PHYSICS 410 - Project 1 write up

Rodrigue de Schaetzen

October 24, 2020

1 Gravitating masses

1.1 Toomre model

The Toomre model is a simplified model to describe the dynamics of galaxy interactions including collisions. There are two major assumptions that facilitate the model complexity. First, each galaxy consists of a central point particle called a "core" which consists of gravitating mass. Second, stars that orbit a galaxy only experience gravitational influence from cores and not from other stars. These two components make it possible to implement an effective and efficient simulation for galaxy interactions/collisions.

1.2 FDA for two gravitating masses

In this section we derive the Finite Difference Approximation (FDA) for two gravitating masses. The finite difference method works by converting the Ordinary Differential Equation (ODE) to an algebra problem described by the following steps:

1. Discretize the dependent variable (in this case time) into a mesh grid
2. Derive the necessary Finite Difference approximations for the derivatives that are part of the ODE
3. Replace the derivative terms in the ODE with these approximations
4. Solve the algebraic expression such that the next solution is determined iteratively by previous solutions

We can apply these steps to provide a solution for the Toomre model. The second-order centered FDA for second-order differential equations is given by Equation 1. This is a 2nd-order approximation which means as $\Delta t \to \frac{\Delta t}{2}$ the $error \to \frac{error}{4}$.

$$\frac{d^2y^n}{dt^2} \approx \frac{y^{n+1} - 2y^n + y^{n-1}}{\Delta t^2} \tag{1}$$

From the N-Body notes we have the equation of motion where subscripts represent the masses acting on each other:

$$\frac{d^2\mathbf{r}_i}{dt^2} = \sum_j \frac{m_j}{r_{ij}^3}\mathbf{r}_{ij} \tag{2}$$

Using Equation 1 we get the following:

$$\frac{d^2\mathbf{r}_i}{dt^2} \approx \frac{\mathbf{r}_i^{n+1} - 2\mathbf{r}_i^n + \mathbf{r}_i^{n-1}}{\Delta t^2} \tag{3}$$

Since we can easily define/compute $\mathbf{r}_i^n$ and $\mathbf{r}_i^{n-1}$ by specifying the position vector and velocity vector initial conditions, we can solve for $\mathbf{r}_i^{n+1}$ using Equation 2:

$$\mathbf{r}_i^{n+1} = \Delta t^2 \sum_j \frac{m_j}{r_{ij}^3} \mathbf{r}_{ij} + 2\mathbf{r}_i^n - \mathbf{r}_i^{n-1} \tag{4}$$

Regarding $\frac{d\mathbf{r}_i}{dt}$ (i.e. $\mathbf{v}_i$) we can solve it using the FDA for first-order differential equations:

$$\mathbf{v}_i = \frac{d\mathbf{r}_i}{dt} \approx \frac{\mathbf{r}_i^{n+1} - \mathbf{r}_i^{n-1}}{2\Delta t} \tag{5}$$

Shown above are the necessary equations (Equations 4, 5) to provide a numerical solution to two mutually gravitating masses. The following are names and descriptions of the two MATLAB functions implementing this approach in 3 dimensions:

1. **nbody.m** - Solves the nbody problem using second order FDA
2. **nbodyacc.m** - Computes the nbody acceleration (Equation 2)

1.3 Setting the initial conditions

The next step consists of defining reasonable initial conditions to qualitatively test the solution. The configuration specified in Section 1.5 of the N-Body Problem notes provides initial conditions for two masses in mutual circular orbit about their center of mass.

Running our scripts with the described configuration (**nbody_test.m**) we get Figures 1, 2. Both plots display circular trajectories in the xy plane of two mutually orbiting particles around their center of mass. Note the x and y axes are scaled differently which is why their orbits look elliptical. The arrows display the initial positions and velocities. As expected, the orbital trajectories are identical for the case of identical masses in the second figure since the acceleration term is equal. Due to these expected behaviors/trajectories we can increase our confidence in the correctness of our solution.

1.4 Three-level convergence test

A 3-level convergence test was used to quantitatively check the error in the FDA is proportional to $O(\Delta t^2)$. In other words, if we increase the resolution of the time mesh by a factor of 2 then we expect the error to go down by a factor of 4. Figure 3 displays the solutions for the x coordinate for one of the orbiting particles in the two-orbiting mass setup. Three different time resolutions are shown where a step increase in level is equivalent to dividing Δt by 2. From these solutions we compute the approximate error by taking the difference of two solutions at consecutive levels. Finally, the errors are scaled based on their increased resolution (or decreased error quantity) and superimposed to compare the 3 curves. These steps are displayed in Figure 4 which clearly displays the Δt^2 nature of the error quantity. The script **nbody_conv_test.m** contains the code to make the necessary plots for the 3-level convergence test.

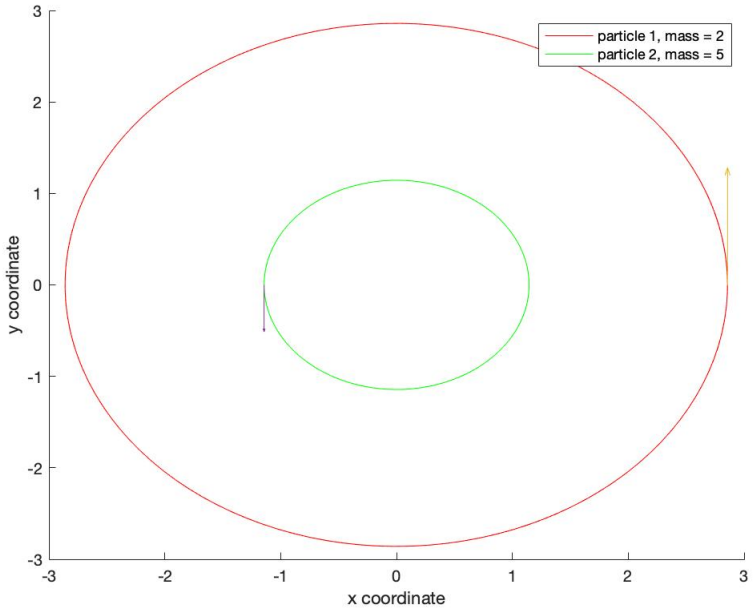

Figure 1: Two mutual circular orbits of particles with different masses.

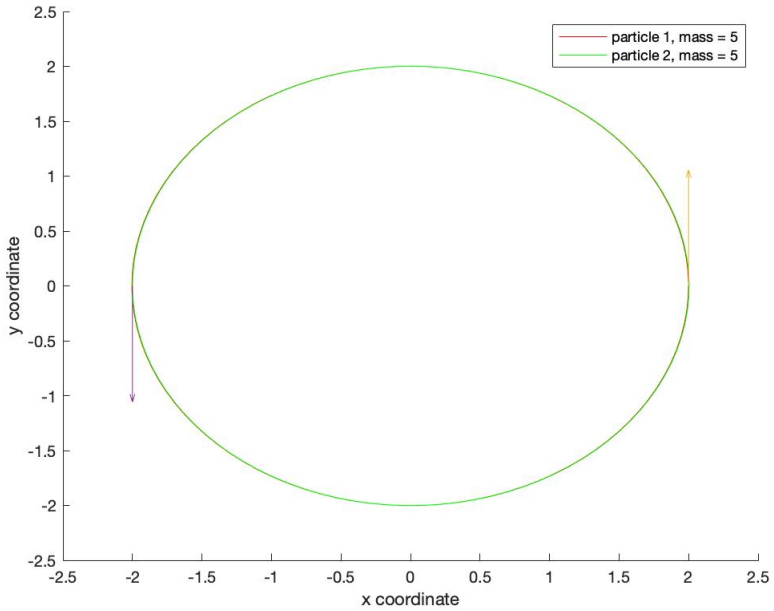

Figure 2: Two mutual circular orbits of particles with identical masses.

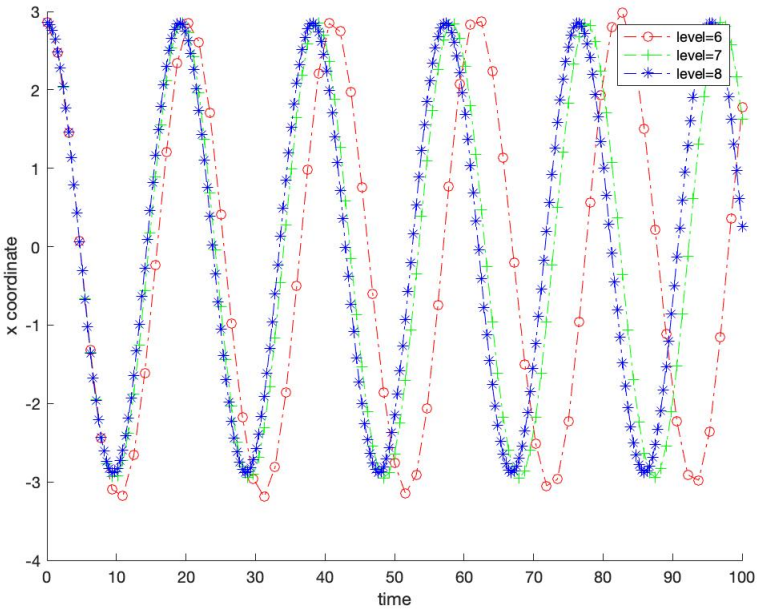

Figure 3: Solutions of the x coordinate for an orbiting particle. 3 different levels or time resoltions are shown.

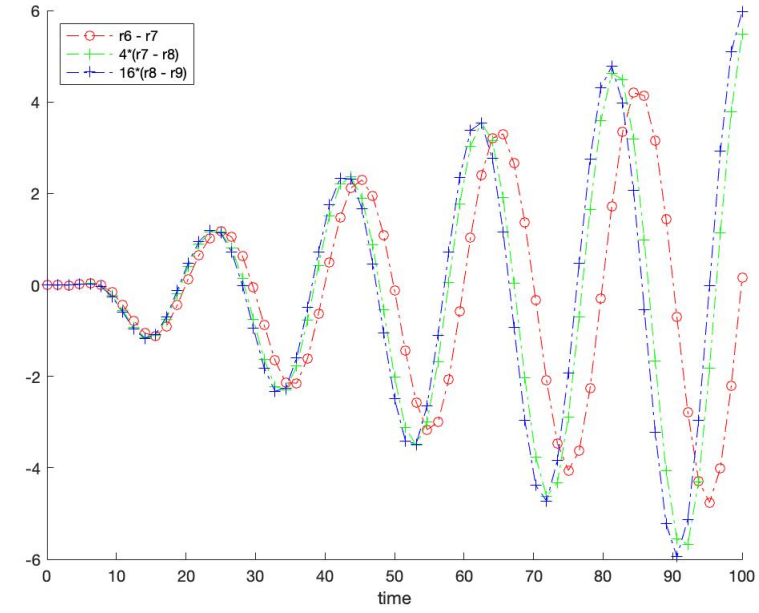

Figure 4: The difference in solutions between two consecutive levels to approximate the errors. These are scaled to display the error is Δt^2 in quantity.

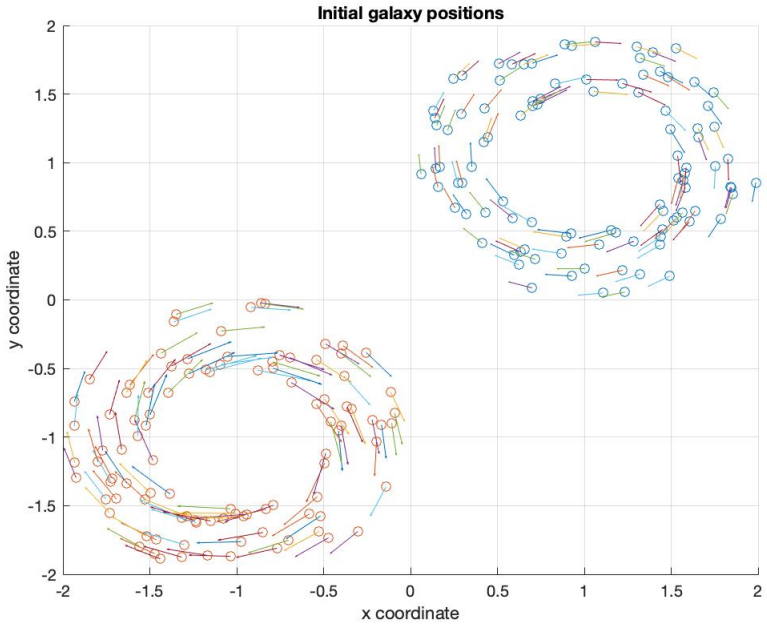

Figure 5: Initial configuration of two galaxies.

2 Galaxy Dynamics

2.1 Adding stars

As described in the previous section, stars in the Toomre model do not have mass so they do not exert gravitational influence on other stars or cores. To incorporate stars into the current model we just need to ignore the terms in the summation that have m_j where j is a star. Additionally, appropriate initial conditions need to be configured so the stars are orbiting cores.

The helper function **generate_stars** in the script **galaxy_sim.m** generates randomly distributed positions of stars around each of the two cores. To obtain stellar obits, the helper **init_star_vel** in the same script computes the necessary initial velocities. Equation 6 was derived by setting the nbody equation of motion (Equation 2) to the centripetal force and solving for the v_i. Its important to note, the $\hat{r}_{ij}$ separation vector is transformed 90 degrees along the z axis so that the velocity is tangential to the surface of the circle/sphere. Figure 5 displays the initial velocity vectors of evenly distributed stars around the two cores.

$$v_i = \sqrt{\frac{M_{core}}{r_{ij}}} * \hat{r}_{ij} \tag{6}$$

The next two experiments were ran to get a qualitative sense of stellar behavior. In Figure 6 we can see a stationary core with the plotted trajectories of 10 orbiting stars. For Figure 7 a core is moving in the positive xy direction and two orbiting stars are following this trajectory.

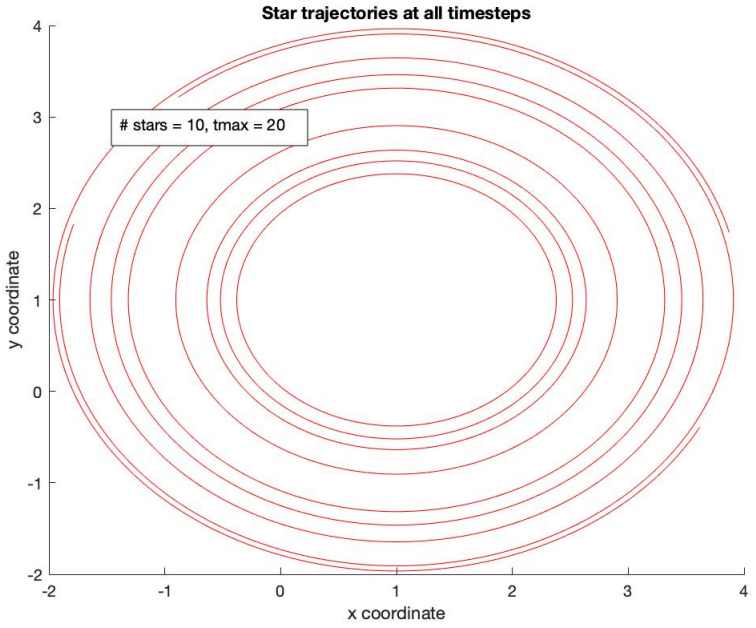

Figure 6:

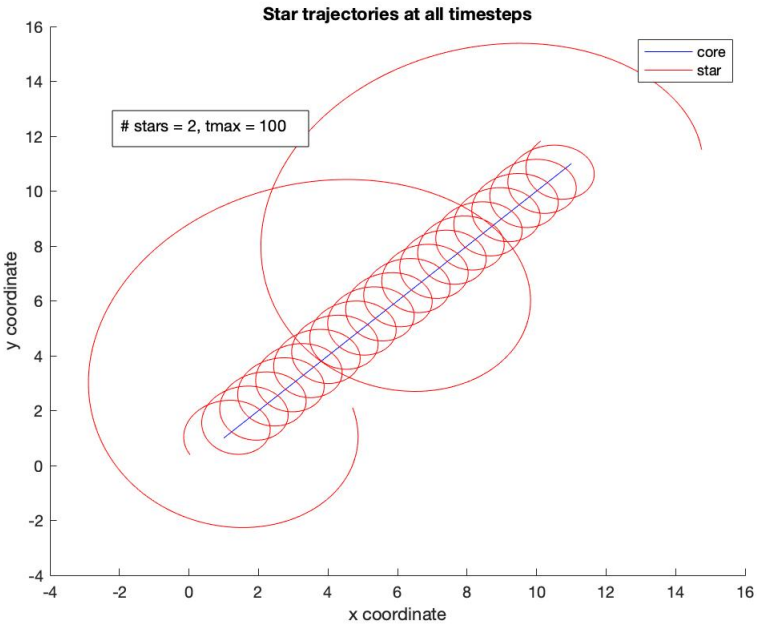

Figure 7:

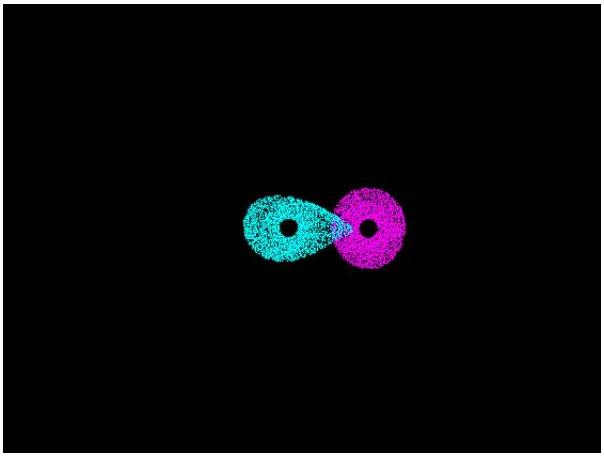

Figure 8: Galaxy snapshot at timestep=tmax/2.

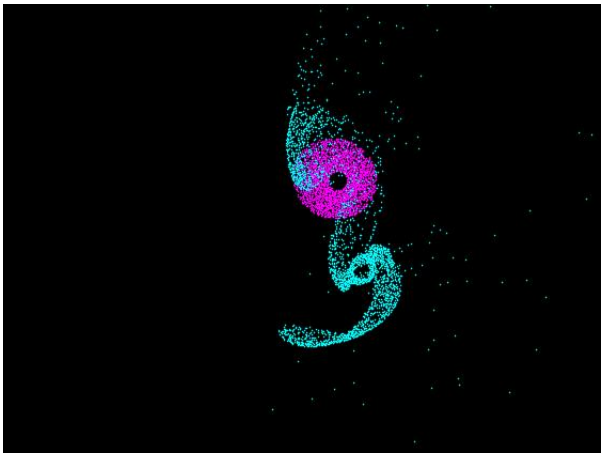

Figure 9: Galaxy snapshot at timestep=tmax.

2.2 Galaxy Collisions

The final step consists of tuning the parameters to produce galaxy collisions and capturing the positions of all the particles in the system to generate animations. The script **galaxy_sim.m** is the wrapper which specifies the simulation parameters and calls all the necessary helper scripts to achieve this final step. Once the positions are computed for all timesteps for all particles, this information is fed to **galaxy_simulation_avi.m** to produce the animation.

Figures 8, 9 display two snapshots of a simulation run. The magenta-heavier galaxy is moving up while the cyan-lighter galaxy is moving down. As the two galaxies approach each other, the stars from the cyan feel a huge gravitational influence from the magenta.

The animation **interesting_collision.mp4** shows a galaxy collision where the galaxies shred each other apart. The resulting galaxies only contain a few left over stars.

2.3 Difficulties Encountered

One of the encountered difficulties was finding the right parameters such that stars do not fly out from the core after a few time steps. This issue was only encountered when star positions are distributed around the core in all 3 spatial dimensions. As a result, a parameter was added to the **generate_stars** helper function to determine whether stars should be evenly distributed in the xy field or xyz.

The other major difficulty was determining the correct steps for computing initial stellar velocities. To debug this issue, the MATLAB **quiver** tool was used to draw arrows for the velocity vectors at time=0. The solution found was applying the 90 degree 3x3 rotation matrix around the z axis.