Kilonova Simulations

Alexandra Stewart and Erin Vincent

March 31, 2023

1 Introduction

The merger of two neutron stars is thought to produce a kilonova, or a bright electromagnetic emission, and a gravitational wave; because of this, binary neutron star (BNS) collisions are fascinating cosmic events.

The purpose of our code is to properly model the merger of two neutron stars using Smooth-Particle-Hydrodynamics (SPH) methods. Our goal is to use SPH to create two neutron stars, add a gravitational force between the two, animate their inspiral, and provide some deliverables.

2 How it's Going so Far

2.1 Current setup

Our code can be located in the Git repository at `https://github.com/ETVincent/spring_23_computational.git` . There are three current branches for development: main, astew925-patch-1, and collisions. The current SPH model resides in astew925-patch-1 while collisions is being used for development of the general inspiral mechanics using point masses (i.e. prior to applying it to the stars themselves). After some debugging, we expect to merge astew925-patch-1 with main shortly.

Our code resides in a Jupyter notebook. We are currently running it with two different masses that we modify as necessary to test the code. For simplicity and speed during development, we are using a reduced number of particles for the SPH method, generally with N around 400. We plan to increase this for higher resolution once the code is ready to run in full.

2.2 Current status

As of this writing, the code in main is capable of outputting two neutron stars modeled with SPH. In the initial project proposal, we had managed to create one SPH neutron star. We have since been able to move many of the neutron star creation processes into functions that can be called to produce two stars, each with a different mass.

Inside astew925-patch-1, we have converted the original functions for creating a neutron star into a star class object. After some debugging, we will be able to initialize two star objects for use. We hope that using this class will simplify the process.

We have also been making progress towards the collision model, though the current code in collisions is very simple and does not yet operate as desired. We are currently testing out the dynamics on point masses to see if we can update the force, position, and velocity accordingly. We were thinking of implementing the gravitational force for each particle, though we plan to first apply the gravitational force to the center of mass of both neutron stars and update the position of each particle based on how the center of mass should have moved. If that is working, we will likely go on to attempt applying the gravitational force per particle.

2.3 SPH Star/EOs Class model

For easier bookkeeping of code and implementation of the star dynamics, we have implemented two classes in python: A neutron star class `NS` and an `EOS` class.

2.3.1 NS Class

The `NS` Class is initialized with a mass, radius, and a tag (string identifier). In total, the star has the following attributes: `tag`, `mass`, `radius`, `num_points`, `radius_unit = 1`, `m` (equal to the mass of each particle), `rho` (the density sampled at the point positions), as well as arrays containing each particle's corresponding positions (`pos`), velocities (`vel`), and accelerations (`acc`) which are dimension (`num_points` x 3). Upon initialization, the star generates initialized point locations and velocities from a random seed. The code then updates `pos` `vel` and `acc` upon calling a separate function `simulate` (which is called on the star after it is initialized). At this point, a function to sample and update the densities of each point is the next step on the star class implementation.

2.3.2 EOS Class

The `EOS` class is designed so that it can contain several known Neutron star parameters, which can be intialized once and then equations, constants, and other important model parameters can be called easily from the EOS object. An EOS object contains parameters relevant to the particular model, including a dictionary which allows constants to be referenced for the calculations. For example, the value of an equation of state constant, k could be called with `k = eos.constants["k"]` from the initialized EOS rather than keeping it in the global environment. At the moment, the EOS only contains one case, called 'standard', which very generally implements the Tolman-Oppenheimer-Volkov equation for the classical case. Now that the code is conveniently working for these two classes this code needs to be further modified; our next step is to tweak the physics of the parameters for our standard EOS class to get a realistic star initialization which matches a realistic NS mass and radius.

$$\frac{dP}{dr} = \frac{Gm}{r^2}\rho(1+\frac{P}{\rho c^2})(1+\frac{4\pi r^3 P}{mc^2})(1+\frac{2Gm}{rc^2})^{-1} \tag{1}$$

All sounds good in the classes.

2.4 Changes of plan

Dr. Wise noted in our project proposal that we could instead use a leapfrog method for time evolution, which we are currently planning to implement but have not yet made progress towards.

At the beginning, we were planning on having specific functions for the creation of the neutron stars, as seen in main. We have since transitioned into a class model wherein we hope that the particles can easily maintain their position even after being moved and the star more easily carries its specific characteristics.

3 List of Figures

First, we plan to provide two figures per neutron star: a distribution of the SPH particles and a plot of density vs radius. An example can be seen in Figures 1 and 2.

We also plan to provide a figure demonstrating the inspiral itself. In our code, we hope for this to be a moving image. Since the poster will be a static image, we will have the inspiral path outlined.

We currently make figures for pressure vs radius and mass fraction vs radius, though we may or may not include these in the final report. See Figures 3 and 4.

One interesting figure to make is the mass fraction that is bound to one of the NSs as a function

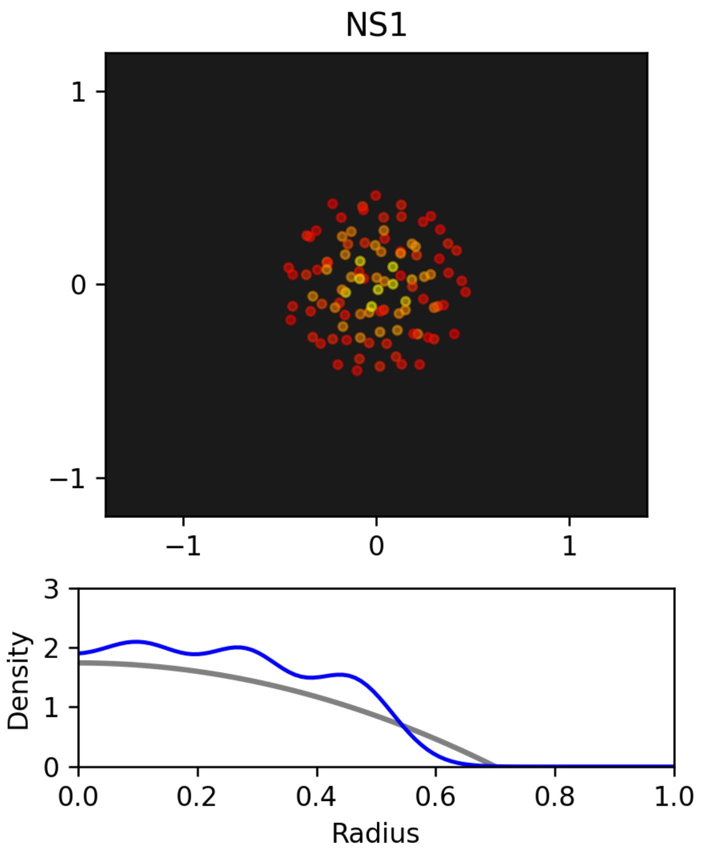

Figure 1: *Neutron star with N=400 and mass approximately equal to the sun.*

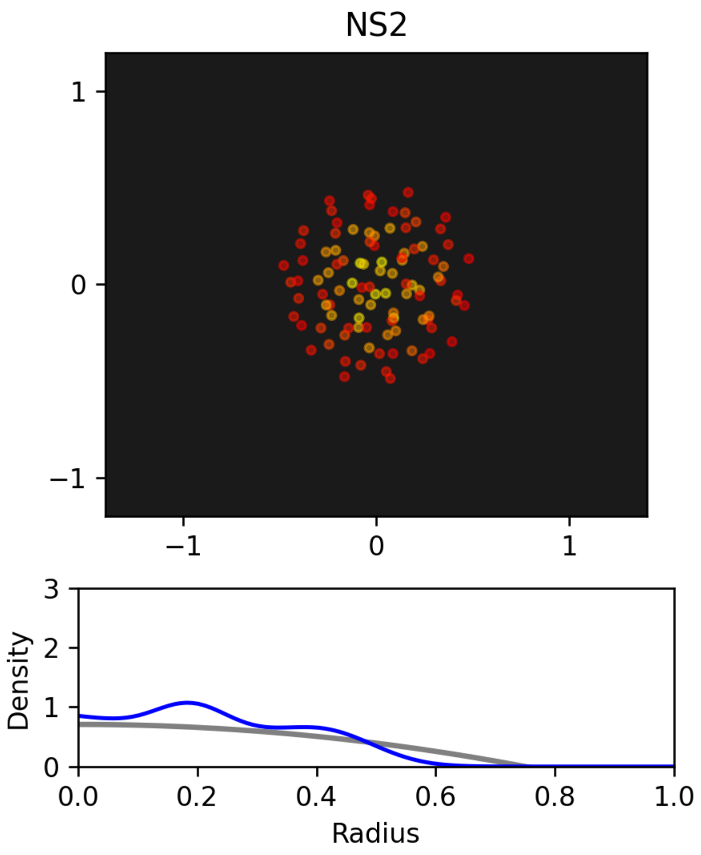

Figure 2: *Neutron star with N=400 and half the mass of NS1.*

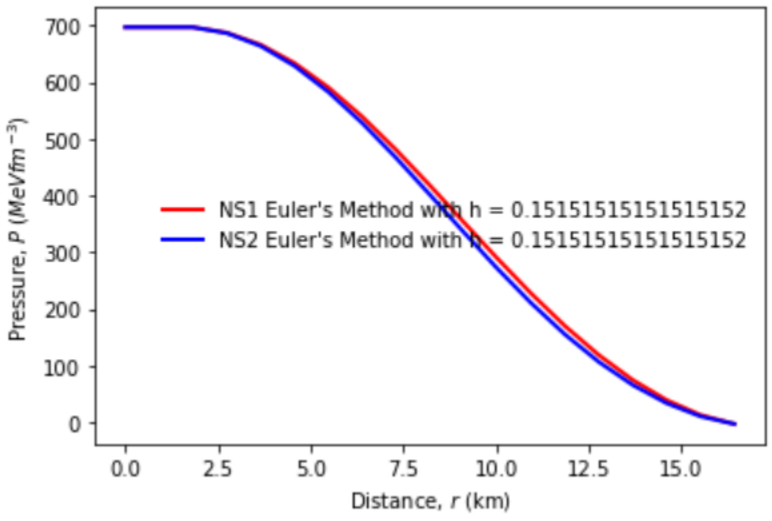

Figure 3: *Pressure vs radius for the two neutron stars in Figures 1 and 2.*

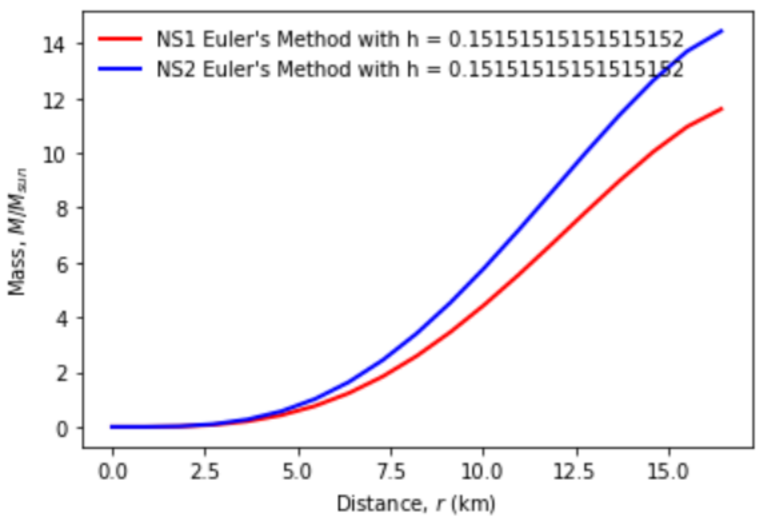

Figure 4: *Mass fraction vs radius for the two neutron stars in Figures 1 and 2.*