm. This is actually most likely because the larger time step made up for an under-approximation in the Laplacian seen in the first test, keeping up with exact solution better. Again, once the diffusion reached the edge, the SPH solution trailed behind. We ran this one for longer, as there were fewer time steps, and after 2 fm, there was a distinct separation everywhere between the SPH and exact solution, the SPH solution having a lower value of ϕ_a everywhere as the edge effect propagated inwards.

At $\Delta t = 0.015$ fm, the SPH solution actually slightly leads the exact solution at the beginning, but matched the closest of any of the approximations so far, as seen in 5. This is again due to the larger time step. At longer times, the solutions match very closely.

At $\Delta t = 0.02$ fm, the SPH solution leads the exact solution by quite a bit. Here, the first time step is so big, that the middle of the Gaussian actually drops down further than the sides, making a small valley in the middle (see figure 5). This is then corrected in the next time step, although the SPH solution looks less Gaussian due to this oscillation. However, the oscillation steadies out by 0.1-0.15 fm, and at long times matches the other solutions.

At $\Delta t = 0.05$ fm, the overestimation in the first time step of the SPH solution actually makes the central ϕ_a go negative, leading to large oscillations in the next time steps which only become amplified making the solution unstable, as seen in figure 5.

These same traits can be seen with all three schemes for calculating the diffusion term. The solutions actually seem to match exactly in the first 0.3 fm. However, at long times, the antisymmetrized method begins to oscillate at the edges, leading to a solution that then blows up, as seen in figure 6. The basic and symmetrized schemes match at long times, and both end up having ϕ_a fields that a lower than the exact solution due to propagating edge effects.

7.1.1 Testing different differential schemes

Up to this point, we had always been using Euler's forward difference method to solve the differential equation. However, we wanted to test if we could make the step-size larger if we used a differential equation solver that had a higher order error in Δt. We tested both Heun's method (a generalized second-order Runge Kutta method) and RK4, the fourth-order Runge Kutta method. To test, we only used the basic SPH scheme in one dimension.

Both Heun's method and RK4 still lead to oscillations at $\Delta t = 0.05$ fm, so they did not lead to significant improvements in the step-size Δt we could use. Hence, it seems that once the solution is unstable, it might be unstable no matter which solver one uses. At $\Delta t = 0.02$ fm, both Heun's method and RK4 did match the exact solution much better than Euler's method, as they avoided the valley in the center caused by the larger step-size since Heun's method and RK4 break up the time step into smaller time steps to get better corrections. Heuns's method and RK4 actually trail the exact solution at the beginning, contrary to Euler's method. Out of the two better differential methods, RK4 matched the solution better, as expected.

7.1.2 Changing Δx

Next, we initialized 61 particles with mass $m_i = 1$ GeV in the 6 fm width box instead of 121 particles, so that $\Delta x = 0.1$ fm and $\rho = 10$ GeV/fm. We wanted to see if just changing Δx would change the value of $\Delta t_{\max}$ at which the solution begins to be unstable. However, it did not change (significantly). The solution

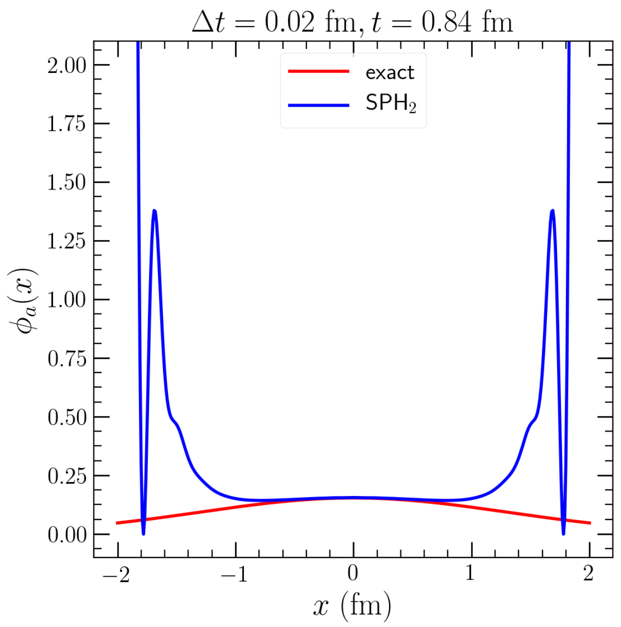

Figure 6: Plot of the SPH vs. exact solution of $\phi_a(x)$ for the diffusion equation in one dimension with diffusion constant $D = 1$ fm for an initialized Gaussian at $t = 0.84$ fm for $\Delta t = 0.02$ fm. Here, we used the antisymmetrized SPH method for calculating the diffusion described in equation (7.4).

was still stable at $\Delta t = 0.02$ fm with the small oscillations at the beginning as before, and the solution was unstable at $\Delta t = 0.05$ fm. Thus, we can conclude that Δx and $\Delta t_{\max}$ are independent in the SPH method.

7.2 Two dimensions

In two dimensions, we placed 3721 particles with mass $m_i = 1$ GeV in a 6 fm×6 fm box, so that $\Delta x = \Delta y = 0.1$ fm and $\rho = 10^2 = 100$ GeV/fm^2. We again set $h = 0.1$ fm, and we tested both $D = 1$ fm and $D = 0.1$ fm. We also varied Δt for both values of D. During the two dimensional tests, we exclusively used RK4 to solve the differential equation. In two dimensions, the simulation takes much longer to run, so we only ran at most 100 times steps.

We started with $D = 1$ fm and $\Delta t = 0.01$ fm. All three diffusion schemes solved the differential equation (seemingly) exactly the same first 100 time steps. They all trail the exact solution mirroring the behavior we saw in one dimension. When taking $\Delta t = 0.02$ fm, the SPH solutions still trail the exact solution at the beginning (as we saw in one dimension when using RK4), but the three SPH solutions still match exactly. Here, we could run to 2 fm using 200 time steps, and, as in one dimension, the antisymmetrized diffusion scheme starts oscillating at the edge once the diffusion reaches the edge, leading to the solution blowing up. This is not scene in either the basic or symmetrized diffusion schemes. Finally, when taking $\Delta t = 0.05$ fm, all three SPH solutions again have the large oscillations due to the large time step, and we see an unstable solution just as we did in one dimension.

We then took $D = 0.1$ fm and varied Δt to see if the maximum allowed Δt would change, and it did. We again started with $\Delta t = 0.01$ fm, where all three diffusion schemes matched exactly and trailed the exact solution, as we have seen before. This same behavior was seen with $\Delta t = 0.02, 0.05$, and 0.1 fm, and the solutions actually seemed to match exactly to those at $\Delta t = 0.01$ fm. At $\Delta t = 0.2$ fm, the solution was still stable and still trailed the exact solution, but it differed from the solutions of the previous four values of Δt. At $\Delta t = 0.333$ fm, the solution had small oscillations at the beginning, but they steadied out over time as the SPH solution began to match the exact solution. Finally, at $\Delta t = 0.5$ fm, we saw the unstable solution, as

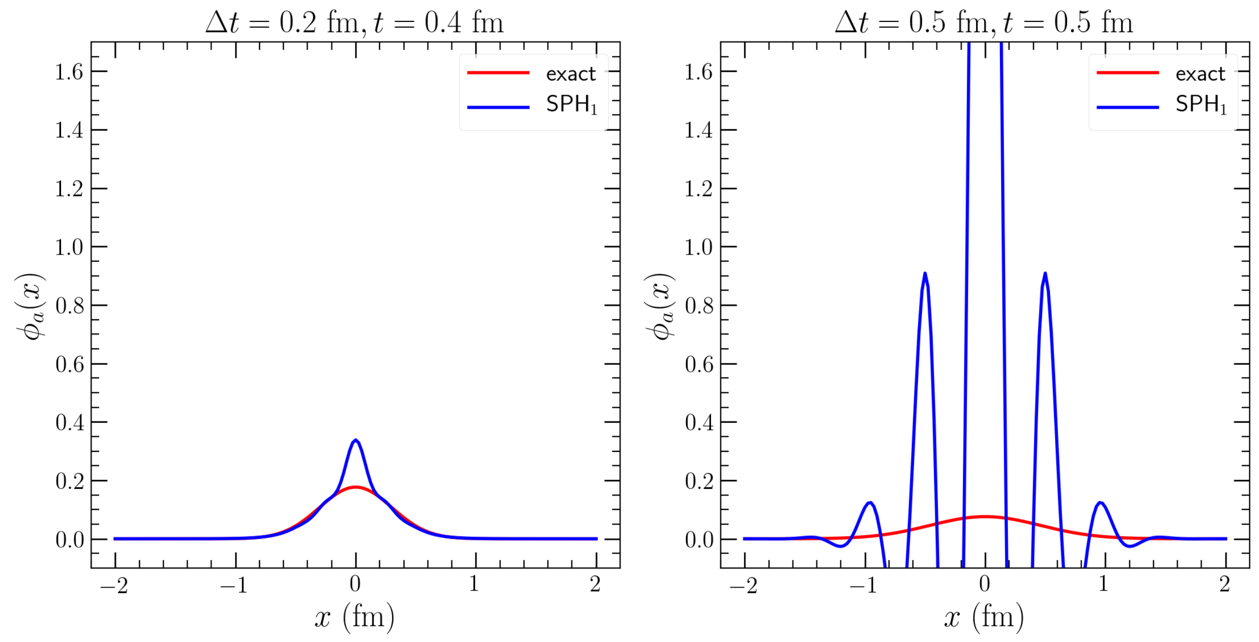

Figure 7: Plots of the SPH vs. exact solution of $\phi_a(x)$ for the diffusion equation in two dimensions with diffusion constant $D = 0.1$ fm for an initialized Gaussian at small t for two values of Δt. Here, we used the basic SPH method for calculating the diffusion described in equation (7.3).

seen in figure 7. The $\Delta t_{\max}$ is a factor of 10 larger from when $D = 1$ fm, so we conclude that

$$\Delta t_{\max} \sim \frac{1}{D}. \tag{7.7}$$

8 Causal diffusion equation

The next test we ran was for the relativistic causal diffusion equation with a general flow velocity u^μ. The relativistic causal diffusion equation can be written in two first order equations as follows:

$$\partial_\mu j^\mu = \partial_\mu(nu^\mu + q^\mu) = 0 \tag{8.1a}$$

$$\gamma\frac{dq^\mu}{dt} = u^\mu \partial_\mu q^\mu = -\frac{1}{\tau}(q^\mu - D\nabla^\mu n). \tag{8.1b}$$

Here, $\nabla^\mu = \Delta^{\mu\nu}\partial_\nu = (g^{\mu\nu} - u^\mu u^\nu)\partial_\nu = \partial^\mu - u^\mu\gamma\frac{d}{dt}$. We have the following metric:

$$g^{\mu\nu} = \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}, \tag{8.2}$$

so $\partial^t = \partial_t$ and $\partial^i = -\partial_i$. Now, since we want to write everything in terms of only full Lagrangian time derivatives and spatial partial derivatives, we use the following two identities to rewrite the above equations:

$$\partial^t = \partial_t = \frac{d}{dt} - v^i\partial_i \tag{8.3a}$$

$$\nabla^i = \partial^i - u^\mu\gamma\frac{d}{dt} = -\partial_i - u^\mu\gamma\frac{d}{dt} \tag{8.3b}$$

$$\nabla^t = \partial^t - \gamma^2\frac{d}{dt} = \frac{d}{dt} - v^i\partial_i - \gamma^2\frac{d}{dt} = -(\gamma^2-1)\frac{d}{dt} - v^i\partial_i. \tag{8.3c}$$

From equation (8.1a), we get

$$\gamma\frac{d}{dt}n+\frac{d}{dt}q^t=-n\theta+v^i\partial_iq^t-\partial_iq^i. \tag{8.4}$$

From equation (8.1b) with $\mu=t$, we get

$$\frac{D}{\tau}(\gamma^2-1)\frac{d}{dt}n+\gamma\frac{d}{dt}q^t=-\frac{D}{\tau}v^i\partial_in-\frac{1}{\tau}q^t. \tag{8.5}$$

From equation (8.1b) with $\mu=i$, we get

$$\frac{\gamma Du^i}{\tau}\frac{d}{dt}n+\gamma\frac{d}{dt}q^i=-\frac{D}{\tau}\partial_in-\frac{1}{\tau}q^i. \tag{8.6}$$

We can write this as the following matrix equation:

$$\begin{pmatrix}\gamma & 1 & 0\\ \frac{D}{\tau}(\gamma^2-1) & \gamma & 0\\ \frac{\gamma Du^i}{\tau} & 0 & \gamma\end{pmatrix}\begin{pmatrix}\frac{d}{dt}n\\ \frac{d}{dt}q^t\\ \frac{d}{dt}q^i\end{pmatrix}=\begin{pmatrix}-\theta & v^i\partial_i & -\partial_i\\ -\frac{D}{\tau}v^i\partial_i & -\frac{1}{\tau} & 0\\ -\frac{D}{\tau}\partial_i & 0 & -\frac{1}{\tau}\end{pmatrix}\begin{pmatrix}n\\ q^t\\ q^i\end{pmatrix}. \tag{8.7}$$

The matrix on the left can then be inverted, leaving the five desired differential equations with full Lagrangian time derivatives.

9 Acknowledgments

First, I would like to thank JETSCAPE REU for funding this research project under the NSF grant ACI-1550300. I would also like to Prof. Chun Shen at Wayne State University for all his guidance throughout the project, as well as the entire Wayne State University physics department for their hospitality. Overall, the experience I have had over the last 10 weeks has been amazing. I learned a lot about computational research in physics and about the area of nuclear physics in general, and I hope that my contribution to the research project will help its development in the future as Prof. Shen continues to explore SPH as a way to solve the equations for these hydrodynamic fluctuations.

References

[1] Akamatsu,Y., Mazeliauskas, A., & Teaney, D.,: 2017, arXiv:1705.08199v1 [nucl-th]

[2] An, X., Başar, G., Stephanov, M., & Yee, H.: 2019, arXiv:1902.09517v2 [hep-th]

[3] Aziz, M. A. & Gavin, S.: 2004, arXiv:nucl-th/0404058v1

[4] Chow, E. & Monaghan, J. J.: 1997, J. Comp. Phys., 134, 296

[5] Cossins, P.: 2010, arXiv:1007.1245v2 [astro-ph.IM]

[6] Denicol, G. S, Gale, C., Jeon, S., Monnai, A., Schenke, B., & Shen, C.: 2018, arXiv:1804.10557v1 [nucl-th]

[7] Liu, G.R. & Liu, M. B.: 2003, *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, World Scientific

[8] Monaghan, J. J.: 1992, *ARA&A* **30**, 543

[9] Price. D.J.: 2010, arXiv:1012.1885v1 [astro-ph.IM]

[10] Stephanov, M. & Yin, Y.: 2017, arXiv:1712.10305v1 [nucl-th]