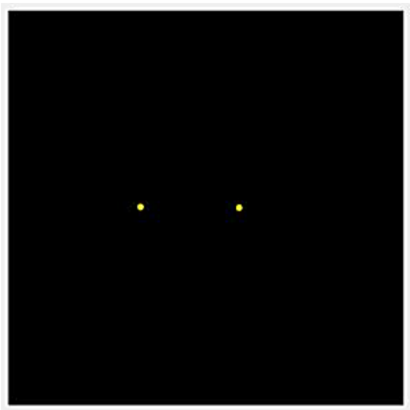
PHYS 410 Project 1

Table of Contents

This project is comprised of multiple scripts and functions used for simulating galactic collisions using a centered second order finite difference approximation.

Two Particle System: Convergence Test FDA



To approximate the trajectories of 2 particles in mutual orbit around another we setup an $O(\Delta t^2)$ finite difference approximation. The masses of the particles are $m_1 = 1$ and $m_2 = 0.5$ and they are separated by a distance of $r = 0.5$. These particles each require a certain initial velocity to ensure they have circular orbits around their center of mass. To determine each initial velocity we must first determine the distance of each particle from the center of mass (COM) of the system. We will assume the COM is located at the origin and that the two particles both lie along the x-axis. The equation for the location of the center of mass is given by

$$COM = \frac{m_1 r_1 + m_2 r_2}{m_1 + m_2} = 0.$$

We will also note that $r = r_1 + r_2$. Using these two equations we can derive an expression for r_1 and r_2.

$$m_1 r_1 = m_2 r_2 = m_2(r - r_1)$$

$$m_1 r_1 + m_2 r_1 = m_2 r$$

$$m r_1 = m_2 r$$

$$r_1 = \frac{m_2}{m} r$$

$\vec{r_1} = \frac{m_2}{m} r\hat{x}$

By using the same COM formula but instead substituting $r_1 = r - r_2$ we find that

$r_2 = \frac{m_1}{m} r$

$\vec{r_2} = -\frac{m_1}{m} r\hat{x}$

Now that we have the distances of each particle from the COM we can determine their initial velocities necessary for mutual orbit. We simply have to notice that the gravitational acceleration due to the other particle (with G=1) provides the centripetal acceleration about the COM. For m_1 we have

$\frac{m_2}{r^2} = \frac{v_1^2}{r_1}$

$v_1 = \sqrt{\frac{m_2 r_1}{r^2}} = \frac{\sqrt{m_2 r_1}}{r}$

$\vec{v_1} = \frac{\sqrt{m_2 r_1}}{r}\hat{y}$

Using the same method for v_2 we find that

$v_2 = \frac{\sqrt{m_1 r_2}}{r}$

$\vec{v_2} = -\frac{\sqrt{m_1 r_2}}{r}\hat{y}$

The mutual orbit simulation can now be performed using these initial conditions. We currently possess $r_1(0)$, $r_2(0)$, $v_1(0)$, and $v_2(0)$. For our second order FDA we require $r_1(\Delta t)$ and $r_2(\Delta t)$: the particle positions at the next iteration. These are determined using the kinematics equation

$r_{n+1} = r_n + \Delta t v_n + 0.5\Delta t^2 a_n$

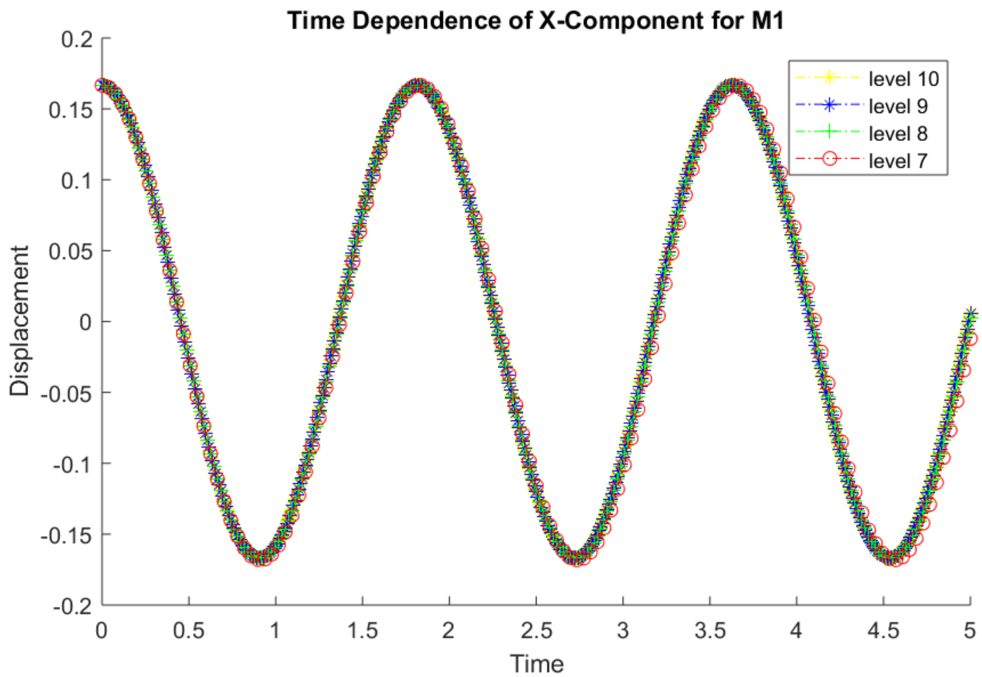
where a_n is the gravitational acceleration experienced by the particle which has been determined in the previous steps. For m_1: $\vec{a_n} = -\frac{m_2}{r^2}\hat{x}$ and for m_2: $\vec{a_n} = \frac{m_1}{r^2}\hat{x}$.

Using these first two positions we can now approximate the time evolution of this system using the centered second order FDA and setting it equal to the gravitational acceleration.

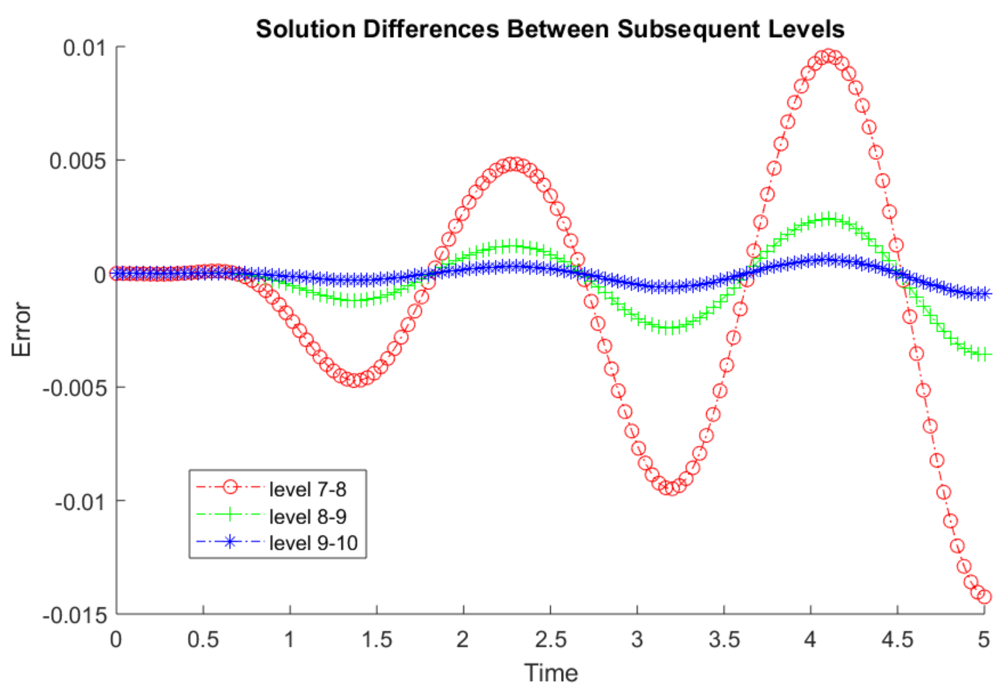
$\vec{a_n} = \frac{d^2\vec{r}}{dt^2}(t_n) \approx= \frac{\vec{r_{n+1}} - 2\vec{r_n} + \vec{r_{n-1}}}{\Delta t^2}$

$\vec{r_{n+1}} = 2\vec{r_n} - \vec{r_{n-1}} + \Delta t^2 \vec{a_n}$

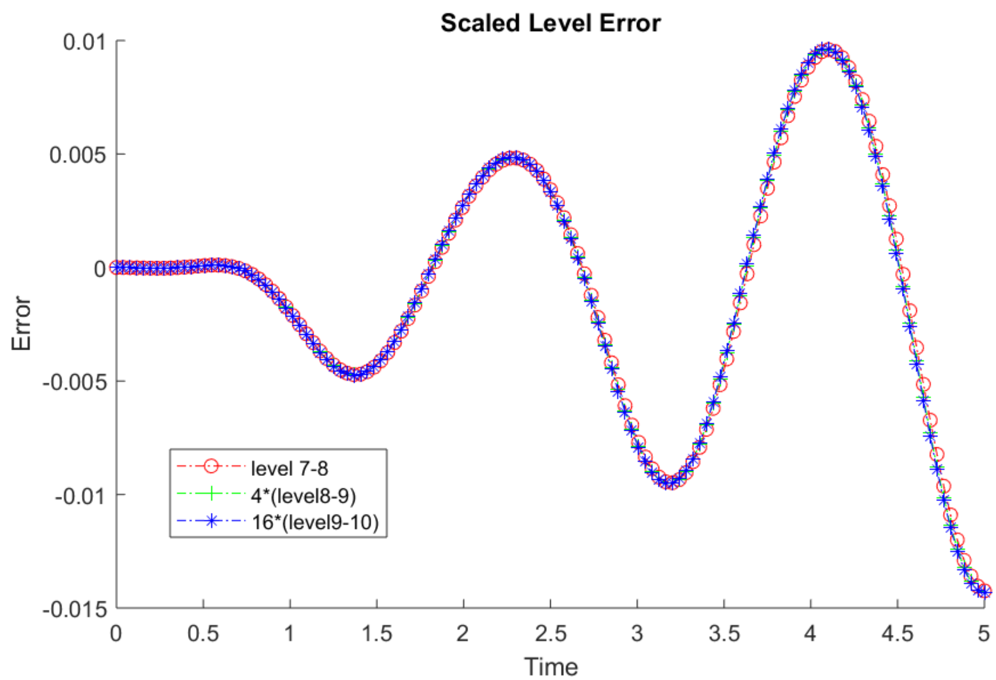
A convergence test was performed to check that there is $O(\Delta t^2)$ error in the x-component of the solution for mass m_1. Below are the plots of the trajectories of m_1 for $l = 7, 8, 9, 10$.



Next we reduce the sample sizes for $l = 8, 9, 10$ to only the times that $l = 7$ possess so that all solutions contain the same number of samples. This allows the datasets to be subtracted from another to obtain the error. Below is a plot of the difference between the x-component solutions for levels 7-8, 8-9, and 9-10.

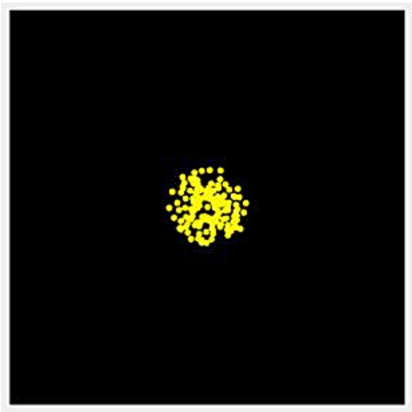


To prove that this FDA exhibits $O(\Delta t^2)$ convergence, we show that when Δt is divided by 2, the error is divided by 2^2. For level 7, Δt is double that for level 8, therefore its error should be 2^2 times greater than that for level 8. These two curves overlap when the error for level 8 is scaled by 2^2. As for level 9, its error will overlap with that of level 7 when scaled by 4^2, since Δt for level 7 is 4 times greater than that of level 9. Below is a plot of these scaled, overlapping curves which proves our $O(\Delta t^2)$ convergence.



Single Galaxy Dynamics

Before we test galaxy collisions, we ensure that the stars orbiting a single core will remain in orbit. This requires certain combinations of values for the minimum orbital radius, cores masses and discretization level. Stable orbits were obtained for $r_{min} = 0.03$, $m_{core} = 1$ and $l = 9$. The stars within a galaxy were intialized with a random distance from the core between $r_{min} = 0.03$ and $r_{max} = 0.2$ and a random angle θ for this radius. The radius and angle of each star was used to determine their initial velocity in the same manner as the two particle system mentioned earlier to achieve a stable orbit. To render a galaxy moving at a velocity $\vec{v}$, we simply add $\vec{v}$ to the initial velocity of each particle including the core. Below is a snapshot for a single galaxy at rest containing 150 stars.



Multi-Galaxy Dynamics

To simulate galaxy collisions I created a function called toomre() which takes the arguments: t_{max}, discretization level, core masses, core initial positions, core initial velocities, and the number of stars per core. The number of cores is inferred from the number of core masses provided. The initial positions and velocities of the cores are used to calculate the initial positions and velocities of the orbiting stars in the same manner as mentioned in the two particle system.

To determine subsequent positions, the toomre function calls another function called nbodyaccn to determine the accelerations of each particle due to the gravitational force of each core. The positions of each particle at the next iteration is calculated using the second order FDA shown below.

$$\vec{r_{n+1}} = 2\vec{r_n} - \vec{r_{n-1}} + \Delta t^2 \vec{a_n}$$

This calculation was repeated a total of 2^{level} times per particle to provide a total of $n_t = 2^{level} + 1$ data points per particle. As the function iterated over the time intervals, the locations of each particle were updated on the plot and saved to an AVI file.

The main difficulty I had with developing this code was generalizing the function to work with an arbitrary number of cores. To accomplish this I created a 4D matrix to deal with each galaxy separately while still being able to iterate over a single matrix. This made it easier to distinguish cores from stars without any messy indexing arithmetic.

I also created a script (collision.m) to run the toomre function with various initial conditions to find those that yielded the most interesting results. A few AVI files were generated for the most interesting scenarios.

With plotting disabled, the multi-galaxy simulation would complete in roughly 80 seconds. With plotting enabled, the simulation would complete in roughly 5 minutes. This introduced some difficulty with assessing certain initial conditions. I would need to wait longer to see how the system evolved.

Conclusions

The functions that I coded were observed to properly perform a second order finite difference approximation to simulate galactic collisions. The implementation was proven to possess $O(\Delta t^2)$ convergence and enabled stable galaxy configurations. The toomre model that was simulated was able to complete in under 2 minutes, indicating that the implementation was concise, efficient, and nonredundant.

Published with MATLAB® R2017b