PHYS 410 Project 1: Toomre Model Write-up

Introduction

This project explores a simplified simulation of galaxy collisions using the Toomre model. The objective is to investigate the orbital dynamics of stars in two galaxies (each represented by a massive core with surrounding non-gravitating stars). The stars move under the gravitational influence of the two cores but do not exert forces on each other or on the cores. The main goal is to implement and test this system numerically, analyze the resulting motion, and perform a convergence test to verify that the system follows the correct order of error from the finite-difference approximation.

Review of Theory

Each core has a mass m_1 and m_2 and is initially placed at specific positions with given velocities (ideally chosen so that the two cores pass each other in close proximity to observe their gravitational interaction). Even though the stars' gravitational effect is not incorporated in this project, each star moves under the gravitational forces from both cores; each core is also under the gravitational effect of the other core.

The acceleration of a star located at position r_i is calculated using:
$a_i = G\,[\,m_1 (r_1 - r_i) / |r_1 - r_i|^3 + m_2 (r_2 - r_i) / |r_2 - r_i|^3\,]$

Each core also experiences acceleration due to the other core:
$a_1 = G\,m_2 (r_2 - r_1) / |r_2 - r_1|^3$
$a_2 = G\,m_1 (r_1 - r_2) / |r_1 - r_2|^3$

In this project, the gravitational constant G is set to 1 in order to simplify calculations. Thus, we can avoid working with units for quantities like position r, velocity v, and mass m.

Stars are initialized with random angles θ between 0 and 2π and random radii between r_{min} and r_{max} in the xy-plane. They are also initially set to have circular orbits around the cores, so their velocities follow:
$v\_cir = \sqrt{(m\_c / d)}\ \hat{\theta} = \sqrt{(m\_c / d)}\ (-\sin\theta\ \hat{x} + \cos\theta\ \hat{y})$

Thus, their total initial velocity is:
$v = v\_cir + v\_core$

In addition, I used certain Cartesian and spherical coordinate conversions for the stars' initial position and velocity:
$x = r \sin\varphi \cos\theta$
$y = r \sin\varphi \sin\theta$
$z = r \cos\varphi$
$\hat{\varphi} = -\sin\varphi\ \hat{x} + \cos\varphi\ \hat{y}$

In this project, vector operations are in 3D, but the simulation is restricted to 2D (with $z = 0$, thus $\varphi = \pi/2$) for simplification.

Numerical Approach

The equations of motion are solved using a second-order finite-difference approximation. The position is updated using:

$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + (\Delta t)^2\, a(t)$

Thus, we need to determine the initial position and the second position of each object to perform the approximation for the next position. The initial positions and velocities are chosen, and the second positions are found by:

$r(\Delta t) = r(0) + \Delta t\, v(0) + (\Delta t)^2\, a(r(0)) / 2$

At each step, the positions and accelerations of all objects (cores and stars) are updated. The total number of time steps is defined as $n_t = 2^{level} + 1$, where "level" determines the time resolution and allows for convergence testing.

After successfully performing the simulation and generating its video, a convergence test is performed to ensure the correct order of error. Since the updating formula is derived from the finite-difference approximation of r'' with second order in time, we expect the ratio:

$R = (u^{jl} - u^{jl+1}) / (u^{jl+1} - u^{jl+2}) = 4$

where u is the position of any object in either the x or y direction.

A series of simulations were performed with varying discretization levels (*level = 8–10*). The total number of stars in both cores was set to 10,000; the mass of both cores was set to 1; and the minimum and maximum radii were set at 0.25 and 1.5, respectively. The trajectories of the two cores and their surrounding stars were visualized over time using MATLAB's VideoWriter to create simulation animations.

Results

The simulation video of each level is attached with this write-up. Below is an image from one of the simulations at time level 8.

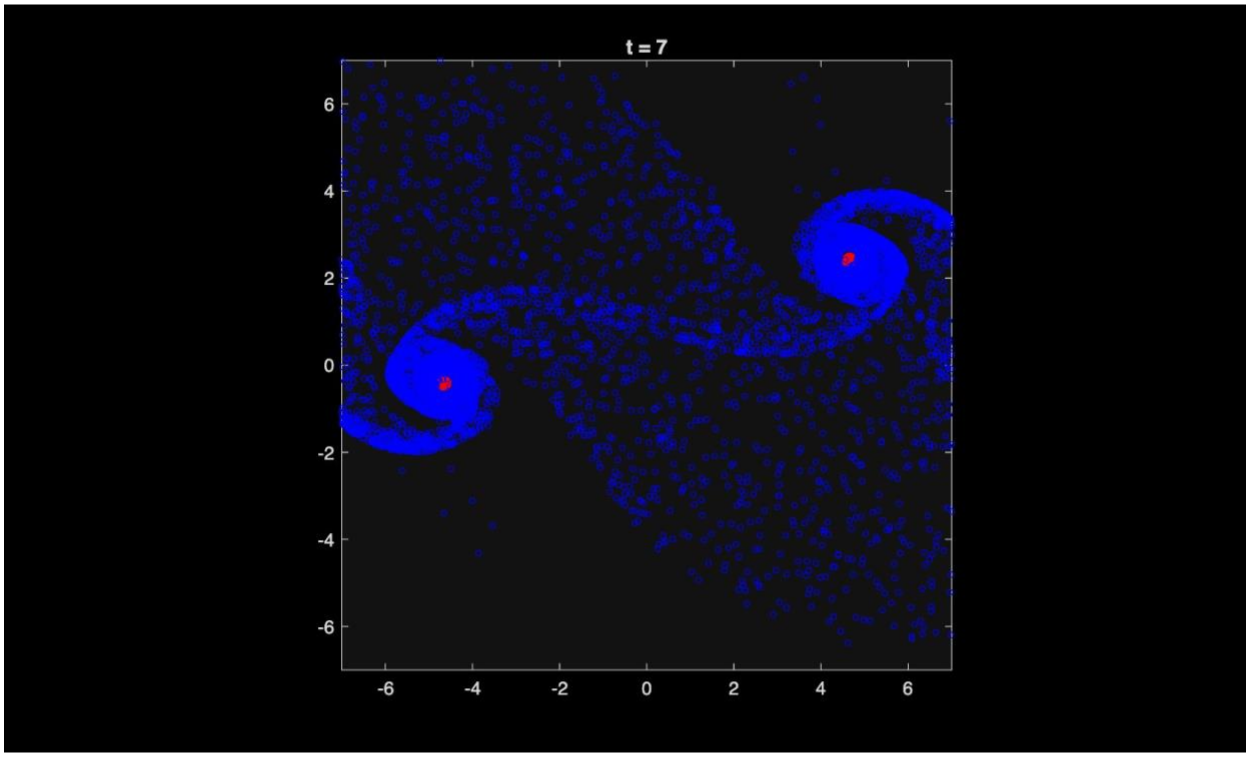

Figure 1. The positions of the cores and stars at time step 7 and time-resolution level 8 (after passing through each other). Some stars are seen to be slingshotted away due to the simulation restriction as they get too close to the core.

A convergence analysis was then performed by selecting core 1's position at time-resolution levels 8, 9, and 10. The position differences between levels 8 and 9, and between levels 9 and 10, were computed and plotted versus the time domain of the simulation in Figure 2 below.

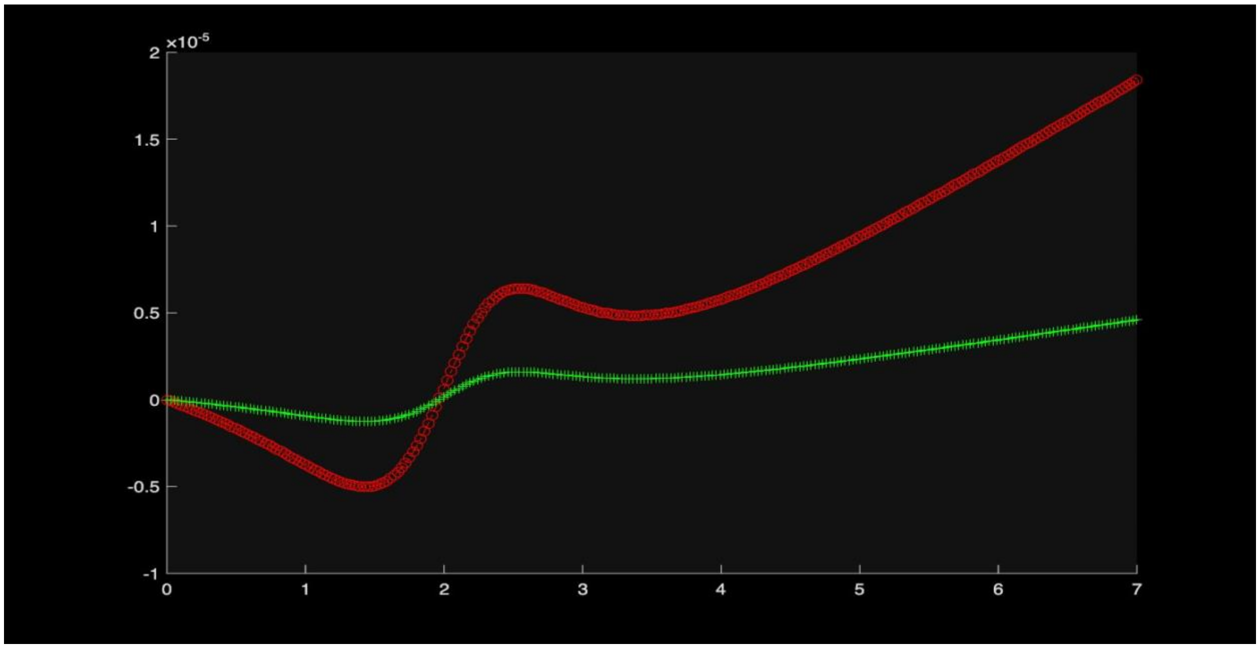

Figure 2. The differences between positions at certain time steps from level 8 vs 9 (red) and level 9 vs 10 (green) plotted versus the time domain of the simulation.

It is expected that the difference ratio between level 8 vs 9 and 9 vs 10 is 4. Thus, I multiplied the 9-versus-10 difference by 4 and plotted again. Indeed, both the red and green lines now almost exactly match, indicating that the system is consistent with theoretical expectations.

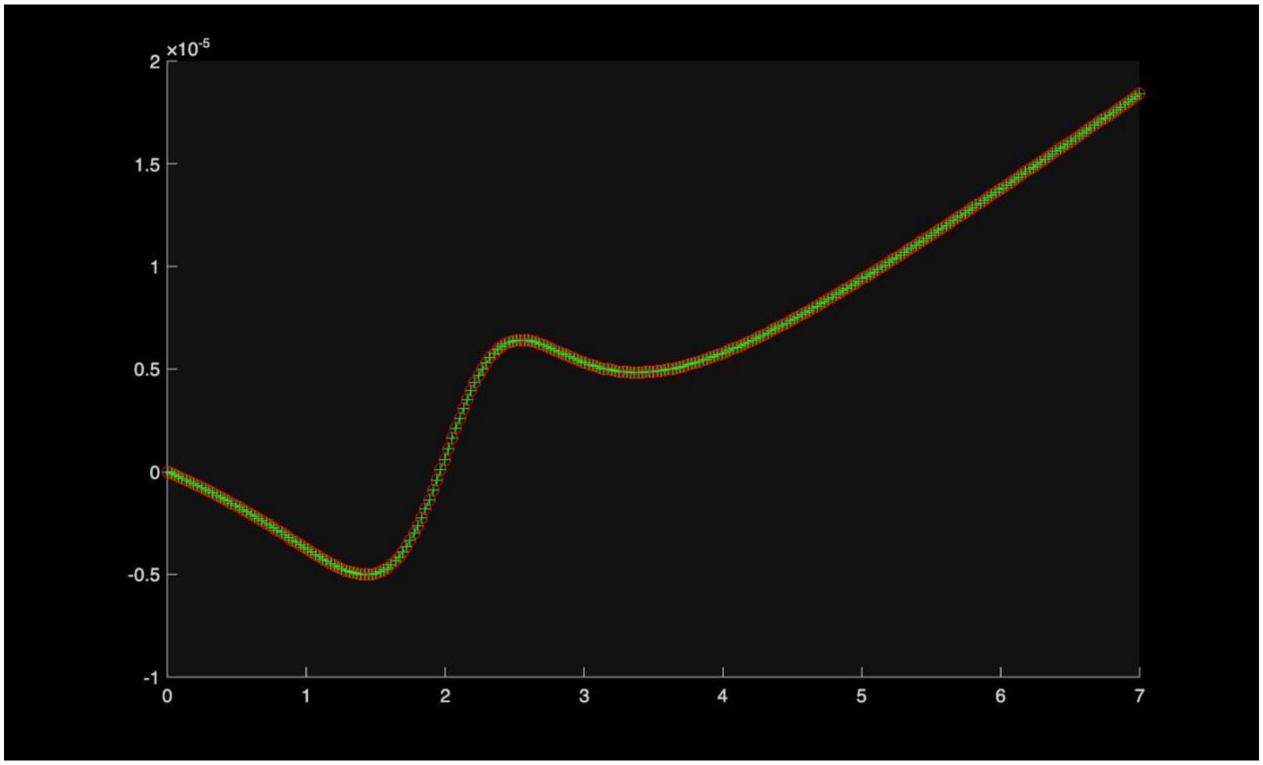

Figure 3. The differences between positions at certain time steps from level 8 vs 9 (red) and 4×(level 9 vs 10) (green) plotted versus the time domain of the simulation.

Discussion / Conclusions

The Toomre model simulation successfully modeled basic features of interacting galaxies. The finite-difference method provided accurate and stable motion for moderate time steps. The motion converged with increasing level, confirming second-order accuracy. This suggests that a simplified two-core system can capture many key behaviors seen in full N-body simulations, such as orbital deformation and tidal-tail formation.

The first challenge I encountered during the project was visualizing matrices in my head in a way that allowed me to implement them correctly in MATLAB. The N-body notes were very helpful in introducing the concept of multi-dimensional arrays. Another challenge was performing the convergence test for multi-dimensional arrays, as I initially thought I could perform the test for multiple objects or coordinates (x, y, z) simultaneously. With the help of generative AI—which suggested the use of the squeeze() function—I was eventually able to obtain the proper position array to perform the convergence test.

Statement on AI Use

I used generative AI to clarify MATLAB syntax, fix indexing issues, suggest useful functions, and help structure & fix grammar of this report. All simulation code, analysis, and results were produced independently.