PHYS 410 - The Toomre Model

Guillermo Cabrera

October 2023

Introduction

Galaxies, celestial structures of immense size and complexity, come in various shapes and sizes, from the regularly structured spirals to the complex ellipticals. However, some galaxies exhibit peculiar and irregular features, and their morphology is often shaped by interactions and collisions with other galaxies. In this project, I aim to implement a simplified model of galaxy collisions, drawing inspiration from the work of Alar Toomre in the early 1970s. While this model cannot capture the intricate details of galactic interactions, it is capable of replicating fundamental morphological features observed in actual galactic collisions.

The Toomre model, which we will explore in detail, simulates an isolated galaxy as a central particle, referred to as a core. This core possesses a gravitating mass while being orbited by stars in circular orbits of varying radii. Notably, the stars have negligible gravitating mass, meaning they are influenced by the core's gravitational pull but do not contribute to the gravitational field themselves. To model a collision of two galaxies, we introduce two cores, each surrounded by a set of stars. As these galaxies approach each other, all stars experience the gravitational influence of both cores, creating complex interactions.

This project's primary objective is to implement the Toomre model and explore galaxy dynamics, studying how stars orbit cores and how galaxies interact during collisions. I will examine the convergence of my numerical simulation by conducting convergence tests on a simplified two-particle system. The analysis will include assessing the error scaling with time step size, demonstrating that the implementation achieves $O(\Delta t^2)$ error.

As I progress, I will evolve from simulating stable galactic orbits to more dynamic scenarios involving galaxy motion, all the way to complex and "glancing" galaxy collisions. The final project will include visualizations, including representative snapshots of particle configurations and an AVI (MPEG) movie of an interesting simulation. Additionally, the project will provide a thorough description of the underlying theory, the finite difference equations used, and the code structure.

In summary, this project delves into the fascinating world of galactic collisions and provides insights into the methods and simulations that enable us to understand the morphological transformations resulting from these cosmic interactions.

Review of Theory and Numerical Approach

In this problem we are trying to understand the dynamics of the system due to the gravitational force exerted by each body on every other body.

By combining Newton's second law with the law of gravitation, cancelling the m on both sides, and letting $G = 1$ we get an equation as follows. $(i = 1, 2, ...N)$

$$\mathbf{a}_i = \sum_{j=1, j\neq i}^{N} \frac{m_j}{r_{ij}^2}\hat{\mathbf{r}}$$

Where r_{ij} is the magnitude of the separation vector $\mathbf{r}_{ij}$ between particles i and j.

We also know that,

$$\mathbf{a}_i = \frac{d^2\mathbf{r(t)}}{dt^2}$$

so,

$$\frac{d^2\mathbf{r(t)}}{dt^2} = \sum_{j=1, j\neq i}^{N} \frac{m_j}{r_{ij}^2}\hat{\mathbf{r}}$$

A numerical approximation for the second derivative is given by this second order centered formula

$$\frac{r^{n+1} - 2r^n + r^{n-1}}{\Delta t^2}$$

So we can equate

$$\frac{r^{n+1} - 2r^n + r^{n-1}}{\Delta t^2} = \sum_{j=1, j\neq i}^{N} \frac{m_j}{r_{ij}^2}\hat{\mathbf{r}}$$

Now that we have made the problem discrete, we can start implementation. First we will need to initialize the first two steps of this three-time-level scheme. The first time step will simply be set to some initial position, while the second time step will be computed using the second order Taylor series expansion of displacement (r), which requires an initial acceleration and the initial position.

We will also need a way to compute acceleration for each time step, we can achieve this by implementing the acceleration equation given above (more on the next section).

Implementation

I spent a while trying to figure out how to compute acceleration, at first I tried using an approach that computed the vector of separations for each component x, y, z separately, this method involved using matrices and linear algebra calculations, did not require a for loop so the code looked more clean and the computation was probably more efficient, but it ended up not working. The two bodies would just diverge as soon as my animation started, I tried to fix it with many different approached but none worked. Here is an example of this approach.

```
x = pos(:,1);
y = pos(:,2);
z = pos(:,3);
% matrix that stores all pairwise particle separations: r_j - r_i
dx = x' - x;
dy = y' - y;
dz = z' - z;
% matrix that stores 1/r^3 for all particle separations
inv_r3 = (dx.^2 + dy.^2 + dz.^2 + softening.^2).^(-3/2);
inv_r3(inv_r3 == Inf) = 0;
ax = (dx .* inv_r3) * mass;
ay = (dy .* inv_r3) * mass;
az = (dz .* inv_r3) * mass;
```

I ended up using the alternate approach of of using for loops and calculating the displacement vector all at once instead of by component, bellow is the code snippet.

```
    for i = 1 : bodies
        acc_j = zeros(1, 3);
        for j = 1 : length(m)
            if j ~= i
                d = abs(r(j, :) - r(i, :));
                rij = norm([d(1), d(2), d(3), softening]);
                acc = m(j)*(r(j,:) - r(i, :))/(rij)^3;
                acc_j = acc_j + acc;
            end
        end
        acceleration(i, :) = acc_j;
    end
```

As you can see above the code incorporates a **softening parameter**, which is a minute value introduced to prevent numerical complications that arise when two particles come into close proximity. This situation can cause the acceleration, calculated using the 'inverse square-law,' to become exceedingly large. In the physical world, masses are not idealized as point particles; they possess finite size and extent.

Implementing FDA

At first I had some difficulties in finding the second position step required for the FDA at first I tried to just create all the time steps at once for each of the cores, but then I realized that I would have to generalize this to N stars, so I realized I should use a for loop, that realization helped me in creating the second for loop that actually implements the finite difference method.

Overall Structure of Simulation Code

The following is a description of my simulation code (Nbody function).

First we start off with initializing, we define the level parameter, the time steps, and the time span. We also define the number bodies in the system N, and the create a 3D position array of zeros (soon to be edited).

```
        nt = 2^level + 1;
    dt = tmax / (nt - 1); % discrete timestep
    t = 0 : dt : tmax;
    N = sum(stars) + length(m); % total number of bodies

    pos = zeros(N, 3, nt); % mesh set up
```

We then move onto defining the initial values for position and velocity using the helper function starmetrics which computes velocity for body given a random position or radius.

```
    [ri, vi] = fn2(m, position, velocity, stars, spin);
    position = [position; ri]; % add random star positions
    velocity = [velocity; vi]; % add calculated star velocities
```

We define an initial acceleration, as well as the positions of the first two time steps (using a for loop).

```
    a1 = fn1(m, position, softening); % intial acceleration
    for i = 1 : N % calculate the first and second inital
       displacements
        pos(i, :, 1) = position(i, :);
        pos(i, :, 2) = position(i, :) + velocity(i, :)*dt + (a1
           (i, :) * dt^2)/2;
    end
```

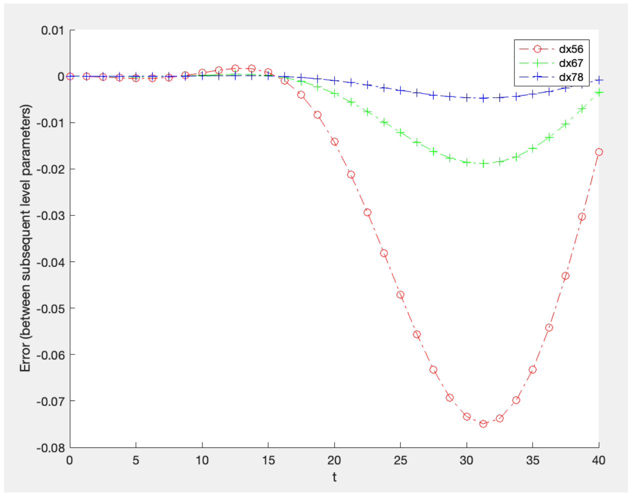
And lastly, we perform the iteration of to find the evolution of positions over time for each body as such

```
        for n = 2 : nt - 1 % calculates rest of timesteps
        a = fn1(m, pos(:, :, n), softening);
        pos(:, :, n+1) = 2 * pos(:, :, n) - pos(:, :, n-1) + a
           * dt^2;
    end
```

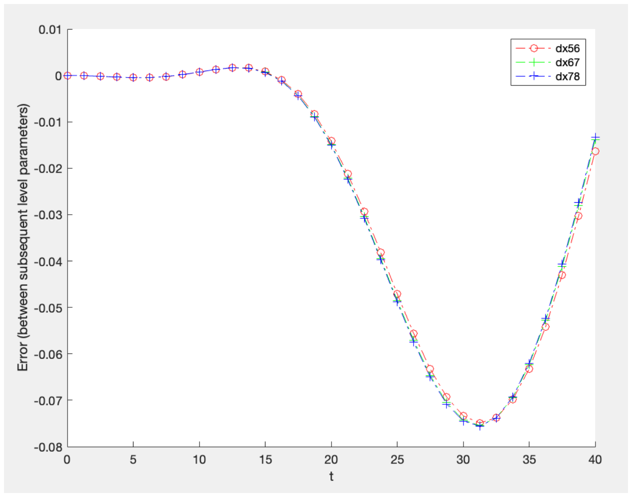
Results

My results for the convergence test of the FDA for the two cores were as expected.

Without scaling, the figure bellow illustrates how the difference between consequent levels becomes smaller and smaller indicating that the results of the simulation are reaching a more accurate solution.



Furthermore, we can re-scale those differences (downsampling)to show that the error appears to have $O(\Delta t^2)$. For example I re-scaled the second error computation by a factor of 4, which is done because a second-order method should exhibit errors that decrease by a factor of 4 when the step size is halved. The figure bellow illustrates the expected behaviour.

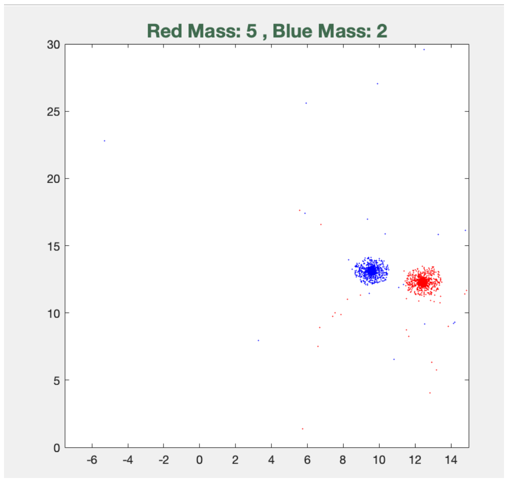


Finding the right initial conditions

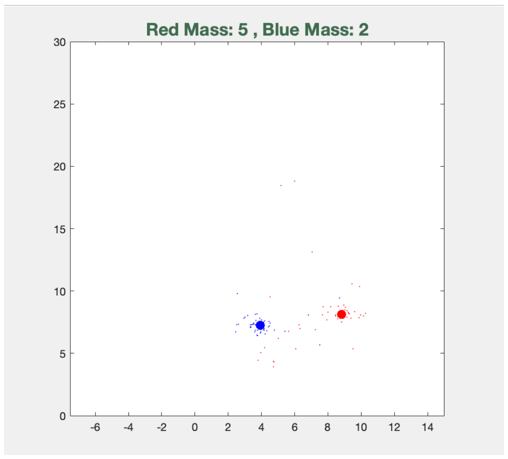
When actually implementing the simulation it self I had to find the correct initial conditions such that,
a. I captured the collision action at about halfway through the video
b. There was enough stars to observe a specific trend/pattern during and after collision.
c. The softening parameter was big enough for the stars not to be shot out of the frame, but small enough that it wouldn't alter the overall acceleration too much.

Bellow I demonstrate the result of changes that I made that illustrates each point.

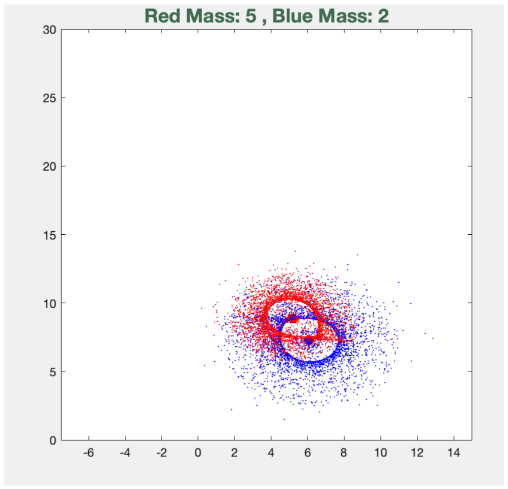
The snapshot is taken at the end of the evolution, as you can see not much happened before that, probably because the stars were placed too far apart.



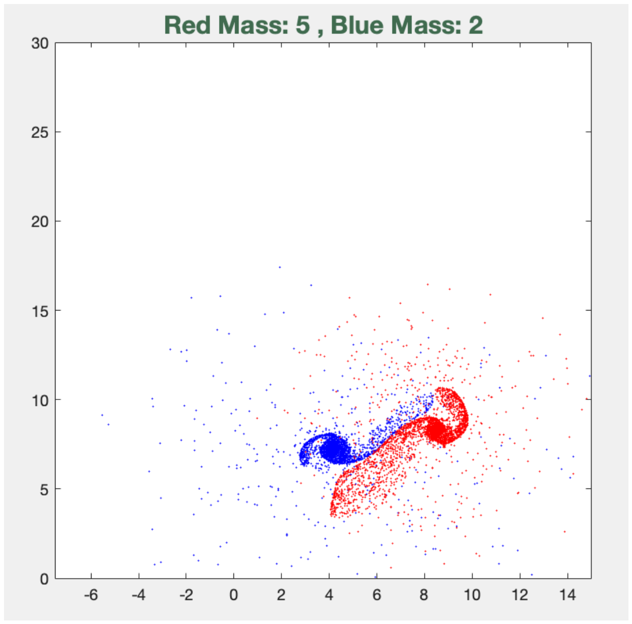
b. As you can see there is just not enough stars to see a distinctive behaviour after collision



c. In this case, the softening parameter is 0.5, as you can see by the coupling up of the start in orbit that this is probably a bit to high and distorts the simulation, I found that a parameter of 0.1 did the best job.



This was the result after applying these considerations (after collision),



Discussion and Conclusion

In this project, I implemented the Toomre model to simulate galaxy collisions and investigated the dynamics of galactic interactions. The project involved several key phases:

Convergence Testing: I began by conducting a convergence test on a two-particle system with distinct masses. This test was essential to validate the finite difference solution and demonstrated that the implementation exhibits second-order accuracy, with an error scaling of $O(\Delta t^2)$.

Initial Conditions: Finding suitable initial conditions for galactic collisions was a crucial step. I needed to ensure that the collision dynamics were captured effectively, sufficient stars were present to observe patterns, and the softening parameter was appropriately chosen to avoid stars being ejected from the simulation.

Finite Difference Approximation: I employed a three-time-level scheme finite difference approximation for motion, which allowed for the evolution of particle positions without relying on ODE integrators. This approach proved to be suitable for simulating galaxy interactions and dynamics.

Throughout the project, I encountered certain challenges and made important decisions in the implementation. One of the main challenges was determining the proper acceleration calculation. I initially attempted a matrix-based approach to calculate pairwise particle separations, but this method led to unsatisfactory results. After shifting to a for-loop-based calculation that computed displacements in one go, I achieved more stable and accurate results.

The choice of softening parameter was also critical. The softening parameter prevented numerical issues when particles came into close proximity, ensuring that the simulation remained stable. The correct choice of this parameter was crucial to capturing realistic collision dynamics without excessive distortion.

In summary, this project provided valuable insights into the complexities of simulating galactic collisions. By conducting convergence tests, fine-tuning initial conditions, and implementing a finite difference solution, I successfully explored the behavior of galaxies in various scenarios. The visualizations and analysis demonstrated the potential of the Toomre model in replicating key features of real galactic collisions. This project highlighted the significance of understanding numerical methods and the challenges of modeling complex physical systems.

AI help: I used the help of AI to learn more about certain topics, for example, I searched up what it means to do convergence testing in this specific context, it helped understand the results better as well as the process.