Chapter 1

Simulation Solution

1.1 Smoothed Particle Hydrodynamics

Since we model neutron stars as a fluid, in order to simulate the collision of a primordial black hole with a neutron star, we need to use computational fluid dynamics, more specifically, smoothed particle hydrodynamics. Smoothed particle hydrodynamics was initially developed in 1977 for simulating stars ***. A major assumption in the modelling of stars is that they are spherically symmetric, however, there are a lot of cases where this is not true. For example the collapse of a protostar will be very lumpy; or if the star has a large angular momentum, or strong magnetic fields the star will considerably deformed ***.

Smoothed particle hydrodynamics has since been extended more generally so it can be applied outside of astrophysics such as solid mechanics, and fluid dynamics. In essence smoothed particle hydrodynamics uses similar methods of N-body code – the continuous media is represented as a set of particles, and the equations are solved numerically. Mathematically, the particles represent the points at which the properties of the fluid are calculated, the rest can then be interpolated. Physically, the particles are simply a subset of all the par-

ticles in the system ***. One of the biggest advantages of smoothed particle hydrodynamics is that it is mesh-free, unlike most computational fluid dynamics methods. Therefore, it is very useful for interactions between different types of particles, and for free surface simulations.

Typically, smoothed particle hydrodynamics uses a nearest neighbour scheme for calculating the changes in a particles properties. This is done by using a cubic Hermite spline to interpolate properties within a radius of twice the smoothing length of a particular particle. A cubic Hermite spline is a set of cubic polynomials specified by its values, and the values of its first derivative at the boundaries of the interval.

1.1.1 PySPH

The simulations were conducted with PySPH ***, open source SPH code written in Python and compiled in Cython. This allows a majority of the code to be written in pure Python, but is then converted to high performance Cython code to run at speeds closer to C and FORTRAN. For further optimizations, PySPH can run in parallel with OpenMP and MPI to take advantage of multiple cores / threads. One of the main reasons for choosing PySPH is that it allows user defined classes, and equations which was needed to incorporate the primordial black hole.

1.2 Code

In order to simulate a primordial black hole collision with a neutron star we must first write a class to include the acceleration. The acceleration of course is just

$$\vec{a} = -\frac{1}{m_{part}} \vec{\nabla} \Phi.$$

Within the coordinates in PySPH

$$\vec{a} = \frac{-m}{(x^2 + s^2 + (y + v\tau)^2)^{3/2}} \begin{pmatrix} x \\ y + v\tau \end{pmatrix}$$

with $G = 1$, $\tau = t - t\_hit$, and s a softening length so the solution does not diverge at $\tau = 0$ at the origin. The mass of the particle is also omitted in the simulation because PySPH uses acceleration per unit mass. In the class function (Figure 1.1) the velocity is hard coded to be 1, and the parameter *t_hit* is used to offset the collision of the primordial black hole since the simulation starts at $t = 0$.

Now that the class function has been written, it can be called within the equations block of the main file, *blackhole.py*. Figure 1.2 shows the important sections of *blackhole.py*; first the acceleration equation is imported, and the values for the simulation are specified in the preamble. Then, the black hole can be added to the equations group in the main acceleration block. The included example *hydrostatic_tank.py* *** was used as the base for *blackhole.py*, it was the most useful starting point – it is a 2D tank filled with fluid, and includes the force of gravity.

The parameters of the simulation were mostly the same as what were used to generate the images of the analytic solution – everything set equal to unity – with the exceptions of *t_hit* = 200, and $s = 0.01$. The reason *t_hit* is so large is due how the particles are initially placed. They are simply placed on a grid with spacing dx; once the simulation starts gravity pulls the surface down attempting to equilibrate to a linear pressure gradient. However, since the fluid is not at equilibrium, the surface undergoes harmonic motion and is slowly dampened to equilibrium, and so the long *t_hit* is an attempt to dampen out a majority of the oscillations.

```
from pysph.sph.equation import Equation

class BlackHole2D(Equation):
    def __init__(self, dest, sources, soft=0.05, t_hit=5.0, M=1.0):
        self.soft = soft # softening length to not divide by zero
        self.t_hit = t_hit # time when the black hole crosses the origin
        self.M = M # mass of black hole
        super(BlackHole2D, self).__init__(dest, sources)

    def initialize(self, d_idx, d_au, d_av):
        d_au[d_idx] = 0.0
        d_av[d_idx] = 0.0

    # calculate the force due to the black hole
    def loop(self, d_x, d_y, d_idx, d_au, d_av, t):
        d_au[d_idx] += -self.M * d_x[d_idx] / pow((d_x[d_idx]**2 + self.soft**2 + (d_y[d_idx] + t - self.t_hit)**2),3.0/2.0)
        d_av[d_idx] += -self.M * (d_y[d_idx] + t - self.t_hit) / pow((d_x[d_idx]**2 + self.soft**2 + (d_y[d_idx] + t - self.t_hit)**2),3.0/2.0)
```

Figure 1.1: Class file for adding the acceleration due to the primordial black hole, *BlackHoleEquation.py*.

```
19 # Import the equations
20 from pysph.sph.equation import Group
21 from pysph.sph.BlackHoleEquation import BlackHole2D

40 # Domain and reference values
41 Lx = 120.0; H = 15.0; Ly = 1.5*H
42 gy = -1.0
43 Vmax = np.sqrt(abs(gy) * H)
44 c0 = 10 * Vmax; rho0 = 1.0
45 p0 = c0*c0*rho0
46 gamma = 1.0
47
48 soft = 0.01
49 t_hit = 200.0
50 Mass = 1.0
51 tf = 300.0
52
53 # Reynolds number and kinematic viscosity
54 Re = 0; nu = 0.01 # Ideal fluid
55
56 # Numerical setup
57 nx = 1600; dx = Lx/nx
58 ghost_extent = 5.5 * dx
59 hdx = 1.2

82 class BlackHole(Application):

171     def create_equations(self):
172         # Formulation for REF1
173         equations1 = [

194             # Main acceleration block
195             Group(equations=[

212                 # Add the black hole
213                 BlackHole2D(dest='fluid', sources=None, soft=soft,
        t_hit=t_hit, M=Mass)
214
215             ]),
216         ]
```

Figure 1.2: Modifications of the hydrostatic tank example, *blackhole.py*.

Determining the dimensions of the tank required a considerable amount of care. If the tank was too narrow, the walls at the edges would reflect the initial waves which would then start to interfere with the secondary waves. This caused erroneous results in the energy calculations during testing. Similarly, if the tank was too shallow, the waves resembled shallow water waves instead of deep, as in our model. And of course, if the tank was needlessly large, it greatly effected the computation time of the simulation. In the end, $Lx = 120$, $Ly = 15$, and $dx = 0.075$ were decided on.

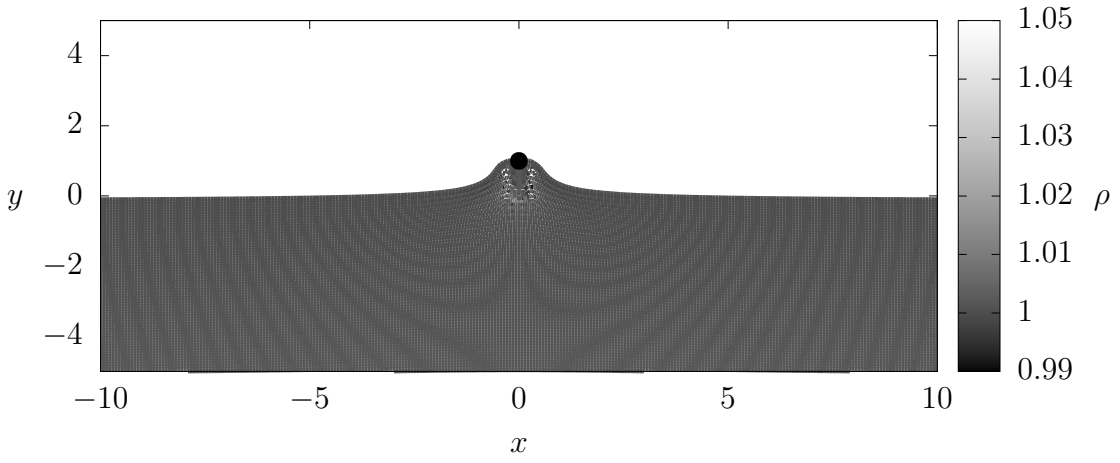
1.3 Simulation Results

In total the simulation took about 330 CPU hours over the course of three days. The resulting surface waves from the collision can be seen in Figure 1.3. As with the analytic results, we see that the surface of the neutron star pulls upwards before the collision due to the gravitational force of the primordial black hole. Then, after the collision, the initial wave propagates outwards. It is fairly difficult to notice, however, a second much smaller amplitude wave is created as well. Unfortunately, because of the resolution of the simulation, waves smaller than dx cannot be seen, unlike in the analytic solution which had a much smaller spacing.

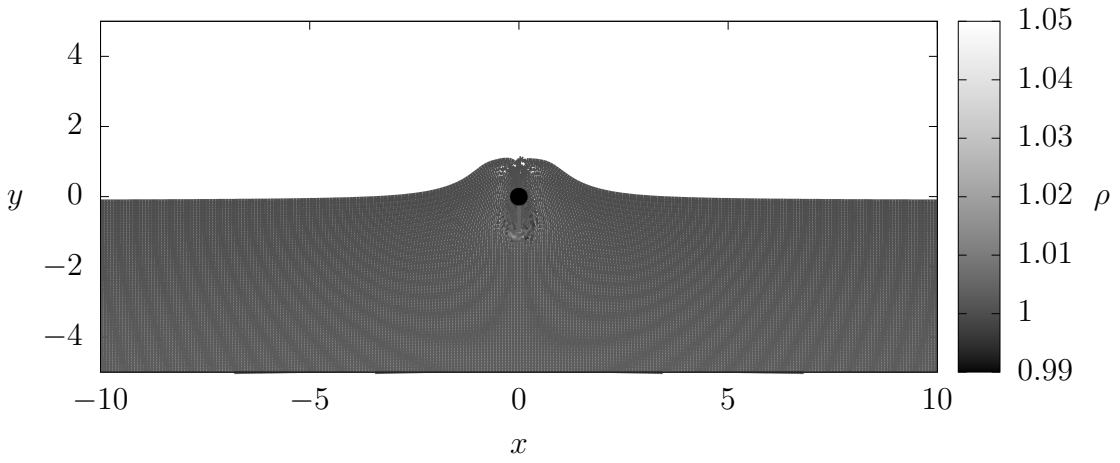
In order to calculate the energy we do so in the traditional way, by taking the sum of the potential and kinetic energies of each particle,

$$\begin{aligned} E &= \sum_i T_i + U_i, \\ &= \sum_i \frac{1}{2} m_i \left(u_i^2 + v_i^2\right) + m_i g y_i. \end{aligned}$$

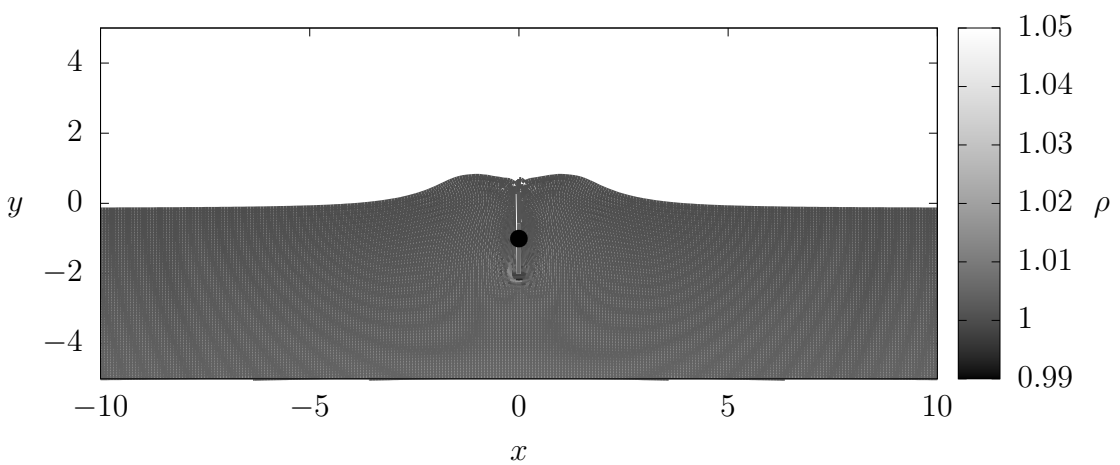
However, to compare to the analytic result, we must transform this into three dimensions



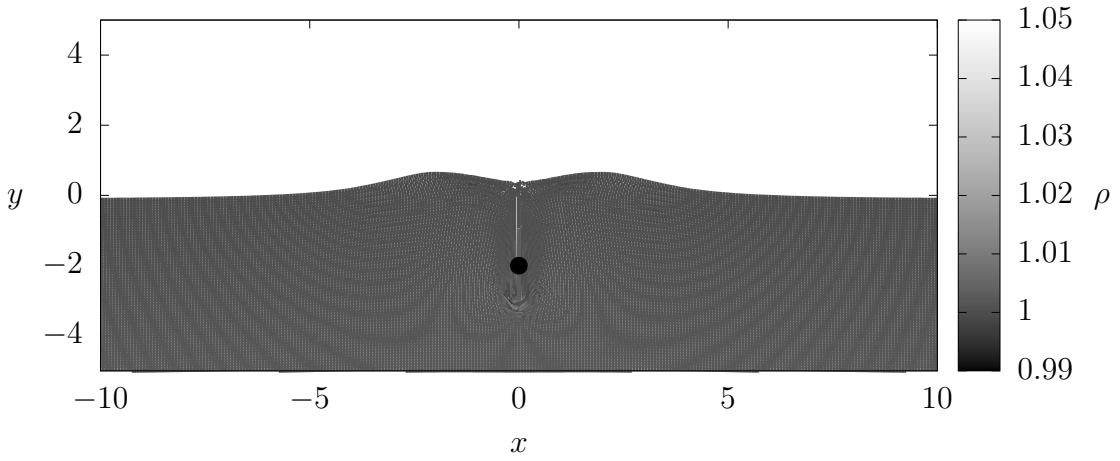
(a) $\tau = -1$



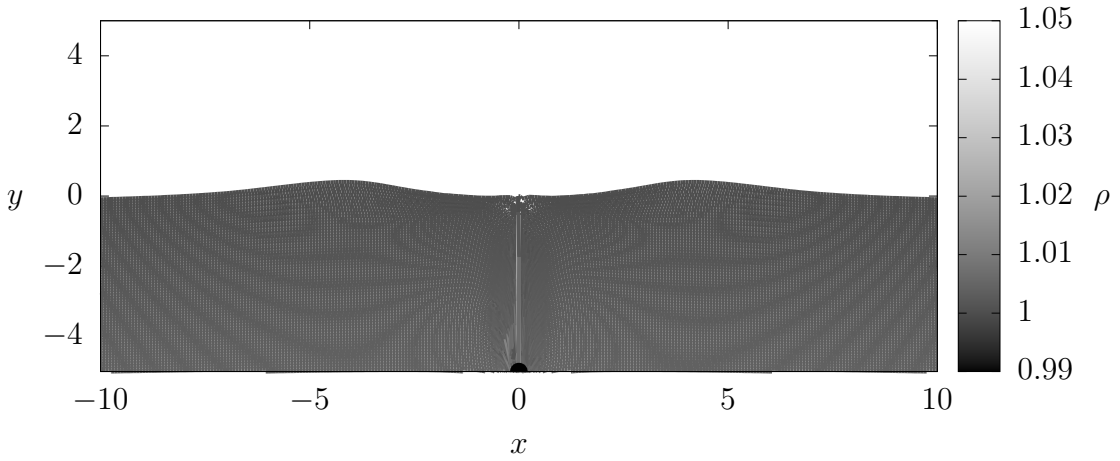
(b) $\tau = 0$



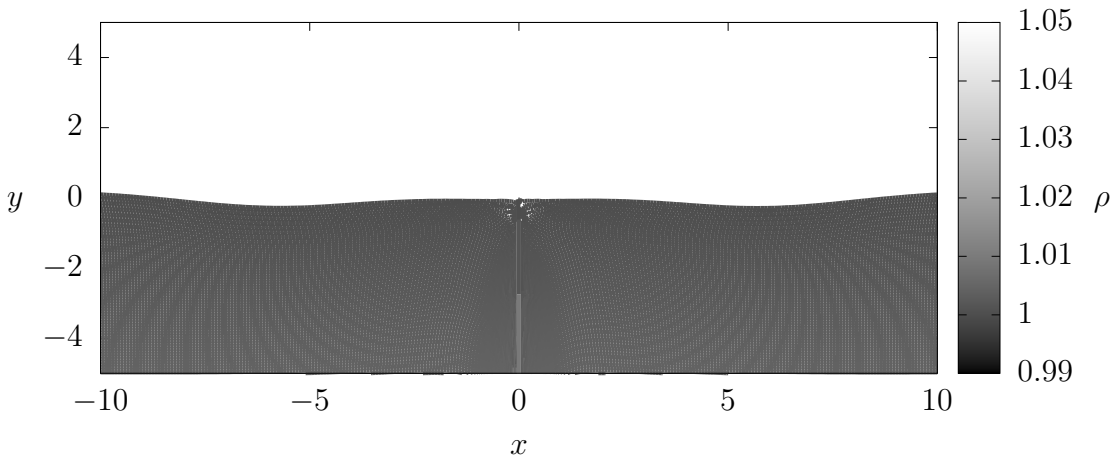
(c) $\tau = 1$



(d) $\tau = 2$



(e) $\tau = 5$



(f) $\tau = 10$

Figure 1.3: Simulated results. ***

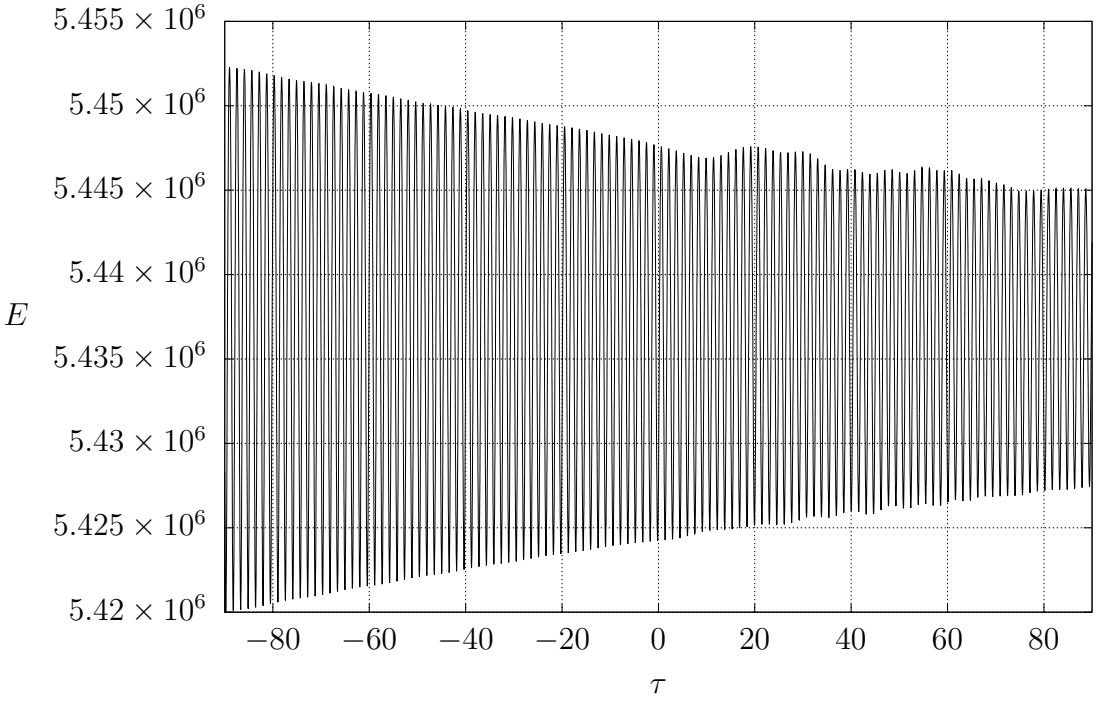
by revolving about the y axis. This can be done by weighting each particle by $\pi|x_i|/dx$,

$$E = \frac{\pi}{dx} \sum_i \left(\frac{1}{2} \left(u_i^2 + v_i^2 \right) + g y_i \right) m_i |x_i|.$$

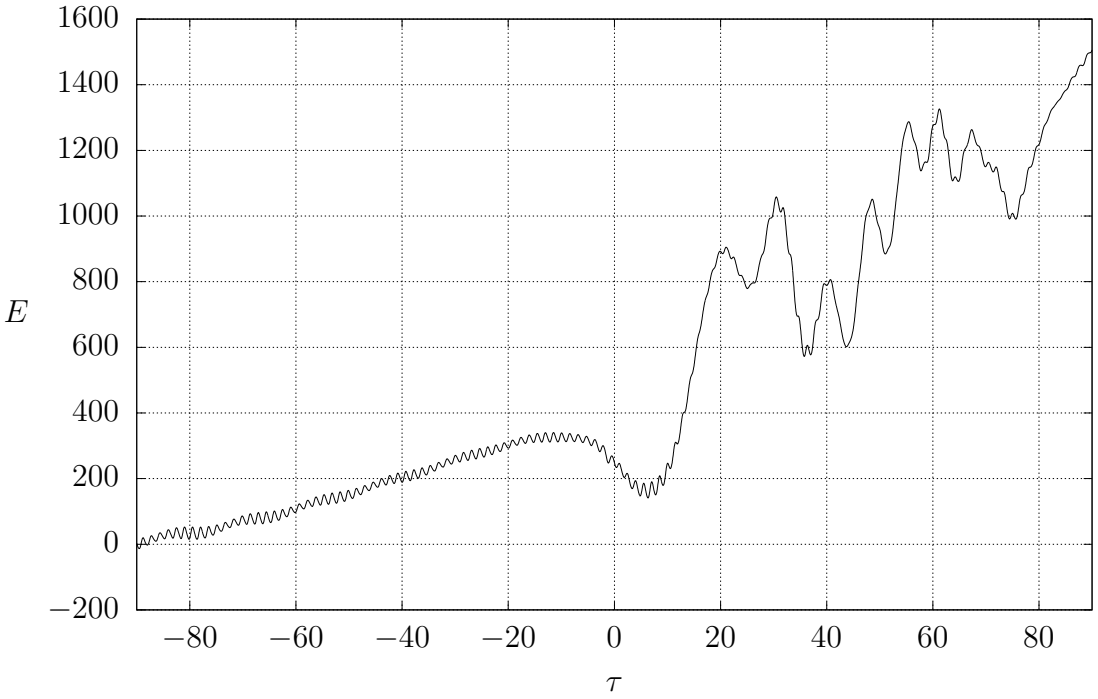
This is essentially the same as a shell integration, however, we only integrate through π, and the absolute value of the x coordinate is needed. The energy is calculated in this fashion for all times steps and is plotted in Figure 1.4a. Clearly, this is of little use since the energy is full of noise and highly oscillatory. As mentioned before, this is also the result of how the particles are initially placed within the simulation. As the surface oscillates up and down while attempting to equilibrate, so too does the potential energy. The oscillations have a period of 39 time steps, and therefore, by taking running averages of 39 time steps, most of the noise is removed. Also, for clarity the damping is removed, and the energy has been shifted to start at 0, the resulting plot is much cleaner and is in Figure 1.4b.

This cleaned up energy has a very similar shape to the analytic calculation; initially, the neutron star gains energy as the primordial black hole approaches. And after the collision there is a large peak in the energy similar to the analytic solution. However, this jump in energy is much larger. Furtheremore, at approximately $\tau = 40$ the initial wave hits the boundaries which causes the energy of the system to increase as the wave climbs the wall. After this the wave is reflected back towards the centre of the tank. This also causes the first wave to interfere with the smaller waves trailing it, which again causes the energy to increase – this is the main cause of the erratic behaviour of the energy at times greater than 40.

Another point of interest is that since the tank has a depth of 15, at $\tau = 15$ the primordial black hole exits the tank. During the lower resolution testing this did not appear to effect the results of the simulation, however, it indeed does. As the primordial black hole exits the tank a shock wave is created. In Figure 1.5 the shock wave can be seen moving outwards



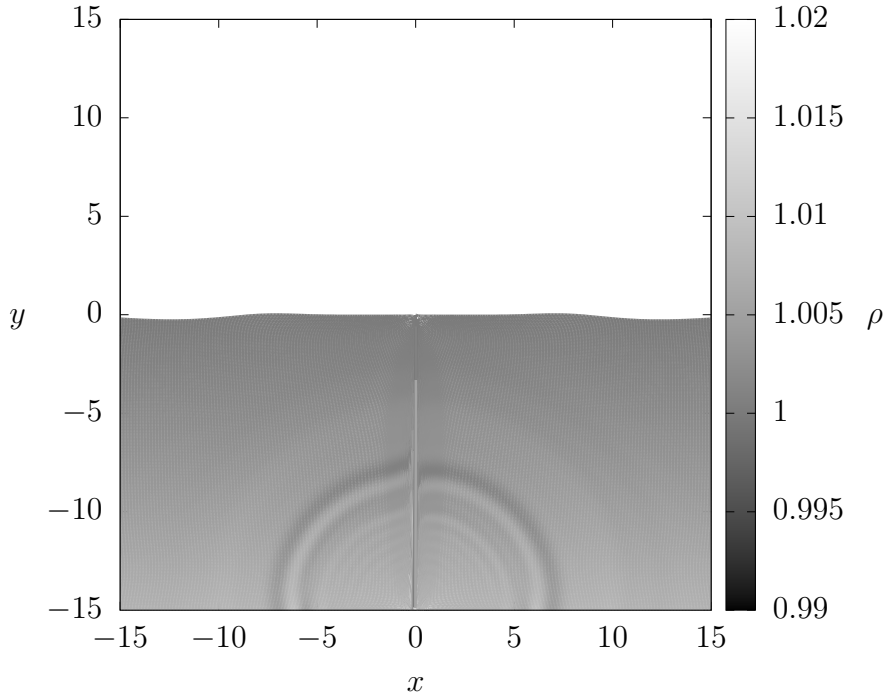
(a) Crazy looking energy.



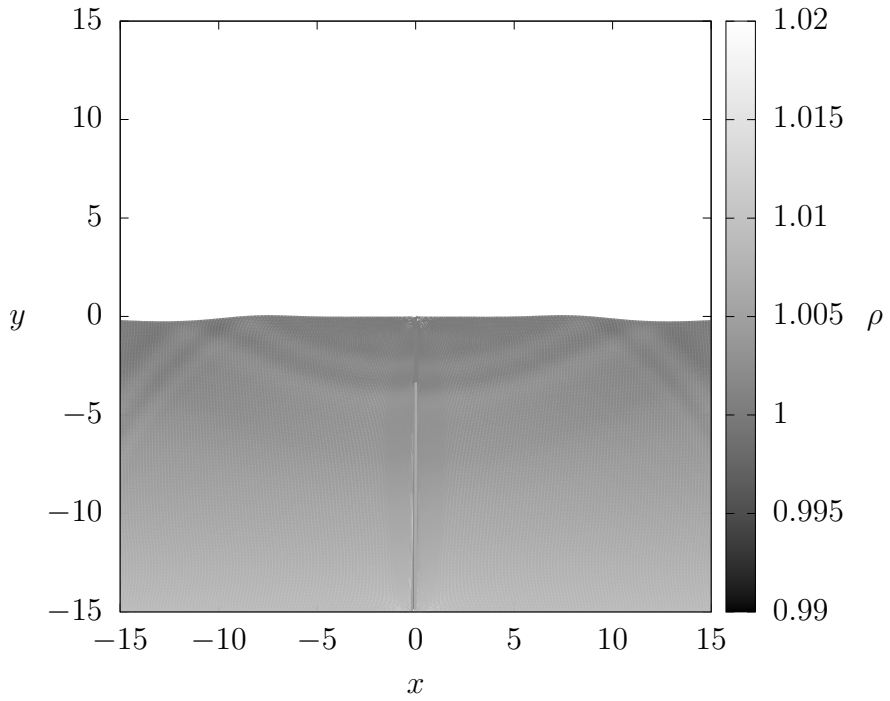
(b) Smoothed energy without noise.

Figure 1.4: Energy transfer from the simulation.

from the bottom of the tank, and when it reaches the surface, it is reflected back into the star. Given the speed of the shock wave and the dimensions of the tank, it will be dampened out within a few time units and does not appear to effect the energy of the system.



(a) Initial shock-wave. $\tau = 15.24$.



(b) Reflected shock-wave. $\tau = 15.56$.

Figure 1.5: Shock-wave created by PBH leaving tank.