PHYS 410 Project 1

Name: Steven Brown
Student Number: 90169772

1 Introduction

In this project the goal was to implement a very simplified model of galaxy collisions, motivated by work due to Alar and Juri Toomre in the early 1970's. This model is grossly inadequate for the detailed simulation of galaxy interactions but nevertheless capable of reproducing some basic morphological features that are observed in actual galactic collisions.

The model that will be studied in this project will be the Toomre Model. In the Toomre model, an isolated galaxy is modeled as a central particle (core particle) that has some gravitating mass, m_c. Orbiting this core is some number, n_s, of stars each with circular orbits of varying radii. The stars have vanishing gravitating mass which means they experience an acceleration due to the core's gravitational influence, but do not contribute to the gravitational field themselves. Therefore, the core acts on each of the stars, but the stars do not exert gravitational forces on either the core or the other stars.

To model a two galaxy collision we consider two cores (masses m_{c1} and m_{c2}) surrounded by n_{s1} and n_{s2} particles. The galaxies start with initial positions that ensure that the interaction of core 1 with the stars from galaxy 2 and vice versa is negligible. The galaxies will approach and interact in some fashion with one another. All of the stars feel the gravitational influence of both cores, and each core feels the gravitational influence of the other core. Therefore, any given particle will experience a gravitational force from either one or two core particles.

Since we are enforcing that the stars do not enter the force/acceleration calculation, if there are $O(N)$ stars, then the cost of the simulation is $O(N)$ vs. $O(N^2)$ which would be the case if all of the particles were gravitating.

Note that code snippets will be included in this report but to see the full code and function descriptions see the .m files submitted.

2 Review of Theory & Numerical Approach

2.1 Second Order Centered Finite Difference Approximation

We desire an $O(\Delta t^2)$ approximation of $f''(t)$ which can be done through a second order centered finite difference approximation. First use a Taylor Series to approximate $f(t+\Delta t)$. Then we can use a scaled combination of $f(t+\Delta t)$, $f(t)$ and $f(t-\Delta t)$ to get an approximate for $f''(t)$ and some $O(\Delta t^2)$ terms. This means that our approximation will be second order accurate.

$$f(t+\Delta t) = f(t) + \Delta t f'(t) + \frac{1}{2}\Delta t^2 f''(t) + \frac{1}{6}\Delta t^3 f'''(t) + \cdots$$

$$f(t-\Delta t) = f(t) - \Delta t f'(t) + \frac{1}{2}\Delta t^2 f''(t) - \frac{1}{6}\Delta t^3 f'''(t) + \cdots$$

Add these two equations together and rearrange to find the approximation for $f''(t)$ with $O(\Delta t^2)$ accuracy.

$$f(t+\Delta t) + f(t-\Delta t) = 2f(t) + \Delta t^2 f''(t) + \frac{1}{12}\Delta t^4 f''''(t) + \cdots$$

$$\frac{f(t+\Delta t) - 2f(t) + f(t-\Delta t)}{\Delta t^2} = f''(t) + O(\Delta t^2) \tag{1}$$

Now we can translate this result into grid spacing terminology. Grid spacing terminology lets us transform from the physical (continuim) domain with $0 \leq t \leq t_{max}$ to a uniform finite difference gird domain t_j. We define the number of grid points as n_t and the following equations.

$$f_{j+1} = f(t_j + \Delta t), \quad f_{j-1} = f(t_j - \Delta t), \quad f_{j+2} = f(t_j + 2\Delta t)$$

We then define the grid spacing to be:

$$\Delta t = \frac{t_{max}}{n_t - 1}$$

The democratization parameter is defined as level l where changes by a factor of 2 are most convenient.

$$\Delta t, \frac{\Delta t}{2}, \frac{\Delta t}{4} \rightarrow l, l+1, l+2$$

$$n_t = 2^l + 1 \rightarrow \Delta t = \frac{t_{max}}{2^l}$$

As l and n_t get larger, Δt gets smaller and the FDA should be more accurate. Therefore the discretized version of FDA equation (1) is:

$$f''(t) \approx \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta t^2} \tag{2}$$

2.2 The Gravitational N-Body Problem Formulation

2.2.1 Physical & Mathematical Formulation

First we define N point particles labelled by an index i with masses m_i to have position vectors $\mathbf{r}_i(t)$

$$\mathbf{r}_i(t) \equiv [x_i(t), y_i(t), z_i(t)], \quad i = 1, 2, \ldots, N$$

in a Cartesian coordinate system (x, y, z). Using the law of gravitation as well as Newton's second law, we can formulate the basic equations of motion in vector form.

$$m_i \mathbf{a}_i = G \sum_{j=1, j\neq i}^{N} \frac{m_i m_j}{r_{ij}^2} \hat{\mathbf{r}}_{ij} \quad i = 1, 2, \ldots, N \quad 0 \leq t \leq t_{max} \tag{3}$$

where $\mathbf{a}_i = \mathbf{a}_i(t)$ is the acceleration of the i-th particle, G is Newton's gravitational constant, and r_{ij} is the magnitude of the separation vector $\mathbf{r}_{ij}$ between i and j particles. Thus, we define the following:

$$\mathbf{r}_{ij} \equiv \mathbf{r}_j - \mathbf{r}_i, \quad r_{ij} \equiv |\mathbf{r}_j - \mathbf{r}_i| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}, \quad \hat{\mathbf{r}}_{ij} \equiv \frac{\mathbf{r}_j - \mathbf{r}_i}{r_{ij}}$$

It will be more convenient to eliminate the unit vector and to non-dimensionalize the system of equations, which in this case means choosing units in which G = 1 in equation (3) as follows:

$$m_i \mathbf{a}_i = \sum_{j=1, j\neq i}^{N} \frac{m_i m_j}{r_{ij}^3} \mathbf{r}_{ij} \quad i = 1, 2, \ldots, N \quad 0 \leq t \leq t_{max} \tag{4}$$

Additionally, we know

$$\mathbf{a}_i(t) = \frac{d^2\mathbf{r}(\mathbf{t})}{dt^2} \tag{5}$$

meaning we can combine equations (4) and (5) and cancel the m_i to get

$$\frac{d^2\mathbf{r}_i}{dt^2} = \sum_{j=1, j\neq i}^{N} \frac{m_j}{r_{ij}^3}\mathbf{r}_{ij} \quad i = 1, 2, \ldots, N \quad 0 \leq t \leq t_{max} \tag{6}$$

Equation (6) is a system of second order differential equations in time for the vector quantities $\mathbf{r}_i(t)$. Canceling the m_i allows us to use this equation for "mass-less" objects such as our stars. For a unique solution we will need initial position and velocity conditions for each particle. This will be specified with the following vectors.

$$\mathbf{r}_i(0) = \mathbf{r}_{0i}, \quad \mathbf{v}_i(0) = \frac{d\mathbf{r}}{dt}(0) = \mathbf{v}_{0i} \quad i = 1, 2, \ldots, N \tag{7}$$

2.2.2 N-Body Acceleration Implementation

The following code implements the nbodyaccn.m script. This code is based off equation (6) to calculate the acceleration of all particles due to core points. Note that since stars have vanishing gravitational mass, I only included the core masses in m. This means that the outer loop loops over all particles but the inner loop only loops over the first N particles, which are the core points (the position & velocity array's was purposely organized with core points first, then stars). Array slicing (using :) was also used for code cleanliness and computation speedup because they are faster than for loops in MATLAB.

```
% m: Vector of length N containing the core masses
% r: N + ns x 3 array containing the core + star positions
% a: N + ns x 3 array containing the computed core + star accelerations
function [a] = nbodyaccn(m, r)
    a = zeros(size(r, 1), 3);
    for i = 1 : size(r, 1)
        for j = 1 : length(m)
            if i == j
                continue
            end

            rij3 = norm((r(j,:) - r(i,:)))^3;

            a(i,:) = a(i,:) + m(j) * (r(j,:) - r(i,:)) / (rij3);
        end
    end
end
```

2.2.3 Solution Using Finite Difference Approximation

In the physical (continuum) domain of this problem we have $0 \leq t \leq t_{max}$ and will assume we can use a uniform time mesh to discretize. This means that the time at which we step the simulation will be constant. We define the following with a level parameter l similar to what was done in section 2.1. When running the simulation, l will be specified.

$$n_t = 2^l + 1, \quad \Delta t = \frac{t_{max}}{n_t - 1} = \frac{t_{max}}{2^l}, \quad t^n = (n-1)\Delta t, \quad n = 1, 2, \ldots, n_t$$

Next we define the following finite difference notation

$$\mathbf{r}_i^n \equiv \mathbf{r}_i(t^n)$$

and then combine it with equation (2) to get

$$\left.\frac{d^2\mathbf{r}(t)}{dt^2}\right|_{t=t^n} \approx \frac{\mathbf{r}^{n+1} - 2\mathbf{r}^n + \mathbf{r}^{n-1}}{\Delta t^2}$$

which we can then substitute into equation (6)

$$\frac{\mathbf{r}_i^{n+1} - 2\mathbf{r}_i^n + \mathbf{r}_i^{n-1}}{\Delta t^2} = \sum_{j=1, j\neq i}^{N} \frac{m_j}{(r_{ij}^n)^3}(\mathbf{r}_j^n - \mathbf{r}_i^n) \quad i = 1, 2, \ldots, N \quad 0 \leq t \leq t_{max} \quad n+1 = 3, 4, \ldots, n_t \tag{8}$$

We can then solve for the advanced time $\mathbf{r}_i^{n+1}$ because $\mathbf{r}_i^n$ and $\mathbf{r}_i^{n-1}$ are known from initial conditions and then previous time steps. The rearranged equation is the following

$$\mathbf{r}_i^{n+1} = \left(\Delta t^2 \sum_{j=1, j\neq i}^{N} \frac{m_j}{(r_{ij}^n)^3}(\mathbf{r}_j^n - \mathbf{r}_i^n)\right) + 2\mathbf{r}_i^n - \mathbf{r}_i^{n-1} \tag{9}$$

2.2.4 N-Body Position & Velocity Implementation

The following describes how the position was calculated for each particle. Note that the plot graphics section was removed to only show particle dynamics. This code was based off equation (9) and again array slicing was used. Linear extrapolation was used to calculate the velocity at the final time step.

```
for n = 2 : nt - 1
    % ---------- Graphics ----------
    ...
    % ---------- Dynamics ----------

    r(:,:,n+1) = dt^2 * nbodyaccn(m, r(:,:,n)) + 2*r(:,:,n) - r(:,:,n-1);
    v(:,:,n) = (r(:,:,n+1) - r(:,:,n-1)) / (2*dt);
end

% Use linear extrapolation to determine the value of v at the
% final time step.
v(:,:,nt) = 2 * v(:,:,nt-1) - v(:,:,nt-2);
```

2.2.5 Initial Conditions

Since we are using a three time level scheme, we need to determine values for $\mathbf{r}_i^1 = \mathbf{r}_i(0)$ and $\mathbf{r}_i^2 = \mathbf{r}_i(\Delta t)$. Note that $\mathbf{r}_i^1$ and $\mathbf{v}_i^1$ are given by the specified initial conditions. We need $\mathbf{r}_i^2$ to be calculated to $O(\Delta t^3)$ accuracy so that the overall solution is $O(\Delta t^2)$. This is because $n_t \sim \Delta t^{-1} = O(\Delta t^{-1})$ meaning that per time step, the error needs to be $O(\Delta t^3)$ in order for the overall solution to be $O(\Delta t^2)$. We can use Taylor Series to calculate $\mathbf{r}_i^2$ as follows

$$\mathbf{r}_i(\Delta t) = \mathbf{r}_i(0) + \Delta t \frac{d\mathbf{r}_i}{dt}(0) + \frac{1}{2}\Delta t^2 \frac{d^2\mathbf{r}_i}{dt^2}(0) + O(\Delta t^3)$$

and now substituting initial velocity and equation of motion in we get

$$\mathbf{r}_i^2 \approx \mathbf{r}_i^1 + \Delta t \mathbf{v}_i^1 + \frac{1}{2}\Delta t^2 \sum_{j=1, j\neq i}^{N} \frac{m_j}{(r_{ij}^1)^3}(\mathbf{r}_j^1 - \mathbf{r}_i^1) \tag{10}$$

2.2.6 Initial Conditions Implementation

The following code shows how the initial conditions were implemented. Note that v0 and r0 are defined and we use equation (10) to find the position at the second time step.

```
% Initial Conditions r^1
r(1:N,:,1) = r0;
v(1:N,:,1) = v0;

% Calculate r^2
r(:, :, 2) = r(:,:,1) + dt*v(:,:,1) + dt^2 * nbodyaccn(m, r(:,:,1)) / 2.0;
```

2.3 Suggested Test Case Derivation

A good, non-trivial configuration that was used to develop and test the FDA implementation describes two particles with arbitrary masses in mutual circular orbit about their center of mass, and in the x-y plane. Let the particle masses be m_1 and m_2, respectively, and let the particles be separated by a distance r. Let the initial position and velocity vectors be

$$\mathbf{r}_1(0) = (r_1, 0, 0)$$
$$\mathbf{r}_2(0) = (-r_2, 0, 0)$$
$$\mathbf{v}_1(0) = (0, v_1, 0)$$
$$\mathbf{v}_2(0) = (0, -v_2, 0)$$

where r_1, r_2, v_1 and v_2 are all positive quantities, so that the particle separation is given by $r = r_1 + r_2$. The initial position vector components were selected so that the two particles start on the horizontal axis. The initial velocity vector components were selected to be purely tangential to the circular orbit. Additionally, let $m = m_1 + m_2$. In calculating the CM between two spheres you can assume each has its mass concentrated at its geometric center. The center of mass between the spheres is then a point that is a ratio of the separations and masses of the objects.

$$m_1 r_1 = m_2 r_2$$

then rearrange for r_1 and r_2 and sub in $r = r_1 + r_2$

$$r_1 = \frac{m_2}{m_1}(r - r_1) \rightarrow r_1 = \frac{m_2 r}{m_1\left(1 + \frac{m_2}{m_1}\right)} = \boxed{\frac{m_2}{m} r}$$

$$r_2 = \frac{m_1}{m_2}(r - r_2) \rightarrow r_2 = \frac{m_1 r}{m_2\left(1 + \frac{m_1}{m_2}\right)} = \boxed{\frac{m_1}{m} r}$$

The centrifugal inertial force on each particle causes its circle of travel. Note that the velocity of both particles is purely tangential to the circular orbit.

$$F_1 = m_1 \frac{v_1^2}{r_1} \quad F_2 = m_2 \frac{v_2^2}{r_2}$$

The centrifugal force equals the gravitational force (remember we set G = 1) for a circular orbit so we can solve for the velocity.

$$F_1 = F_{g1} \rightarrow m_1 \frac{v_1^2}{r_1} = \frac{m_1 m_2}{r^2} \rightarrow v_1 = \boxed{\frac{\sqrt{m_2 r_1}}{r}}$$

$$F_2 = F_{g2} \rightarrow m_2 \frac{v_2^2}{r_2} = \frac{m_2 m_1}{r^2} \rightarrow v_2 = \boxed{\frac{\sqrt{m_1 r_2}}{r}}$$

2.4 Suggested Test Case Implementation

The following code shows how the suggested test case was implemented based off section 2.3.

```
% ICs for mutual circular orbit
tmax = 140;
r = 4;
mc1 = 1;
mc2 = 0.5;
r1 = mc2 * r / (mc2 + mc1);
r2 = mc1 * r / (mc2 + mc1);
v1 = sqrt(mc2 * r1) / r;
v2 = sqrt(mc1 * r2) / r;

% Get position values at levels 6
[t6 r6 v6] = galaxy(2, tmax, [mc1 mc2], 6, [r1 0 0; -r2 0 0], [0 v1 0; 0 -v2 0], 0);
```

2.5 FDA Convergence Test

We want to examine the behavior of the solution as $\Delta t \rightarrow 0$. Let $u_*(t)$ be the exact (continuum) solution of some differential equation then the error is

$$e(t^n) = u_*(t^n) - u(t^n)$$

where $u(t^n)$ is computed. Then we can also state

$$\lim_{\Delta t \rightarrow \infty} e(t^n) = \Delta t^2 e_2(t^n) + O(\Delta t^4) \tag{11}$$

where $e_2(t^n)$ is some function and the $O(\Delta t^4)$ is due to the FDA being centered. We use this as an assumption for convergence analysis.

We always want to use at least 3 meshes (values of l) to then form the following from (11)

$$u_l^n \approx u_*^n - (\Delta t_l)^2 e_2^n, \quad u_{l+1}^n \approx u_*^n - (\Delta t_{l+1})^2 e_2^n, \quad u_{l+2}^n \approx u_*^n - (\Delta t_{l+2})^2 e_2^n$$

We can think of n labelling common set of times. We then subtract the solution on adjacent levels

$$u_l^n - u_{l+1}^n = -((\Delta t_l)^2 - (\Delta t_{l+1})^2) e_2^n = \frac{-3}{4} \Delta t_l^2 e_2^n$$

$$u_{l+1}^n - u_{l+2}^n \approx \frac{-3}{4} (\Delta t_{l+1})^2 e_2^n = \frac{-3}{16} \Delta t_l^2 e_2^n$$

We can see that the error was reduced by a factor of 4. We can then plot these differences on a single plot as a function of t^n, scaling the lower level difference by the following ratio (in our simulation this will be a factor of 4):

$$\frac{u_l^n - u_{l+1}^n}{u_{l+1}^n - u_{l+2}^n}$$

The curves should be nearly coincident and alignment should get better for higher levels.

2.6 FDA Convergence Test Implementation

The following is the function that will calculate the convergence test and plot the three graphs. The code was derived from section 2.5.

```
% Convergence test of basic finite difference solution using two
% particles (cores) with distinct masses (say 1.0 and 0.5) in
% mutual circular orbit about each other.
function convtest()
    tmax = 140;
    plot1en = 1;
    plot2en = 1;
    plot3en = 1;

    % ICs for mutual circular orbit
    % See section 2.4 for removed initial condition calculation code
    % Get position values at levels 6, 7, 8
    [t6 r6 v6] = galaxy(2, tmax, [mc1 mc2], 6, [r1 0 0; -r2 0 0], [0 v1 0; 0 -v2 0], 0);
    [t7 r7 v7] = galaxy(2, tmax, [mc1 mc2], 7, [r1 0 0; -r2 0 0], [0 v1 0; 0 -v2 0], 0);
    [t8 r8 v8] = galaxy(2, tmax, [mc1 mc2], 8, [r1 0 0; -r2 0 0], [0 v1 0; 0 -v2 0], 0);

    % Reshape vectors to only consider one particle's x dimension
    r6 = reshape(r6(1,1,:), [1, length(t6)]);
    r7 = reshape(r7(1,1,:), [1, length(t7)]);
    r8 = reshape(r8(1,1,:), [1, length(t8)]);

    if plot1en
        % Removed all non-plot lines for clarity
        plot(t6, r6, 'r-.o');
        plot(t7, r7, 'g-.+');
        plot(t8, r8, 'b-.*');
    end

    % Downsample level 7 & 8 to length of level 6
    r7 = r7(1:2:end);
    r8 = r8(1:4:end);
    % Compute differences of grid functions between levels
    r67 = r6 - r7;
    r78 = r7 - r8;

    if plot2en
        % Removed all non-plot lines for clarity
        plot(t6, r67, 'r-.o');
        plot(t6, r78, 'g-.+');
```

```
    end

    % Scale level 7,8 difference
    r78 = r78 * 4;

    if plot3en
        % Removed all non-plot lines for clarity
        plot(t6, r67, 'r-.o');
        plot(t6, r78, 'g-.+');
    end
end
```

2.7 Adding Stars

As stated in the introduction, orbiting this core, in circular orbits of varying radii, are some number of stars. For each galaxy, we want to restrict the stellar radii about the core to some minimum and maximum values, and, for any given radii, distribute the star's angular position randomly. This will yield a more natural appearance for the galaxies at early times. Since we are assuming the starts have vanishing gravitating mass, the circular motion that is about the center of mass of the star and its core will be at the cores position. This allows us to assume $r_1 = 0$ (the separation distance is the radius of the star) from section 2.3. Thus we define the following:

$$\mathbf{r}_s(0) = (r_s cos(\theta), r_s sin(\theta), 0)$$

$$\mathbf{v}_s(0) = (-v_s sin(\theta), v_s cos(\theta), 0)$$

where r_s and v_s are the magnitude of the stars initial radii and velocity respectively and θ is its angle from the positive x-axis. The initial velocity is found by using equations from section 2.3. Note that the direction of circular motion can be switched by switching the sign of the velocity components. In order to achieve the random distribution of stars around the core point we generate r_s and θ randomly between some maximum and minimum values. For θ we choose 0 and 2π.

2.8 Adding Stars Implementation

The only code that needed to be added to the core particle implementation was the handling of random radii and angular position generation for the initial conditions. Note that the first N positions in r and v were set to be for the core points and then the rest of positions were for the stars. We also know there are ns stars per core point. An arbitrary maximum and minimum radii was selected and then a random point between the two was calculated.

```
nostars = ns == 0;
if ~nostars
    % ICs for mutual circular orbit
    secstart = N+1;
    secend = N+ns;
    for i = 1 : N
        rmin = 1.5;
        rmax = 3;
        rs = rmin + (rmax - rmin) * rand(ns, 1);
        theta = 2 * pi * rand(ns, 1);
        vs = sqrt(m(i) * rs) ./ rs;

        rx = rs .* cos(theta);
```

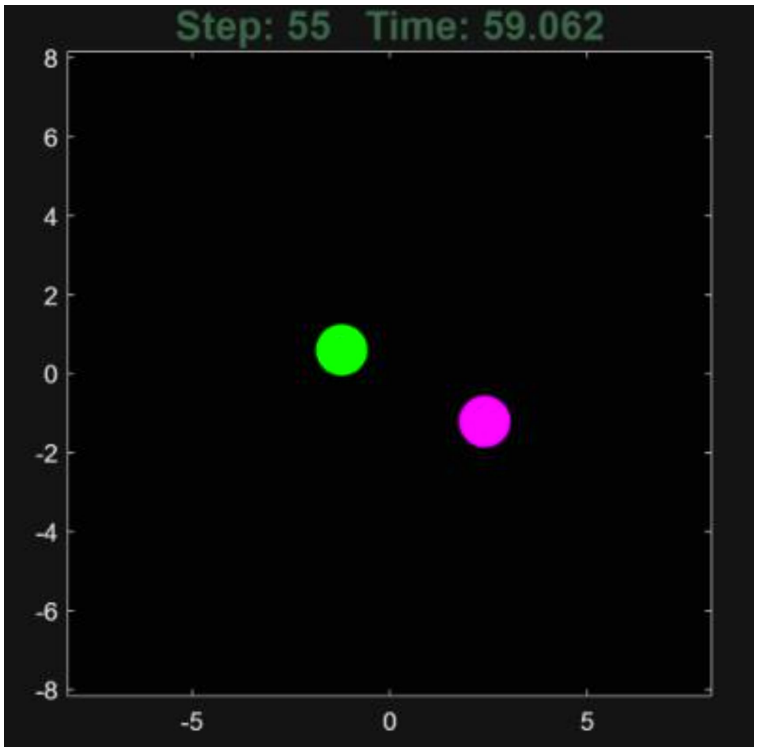
```
        ry = rs .* sin(theta);
        vx = -vs .* sin(theta);
        vy = vs .* cos(theta);

        r(secstart:secend,:,1) = [rx ry zeros(ns, 1)] + r0(i,:);
        v(secstart:secend,:,1) = [vx vy zeros(ns, 1)] + v0(i,:);

        secstart = secstart + ns;
        secend = secend + ns;
    end
end
```

3 Results

3.1 FDA Convergence Test

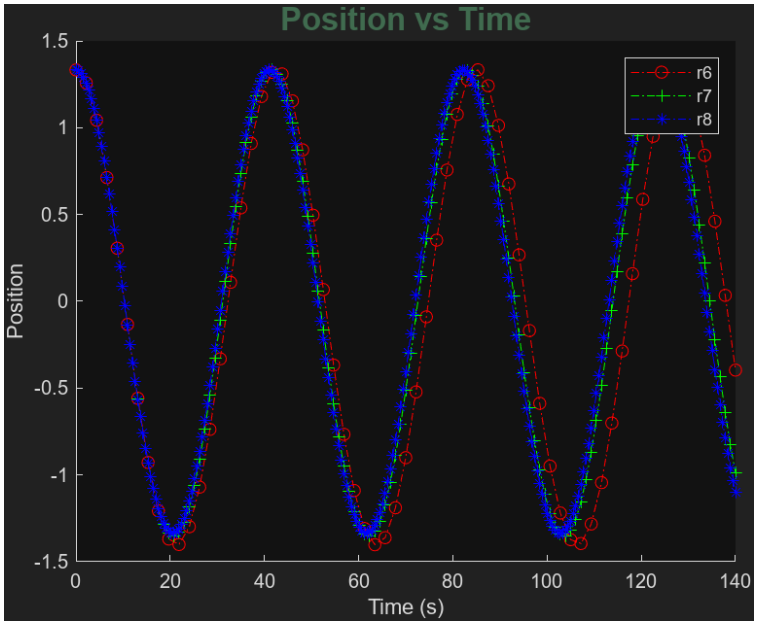
Before we begin studying galactic collisions, we convergence tested our basic finite difference solution using two particles (cores) with distinct masses (say 1.0 and 0.5) in mutual circular orbit about each other. We chose the x-coordinate from one of the particles and demonstrated using at least a three-level convergence test that our implementation appears to have $O(\Delta t^2)$ error.

As a stringent test that our numerical solution is behaving as it should, implying that the implementation is correct, we performed a 3-level (6, 7, and 8) convergence test for our FDA as discussed in class following the steps discussed in section 2.5. Shown below is the initial position for the two core points that are orbiting each-other in a circular orbit. See "galaxy_l7_convtest.avi" for a video of the orbit.

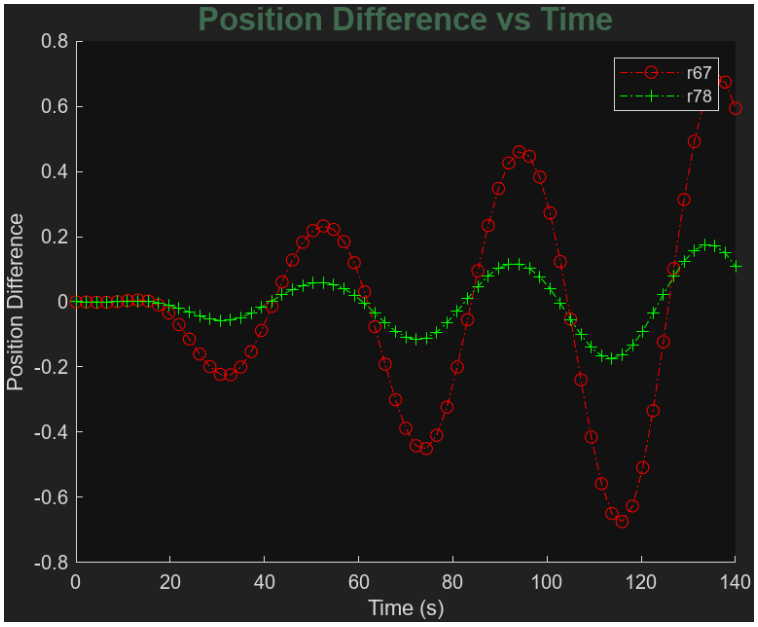


First, the x-position of each level simulation was plotted versus time seen below. Note that the curves show general agreement, but that as time progresses there is an increasing amount of deviation, which is most

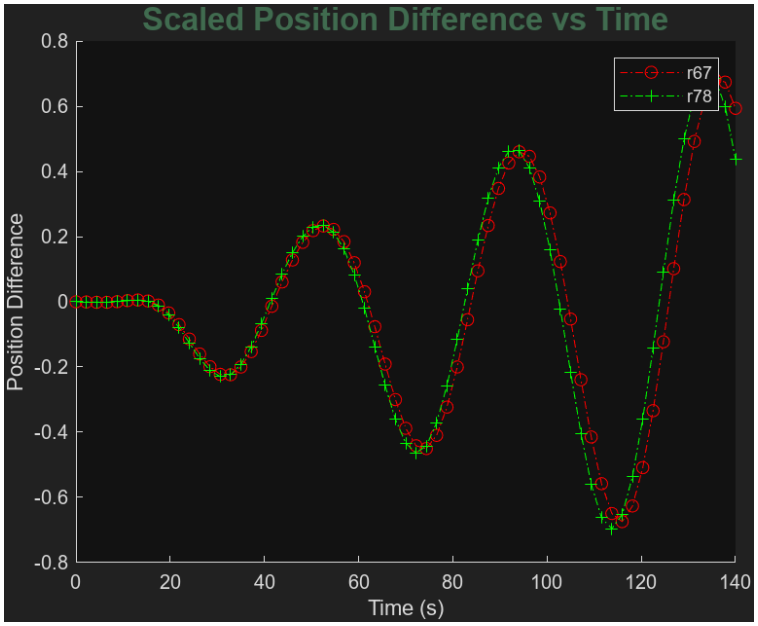
pronounced for the calculation that used the largest grid spacing (level 6, red line). The observed deviations are a result of the approximate nature of the finite difference scheme.



Then we computed the differences in the grid functions from level to level and plotted them on a position difference versus time graph shown below. Note that the two curves have the same shape, but different overall amplitudes where each curve is a direct measure of the error in the numerical solution.



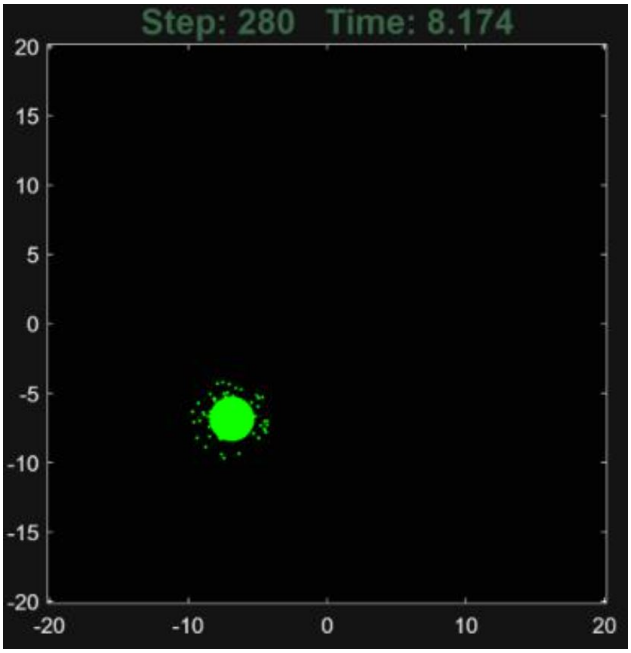
Lastly, we scale r78 by 4 and re-plot together with r67. If the convergence is second order, the curves should be nearly coincident (the grid spacing goes down by a factor of 2, so the differences should go down by a factor of 4). We can see this is true in the plot below. Also note how the magnitude of the error increases with time, which is characteristic of a finite difference solution of this type.



If we wanted to complete a more detailed convergence test we could extend our convergence test by repeating the calculation with an even higher level, for example level = 9.

3.2 Single Galaxy Dynamics

With the convergence tested, we then started with a single galaxy at rest and verified that, when evolved, the stars remain on circular orbits about the core. We then simulated a galaxy which is moving with some velocity. Note that you can translate a galaxy, or give it some overall velocity, simply by adding the translation or velocity vector to the position or velocity, respectively, of the core and each of the stars. We can see from the image below that the stars don't fly away from the core. To see the video check out "single_galaxy_check.avi".



3.3 Galaxy Collisions

It's now time to consider galaxy collisions. We begin with interactions where all particles are initially in some plane and remain in that plane (the xy plane), and all of the work has the dynamics restricted to this plane. The goal here was to perform some simulations which generate "interesting" morphologies, such as the example shown in the movie on the course homework page. Note that these collisions should tend to be "glancing" because the model will cease to be very meaningful if the cores get too close to one another. Different experiments were conducted with different initial conditions to get interesting collision results. It is important to note that the direction of rotation of the stars about a core relative to the overall angular momentum of the collision will have an impact on the collision morphology. Thus to change that direction of the starts orbit all that was needed to be done was flip the sign for initial velocity in section 2.7 equations.

Note in all these simulations, the mass of both cores is 0.5, there were 10,000 stars in total (5000 per galaxy) and the simulation was ran for 2500 time steps with different maximum times.

The first simulation conducted was for a collision where stars from each others cores were swapped and started to orbit the other core. No stars were shot off into infinity but some were left behind in the center of the collision while the core continued to drift away in opposite directions. In this simulation $t_{max} = 60s$, $r_{1,0} = [-3.2, -3.8, 0]$, $r_{2,0} = [3.2, 3.8, 0]$, $v_{1,0} = [0.3, 0, 0]$, $v_{2,0} = [-0.3, 0, 0]$. See Figure 1 below or the "no_lost_stars.avi" video.

The second simulation conducted was for a collision where the star orbits start to unravel and get lost in the center between the two cores as they move away from each-other. In this simulation $t_{max} = 40s$, $r_{1,0} = [-3.5, -3.5, 0]$, $r_{2,0} = [3.5, 3.5, 0]$, $v_{1,0} = [0.4, 0, 0]$, $v_{2,0} = [-0.4, 0, 0]$. See Figure 2 below or the "unravel.avi" video.

The last simulation conducted was for a combination of the previous two situation but in addition, some stars shoot off to infinity and escape the orbits. Some of the stars shoot off to infinity because they get too close to a core and the FDA breaks down. In this simulation $t_{max} = 140s$, $r_{1,0} = [-4, -4, 0]$, $r_{2,0} = [4, 4, 0]$, $v_{1,0} = [0.2, -0.05, 0]$, $v_{2,0} = [-0.2, 0.05, 0]$. See Figure 3 below or the "stars_shoot_off.avi" video.

After examining these simulation it was observed that initial conditions have a large effect on the result of the simulation. Note that for each of these simulation with level parameter 11, lower level parameters were also tested and provided a less accurate simulation with slightly different results (mostly more stars shooting off to infinity after the collision). It was also noticed that the larger the initial velocity of the cores, the more stars were left behind in the middle as in the unravel simulation. This is due to the cores strong gravitational attraction but also the fact that they are moving away from each-other quickly so the stars can get bumped out of orbit and get left behind. Note that the velocities were chosen empirically to try to target glancing collisions. In the figures below, as we go from left to right time has passed.

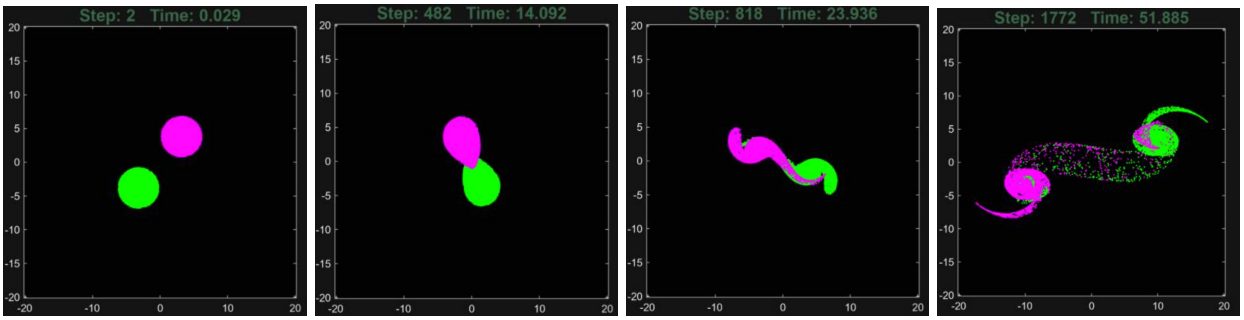


Figure 1: Stars Switch Orbit

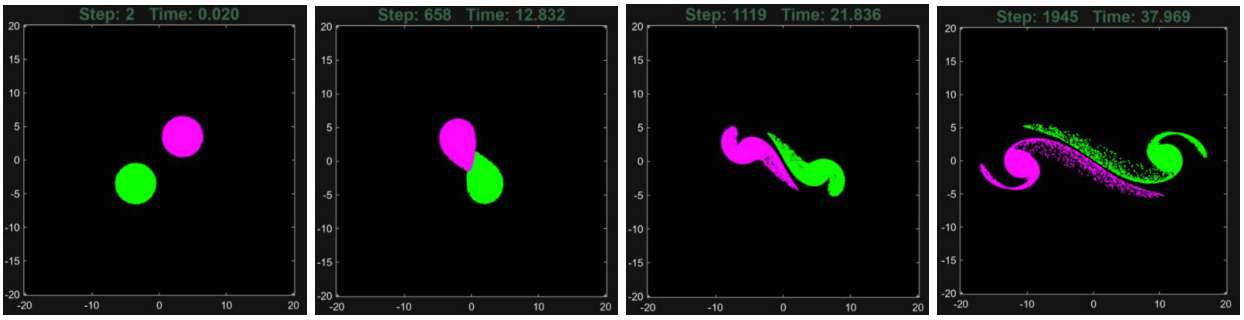


Figure 2: Stars Unravel Orbit

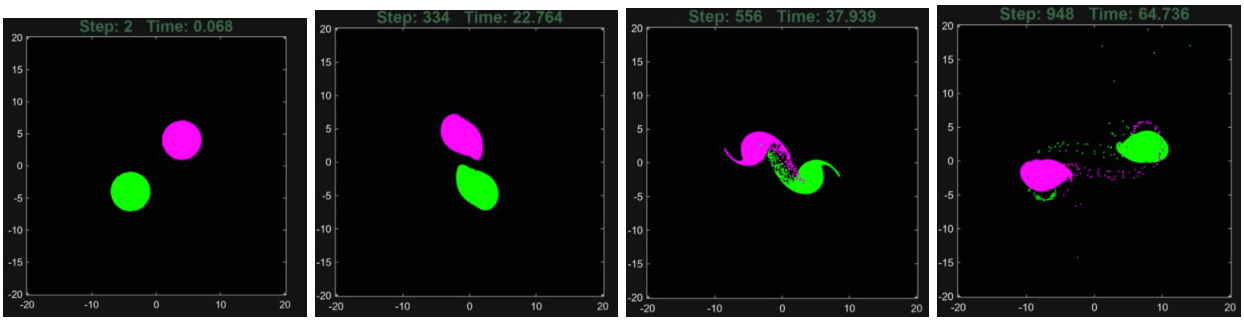


Figure 3: Stars Escape Orbit

4 Discussion & Conclusion

In this project the goal was to implement a very simplified model of galaxy collisions, motivated by work due to Alar and Juri Toomre in the early 1970's. To model a two galaxy collision we considered two cores (masses m_{c1} and m_{c2}) surrounded by n_{s1} and n_{s2} particles. The galaxies started with initial positions that ensured that the interaction of core 1 with the stars from galaxy 2 and vice versa were negligible. The galaxies approached and interacted in different fashions with one another. We found a $O(\Delta t^2)$ approximation of $f''(t)$ which was done through a second order centered finite difference approximation and Taylor Series. We then translated this result into grid spacing terminology and derived the physics and math for an n-body problem. Then, we found the initial conditions for circular orbits for the stars around their cores. Next, we convergence tested our finite difference approximation and found that the curves were nearly coincident and that the magnitude of the error increases with time, which is characteristic of a finite difference solution of this type.

We then moved onto galaxy collision where three simulation were conducted. Simulations with stars switch-

ing, star unraveling, and star escaping orbits were obtained. We saw a large effect on the result of the simulation from slightly varying initial conditions. Simulation differences were also observed by changing the level parameter. Lastly, it was also noticed that the larger the initial velocity of the cores, the more stars were left behind in the middle as in the unravel simulation.

I did not have any problems with implementing the discrete equations of motion or anything else in this project besides learning MATLAB syntax. I did not use generative AI.