Draft version August 27, 2024
Typeset using LaTeX default style in AASTeX631

ASTR660 Final Project: A Journey into Collisional N-Body Simulations

LI, WEI-WEN

(Dated: December 2023)

1. OUTLINE

The original purpose of this project was to evaluate the tidal effect on the evolution of a globular clusters. But latter, as the teacher pointed out, I realized that the collision treatment was indeed a hard task, and I put most of my effort on it. Eventually I gave up understanding all the underlying mechanisms, but I did learn a lot on what must consider to implement a n-body simulation. The main objective of this final project therefore turn to understanding the factor, including dynamical friction, relaxation and binaries, that effect the core collapse of a cluster.

In the following section, I will first introduce the the model of the globular cluster, and how I generate the initial condition. Then, I will show the algorithms of conducting a collisional n-body simulation, and point out some key consideration. Finally, I will show the results and conclusion of the simulations.

This project relies on AI assistance to examine grammatical errors, and also bugs in the program.

2. MODEL DESCRIPTION AND INITIAL CONDITIONS

To be clarified first, the units I'm using follow the normal choice of n-body code, the Henon's unit, which all parameters are normalized as

$$M = R = G = 1 \tag{1}$$

Where M is the total mass, R is the virial radius and G is the gravitational constants.

We utilize Plummer's model (Plummer 1911) as a initial condition for our globular cluster, where the density distribution is given as

$$\rho(r) = \frac{3M}{4\pi r_{pl}^3 (1 + r^2/r_{pl}^2)^{5/2}} \tag{2}$$

and the potential distribution is

$$\Phi(r) = -\frac{GM}{r_{pl}(1 + r^2/r_{pl}^2)^{1/2}} \tag{3}$$

where $r_{pl} = \frac{3\pi}{16}$ is the Plummer radius, defined as the cluster core

For the initial mass function, we use the model given in (Kroupa 2001)

The following subsections show how we derive the generating functions from specific models.

2.1. *Position and Velocity Distribution*

The following derivation are mostly from (Aarseth et al. 2008)

To generate random distribution under these constrains, we created several generating functions. First, using 2, we formulate the accumulate mass with respect to r.

$$M(r) = \int_0^r \rho(r) 4\pi r^2 dr = M_{cl} \frac{(r/r_{pl})^3}{[1 + (r/r_{pl})^2]^{3/2}}$$

Thus we obtain our first random variable

$$X_1(r) = \frac{M(r)}{M_{cl}} = \frac{\zeta^3}{1 + \zeta^2} \text{ where } \zeta \equiv r/r_{pl} \text{ and } X_1(\infty) = 1$$

To derive the generating function, we simply construct the inverse funciton

$$\zeta(X_1) = (X_1^{-2/3} - 1)^{-1/2} \tag{4}$$

From ζ, we may obtain the distance of star with respect to the core by

$$r^2 = (\zeta r_{pl})^2 = x^2 + y^2 + z^2 \tag{5}$$

We can use this to further derive the coordination of every single stars. Skipping the derivation, we get

$$z(X_2) = 2rX_2 - r \tag{6}$$

For x and y, we may assume a they distributed uniformly with respcet to an angle θ. Thus we have

$$\theta(X_3) = 2\pi X_3 \tag{7}$$

$$x = (r^2 - z^2)^{1/2} cos\theta,\ y = (r^2 - z^2)^{1/2} sin\theta \tag{8}$$

With 7, and 5, we obtained the distance from core, and with 6, 7 and 8, we get the coordination of the star.

Next we go on derive the velocity distribution, we first find the escape velocity. From Plummer's model, we have the isotropic distribution funciton

$$f_m(\epsilon_e) = \frac{24\sqrt{2}}{2\pi^3} \frac{r_{pl}^2}{(GM_{cl}^5)} (-\epsilon_e)^{7/2},\ \epsilon_e \leq 0$$

Skipping the complex derivation, we have

$$v_{esc}(r) = \sqrt{-2\Phi(r)} \equiv v/\zeta$$

And then skipping even more, we have

$$X_4(\zeta) = \frac{1}{2}(5\zeta^3 - 3\zeta^5) \tag{9}$$

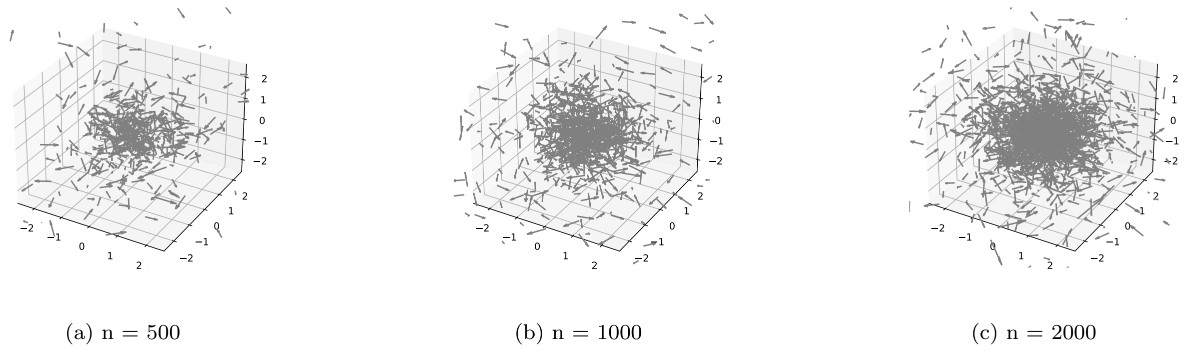
Problematically, this is not a invertible function. Therefore, I use the `fsolve` function from `scipy.optimize` to find the root.

With $v = \zeta v_{esc}(r)$, we can simply obtained the velocity as follows

$$v_z(X5) = (2X_5 - 1),\ \theta(X_6) = 2\pi X_6$$

$$v_x = \sqrt{v^2 - v_z^2} cos\theta,\ v_y = \sqrt{v^2 - v^z} sin\theta$$

For sample plots, see Figure 1



(a) n = 500 (b) n = 1000 (c) n = 2000

Figure 1. (a), (b) and (c) are example of globular clusters generated by Plummer's model with different n

To verify whether the model follows Virial theorem, I compute the potential energy and kinetic and found out that it fit well when increassing the number of particles. See Figure 2.

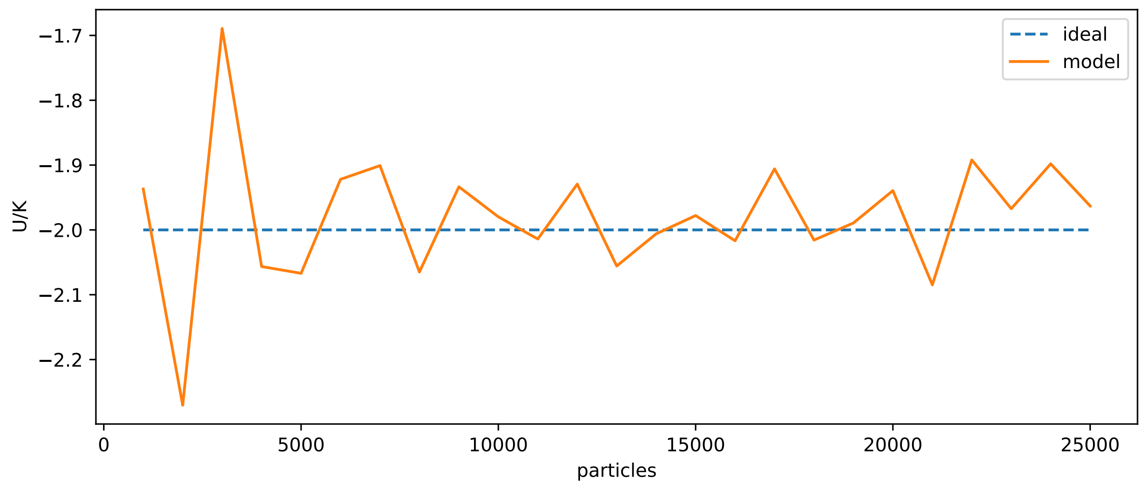


Figure 2. Testing Virial theorem with different amount of stars

2.2. *Initial Mass Function*

There is a little bug with the Plummer's model since it consider all stellar object to be in the same mass. Yet here, we still defined an IMF based on Kroupa's broken power law as follows.

$$\xi(m) \propto m^{-\alpha}$$

$$\alpha_0 = 0.3 \text{ for } 0.01 \leq m \leq 0.08 \tag{10}$$

$$\alpha_1 = 1.3 \text{ for } 0.08 \leq m \leq 0.5 \tag{11}$$

$$\alpha_2 = 2.3 \text{ for } 0.5 \leq m \leq 1 \tag{12}$$

$$\alpha_3 = 2.7 \text{ for } m > 1 \tag{13}$$

Solving the condition of $\int_{0.01}^{10} \epsilon(m)dm = 1$, while making sure the boundary of the four equations are continuous, we get the initial mass distribution function. Inverting the function helps us find the generating function of the initial mass for each star. At first, I thought the error within solving the equation was too large, so planned to build a table connecting the random variable X and the respective mass m. Then, I may interpolate the X's to get the m for actual implementation. But, fortunately, the mass distribution computed analytically by myself worked out pretty well, see Figure 3

Note that since we're using Henon's unit, we'll normalize the mass, such that the total mass $M = 1$. Additionally, we assume that the mass distribution is uniform with large N, so that it wouldn't violates Plummer's distribution.

3. ALGORITHMS AND DATA STRUCTURES

To implement collisional N-body code, though inefficient, I implement direct n-body with additional collision treatment.

3.1. *Non-collisional Treatment*

The derivation here is mostly from Heggie's lecture slide (2003).

We chose Hermite integrator as the integrating method. Which is saying

$$x(t + \Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a\Delta t^2 + \frac{1}{6}\dot{a}\Delta t^3$$

$$v(t + \Delta t) = v(t) + a(t)\Delta t + \frac{1}{2}\dot{a}\Delta t^2$$

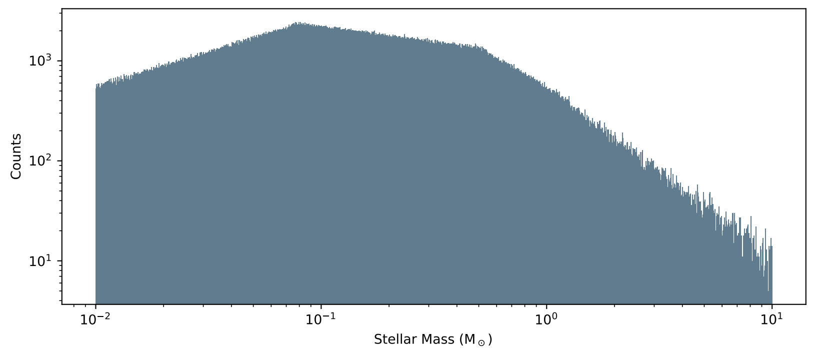


Figure 3. The distribution plot of 1 million stars generated by our initial mass function. It is obvious that there are three power law distribution between 0.01 $M_\odot$ and 0.08 $M_\odot$, 0.08 $M_\odot$ and 0.5 $M_\odot$, 0.5 $M_\odot$ and 1 $M_\odot$, 1 $M_\odot$ and 10 $M_\odot$. Where the cutoff occurs at 10 $M_\odot$ and 0.01 $M_\odot$

The calculation of $\dot{a}$, also denoted as j, the jerk, is as follows

$$\mathbf{j_i} = -G \sum_{j=1,j\neq i}^{N} m_j \left(\frac{\mathbf{v_i} - \mathbf{v_j}}{|\mathbf{r_i} - \mathbf{r_j}|^3} - 3\frac{(\mathbf{v_i} - \mathbf{v_j})(\mathbf{r_i} - \mathbf{r_j})}{|\mathbf{r_i} - \mathbf{r_j}|^5} \right)(\mathbf{r_i} - \mathbf{r_j}) \tag{14}$$

3.2. *Time and timestep selection*

Normally, the best time step would be variable individual timesteps, where the criterion of chosing is that,

$$\frac{1}{6}\Delta t^3 |j_i(t)| \ll \frac{1}{2}\Delta t^2 |a_i(t)|$$

$$\text{and we will have } \Delta t \ll \frac{|a_i|}{|j_i|} \tag{15}$$

However, this criterion requires extrapolation for synchronization, and that separating the computation of each particle should be a bad idea for when working with `numpy`. Therefore, currently I set the time step to

$$\Delta t = N^{-1/3}$$

This is derived from a more simpler estimation, stating that for an individual particle, the typical choice is

$$\Delta t = \frac{r}{v},\ r = \frac{R}{N^{1/3}}$$

where r is the typical distance to the nearest neighbor, plugging it and we get the result.

I later realize this was the time step used to calculate the complexity and shouldn't be used as the actual time step.

Therefor, I rethink the criterion of 15 and think I could set dt to meet the criterion of close encounter.

For close encounter, we have the radius rc, and suppose $m = M/N$ and since $M = 1$, we have $m = 1/N$. Assuming circular orbit, we have

$$m\frac{v^2}{r_c} = \frac{Gm^2}{r_c^2}$$

$$v = \sqrt{\frac{Gm}{r}} = \sqrt{\frac{1}{Nr_c}}$$

plugging into a and j (14), we have

$$dt = |\frac{a}{j}| = |\frac{r_c^{3/2}}{6\sqrt{N}}|$$

However, this is too small. For example, we choose $N = 1000$ and $r_c = 0.01$, we have $dt = 5.27 \times 10^{-6}$. This is even smaller than the time step of close encounter which is $dt = R$ (will be mentioned in the next section.)

3.3. *Collisional Treatment*

Generally speaking, when detected the two star getting too close, we treated the two star as a binary and conduct regularization. The goal of regularization is to make sure two bodies behave normally even under small R. To ease the computation, we separate the single stars and binaries into different data structure. A beautiful approach to implement this is mentioned in the first few sections in Aarseth's book. It suggest assigning one star the total mass of the two stars, and its position to the position of the center of mass. And then we treated the other star as ghost particle, and set its mass to zero mass. Then, we put the parameters of the two stars into the list of binaries conducting two body regularization.

And for the regularization method, originally I decided to implement Kustaanheimo–Stiefel method, however, I failed to comprehend the physical meanings of it after days of studying and paper reading. Furthermore, I discovered that the book mentioned such method required plenty of coordination transformation and could lead to excessive computations. Thus, I eventually settled on Burdet-Heggie's alternative. Since it only requires time transformation, it allows my brain to understand some of the algebraic meanings. The method is given in (Aarseth 2003).

The regularization method is stated as follows:

First we do the time transformation, we would need plenty of smaller time steps in-between the global time steps to conduct regularization.

$$t' = \frac{dt}{d\tau} = \eta R \tag{16}$$

η is a fraction being chose by experiment. Then, accordingly, the prime derivative of R are the derivative with respect to τ, that is

$$R' = R\dot{R} \tag{17}$$

We define P and B

$$P = -2M/R + \mathbf{R'^2}/R^2 \tag{18}$$

$$\mathbf{B} = M\mathbf{R}/R - \mathbf{R'^2}/R^2 + R'\mathbf{R}'/R \tag{19}$$

$$P' = \mathbf{R}'\dot{\mathbf{F}} \tag{20}$$

$$\mathbf{B}' = -2((\mathbf{R}' \cdot \mathbf{F})\mathbf{R} + (\mathbf{R} \cdot \mathbf{R})\mathbf{F} \tag{21}$$

Eventually, we can compute $\mathbf{R}''$ with B and P

$$\mathbf{R}'' = P\mathbf{R} + \mathbf{B} + R^2\mathbf{F} \tag{22}$$

The overall procedure is given in Figure 12 in the appendix section.

With the given procedure, I tested several parameters and see the corresponding R evolving with time. See Figure 13, 14 in the appendix section. Conducting 2D experiment, we show that the ones with static dt show better stability in R than the ones which we dynamically update the dt. And we found that setting dt to $|R|/10$ is a good choice in both unperturbed and perturbed cases regarding the balance between stability and time consumption.

3.4. *Parallel Computation*

I tried to understand parallel computing modules in python, but it is a little complicated. Therefore, I asked for ChatGPT's help. To boost the speed of calculating the acceleration and the jerk, I adopt `njit` from `Numba` . And for collision regularization, I adopt `multiprocessing` module.

4. RESULTS

It is well known that the destruction of a globular cluster are caused by three main mechanisms, stellar evaporation, tidal shocks and dynamical friction. Here we will analysis the influence of stellar evaporation and dynamical friction based on our naive simulations and analysis the evolution of the cluster based on core collapse, mass loss rate and the Q factor ($K/|U|$).

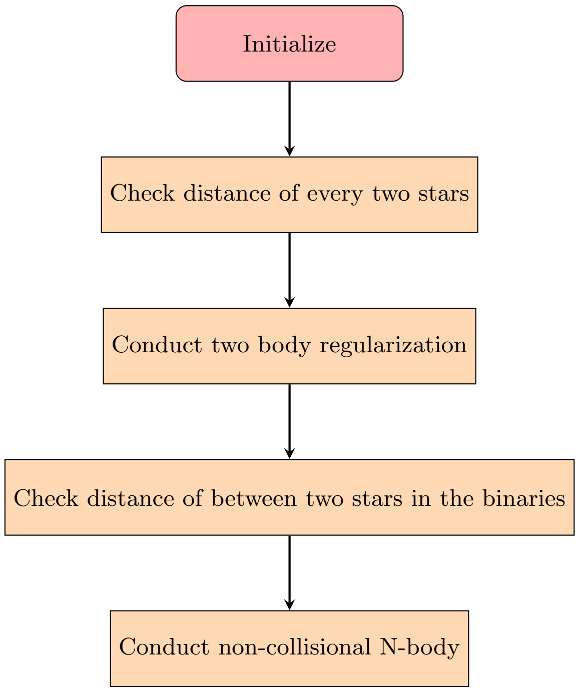

Initialize
Check distance of every two stars
Conduct two body regularization
Check distance of between two stars in the binaries
Conduct non-collisional N-body

Figure 4. Overall Structure of the code

4.1. *Measuring Core Radius*

I followed an empirical definition of Core radius given in Aarseth's book (Aarseth 2003) which came originally from (von Hoerner 1963)

With $\rho_j = \frac{3}{4\pi} 3.5 m / r_{j3}^3$, I first calculate the position of the density center

$$\mathbf{r_d} = \sum_{j=1}^{N} \rho_j \mathbf{r_j} / \sum_{j=1}^{N} \rho_j$$

where r_{j3} is the third nearest neighbor distance. I created a KDTree with the help of `scipy.spatial.KDTree`, and then use the `KDTree.query` function to find r_{j3}

And then, using the density center, we may find the core radius empirically.

$$\mathbf{r_c} = \sum_{j=1}^{N} \rho_j |\mathbf{r_j} - \mathbf{r_d}| / \sum_{j=1}^{N} \rho_j$$

This is basically a weighted sum of each star in the simulation. Base on the above approach, I analyze the cluster under different collision condition. The three conditions are respectively non-collisional, collisional without considering nearby single stars' movement during regularization, and collisional while considering nearby single stars' movement. See Figure 6. We may observe that the non-collision one undergone slight collapse of core which is consistent with the theory.

4.2. *Mass Segregation and Mass Loss*

Due to dynamical friction, large mass stars tend to be slowed down by other stars, while low mass stars tend to be accelerated. Due to the flaw of my code, binaries would be viewed as a single star.

To examine this phenomenon, I made two attempts. The first naive approach I came up with is the average distance from the core of different mass stars. I only take those collisionless simulation into account and separated the stars into three groups. The mass are transformed back from Henon's unit to solar mass. And then we take the logarithm of their mass, while separating them into $-3 < log(m) < -1$, $-1 < log(m) < 0$, and $log(m) > 0$

There are two downsides of this attempt. The first one is the determination of the core, though the initial condition states that the core should be at $(0, 0, 0)$. The second one is the lack of quantification of mass segregation. Since I did

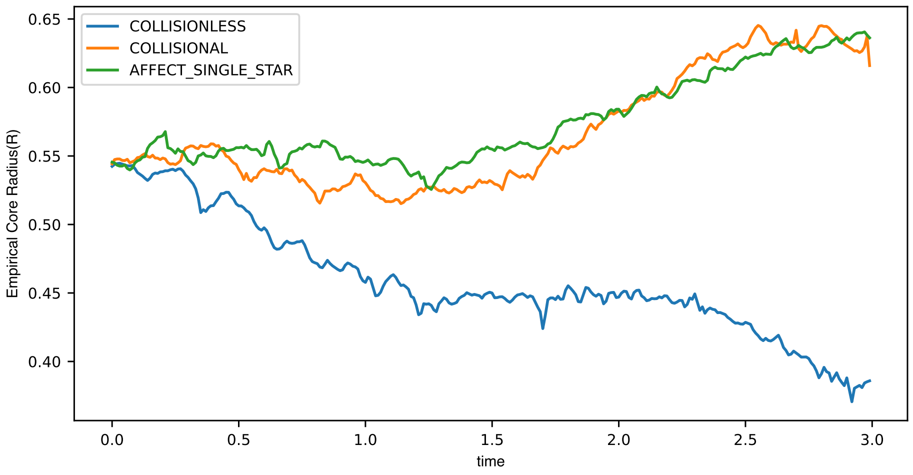


Figure 5. Core Radius of the globular cluster under three conditions, we may observe that the non-collision one undergone core collapse.

not observed obvious result, I went for a second attempt. I bumped into a method given by (Allison et al. 2009). They utilized minimum spanning tree(MST) for calculation of the model. A minimum spanning tree is a tree that connects all the vertices with the minimum total weight of the edges. In our context, the vertices are the stars, and the edges are the distance between every two stars. It can be proved that the length of the MST is unique. Thus comparing the length of the MST of largest star, and the MST of same amount of random picked star could give us a good probe on mass segregation.

In the paper they use the mass segregation ratio Λ_{MST} to quantify the mass segregation of a globular cluster.

$$\Lambda_{MST} = \frac{< l_{norm} >}{l_{massive}} \pm \frac{\sigma_{norm}}{l_{massive}}$$

The $< l_{norm} >$ is the average length of the MST of N random picked stars, and $< l_{massive} >$ is the length of the MST of the N most massive stars in the cluster.

The result is shoown in Figure 6. A slight mistake I made is that my code can not distinguish between binaries and normal stars. Therefore, I did not tested it on collisional simulations. According to the figure, I do not see significant mass segregation, or a overall trend of it.

5. DISCUSSIONS

Strictly speaking, there are a lot of flaws within this simulation and even more inefficiency regarding the code. To begin with, the number of stars being test are significantly smaller than the actual value.

And since we only utilize two-body regularization, the method didn't specify how to compute the effect on perturbing force and how the binaries effect the perturbance. Therefore, I think the best way to conduct collisional simulation would be to do variable time step for each star, so that I wouldn't need to separate the computation of binaries and single stars into two stage.

As for the result analysis, I didn't consider the tidal effect, so that stars could fall back inside the cluster even if it had traveled a long way out.

5.1. *Interpretation of the Results*

5.1.1. *Mass segregation*

Though often time we observe Λ larger than 1, we do not found an overall trend for mass segregation during any of out simulation. I later found out that this is probably because the the relaxation time scale is around $\frac{N}{6ln(N/2} * t_{cross}$, and $t_{cross} = R/V \approx 1$. Plugging $N = 10000$, we found out the timescale is around 200, and our simulation time is only 3. Therefore, it is reasonable that we do not observe this phenomenon. Since I do not have the time to run all the simulation again, I conduct a simple non-collisional simulation with $N = 1000$. The crossing time t_{cross} of which

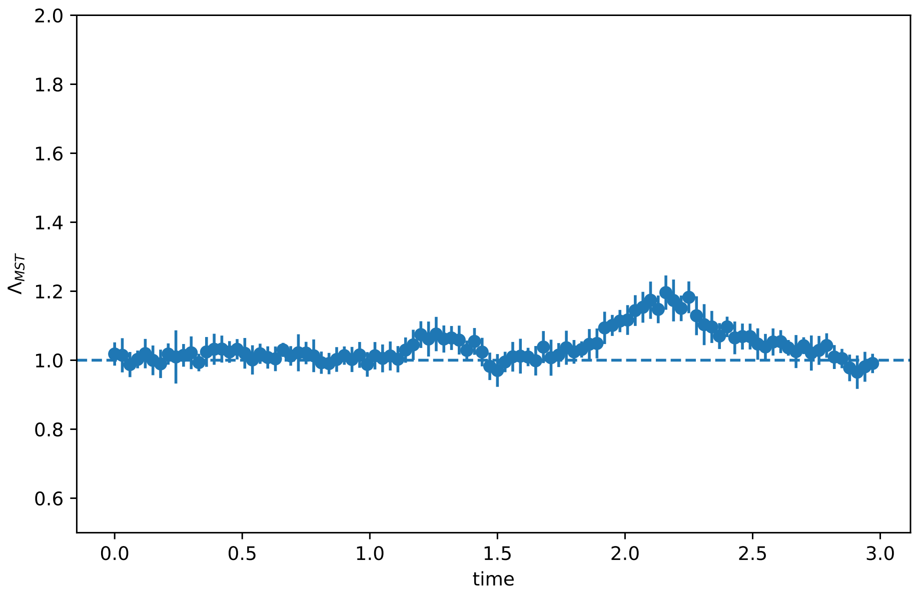


Figure 6. The mass segregation of the cluster, the vertical axis denotes the ratio of separation of massive stars to that of normal stars. $\Lambda_{MST} > 1$ denotes that mass segregation occurs. I chose massive stars to be above 1.5 $M_\odot$. According to the plots, we do not observe significant mass segregation.

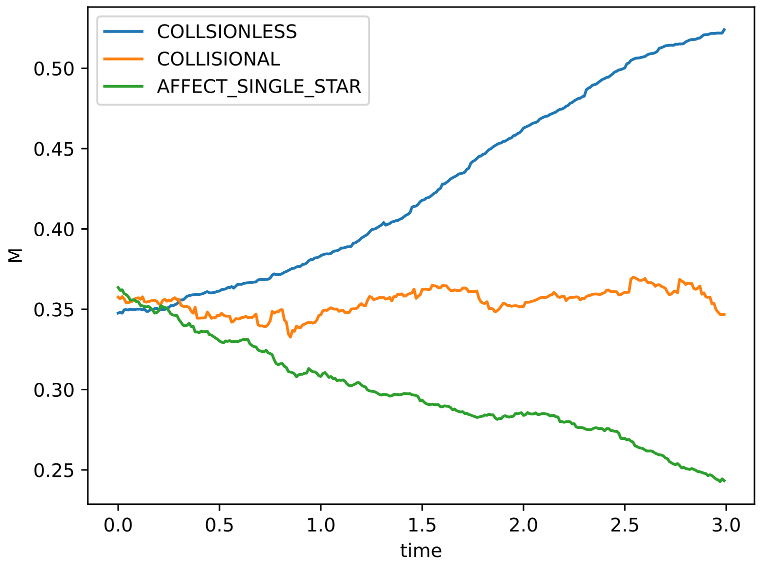


Figure 7. Mass loss of the cluster, calculated by adding up the mass within the original virial radius $R = 1$. We can observe that the simulation without binaries have more mass fall inside the cluster, while those with binaries do not. I suspect this is because of binaries spreading injecting energy into the cluster, thus stars gain energy and thereby leaving the cluster.

is around 20. The result turn out supporting my guess, which we can see the mass segregation climb at around 17. See Figure 8.

5.1.2. *Core Collapse*

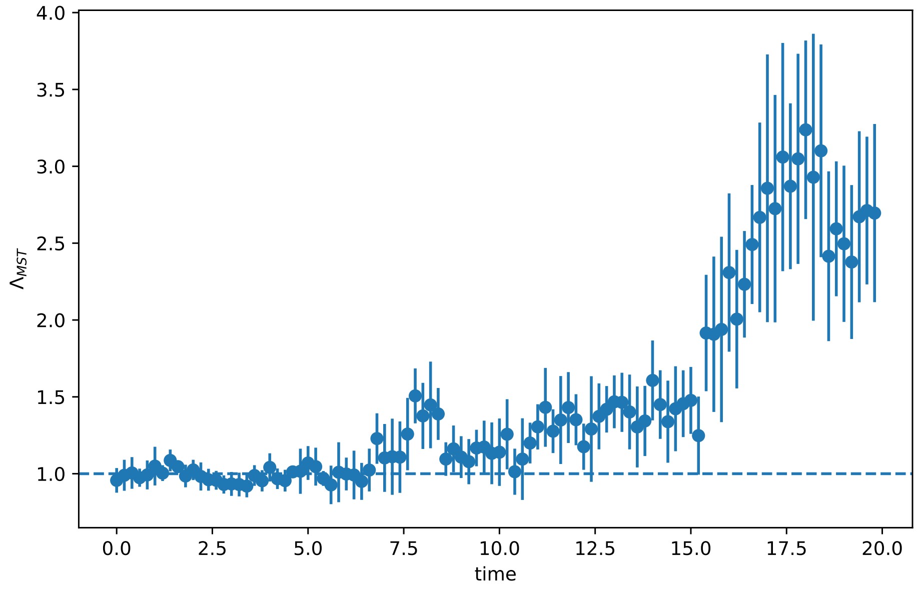


Figure 8. Mass segregation of 1000 particles. We may see a peak in around 17. Which is predicted by calculation.

It is quite obvious that the cluster operates under collisionless condition happened to go through core collapse, while those operates under collisional conditions did not. However, I do doubt authenticity of the simulation. I plot the Q value (given by $E/|U|$) under the three different conditions. Normally, the cluster under equilibrium would have $Q = 0.5$, and the cluster undergoing collapse have $Q < 0.5$. Yet the Q value of the collisional simulation goes to above 1. See Figure 9

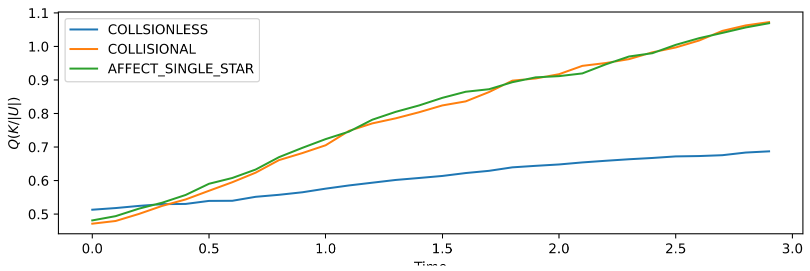


Figure 9. The virial ratio Q under three different conditions. It is given by $Q = E/|U|$

5.1.3. *Confusion on the Cluster's Radius*

Since there is no tidal cutoff, we do not have a obvious way of determining the boundary of the cluster. Thus, most of out analysis are based on the assumption of the radius of the cluster equals its original virial radius $R = 1$.

5.2. *Possible Improvement*

5.2.1. *Initial Condition*

We use the Plummer's model to implement initial conditions. There are several improvement that can be made, but the most fatal one is the error of calculating initial velocity. From the result, we know that there exist a horizontal truncate of star regarding the velocity, and beyond the truncate, there is a strip of stars. The horizontal truncate is a result of a bug in the code, that I did not assign the escape velocity to the function. And for the strip of stars, I suspected that this was the result of root finding. We do not find a proper root when utilizing `fsolve`. Therefore, I plot the function of ζ given in Eq 9, see Figure 10. I judging from the model, the proper choice of root should be in the region of (0, 1) which could be interpret as the speed in the cluster should be lower than the value of escape velocity. I later test my hypothesis by setting constrain on the root finder. To do this, I switched the root finder to `least_squares` from `scipy.optimize`. We can compare the result in Figure 11. However, though the strip of star disappear, the truncate didn't seem to disappear. I therefore plot the escape velocity, and it seems that it is a "steep" decrease rather than a direct truncate. However, there is a problem left, the modified condition doesn't fit virial equilibrium.

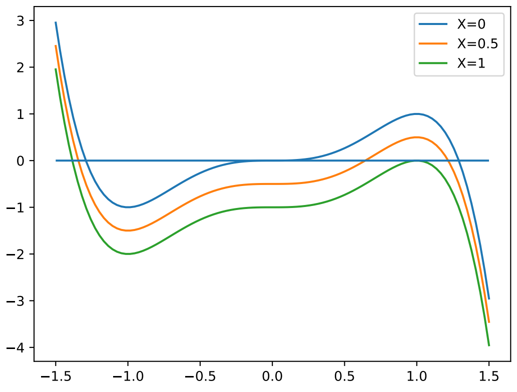
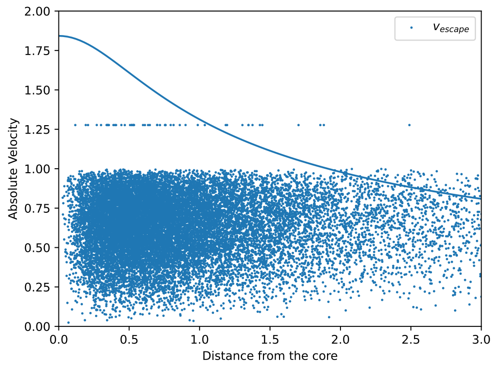
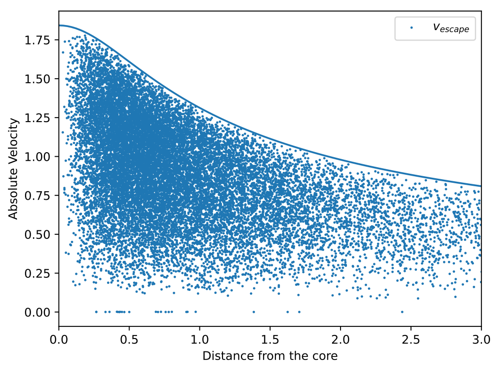


Figure 10. The function plotting with respect to the random variable X. It is solved with `fsolve`. The desired range is possibly between 0 and 1



Phase plot of the initial condition computer with `fsolve`



Phase plot of the initial condition computer with `least_square`

Figure 11. Phase plots of initial condition where the horizontal axis shows the star's distance from the center, and the vertical shows its velocity. We may observe the strip of stars above the cutoff disappear in the least_square plot

, and the horizontal truncate is modified to fit better with the escape velocity.

On the instance of writing this report, I do not come up with another possibility aside from this reason.

Another problem regarding the initial condition is the lack of primordial binaries. We relies heavily on close encounters for the formation of binaries, this is unreal.

5.2.2. *Computation Efficiency*

Since direct summation required $O(n^2)$ time to run, I did consider using somebody else's kd-tree code that may provide a $O(nlogn)$ efficiency. However, I did not find any library that perfectly suit my requirements. (Some are for computer graphics, and others are for machine learning.) The `scipy.KDTree`, as far as I know, did not have the ability to keep track of the summation of large box, but I did use it on several analyzing method.

I abandoned the idea to write my own tree code, since it required a lot of time to design and debugging, also it do not guaranteed performance against numpy, who has already been optimized. In this case, I found the possibilty to gain more efficiency by implementing `njit` (just-in-time compilation) to accelerate the computation of force. I further attempt to do parallel by setting `njit(parallel=True)`, but only to found that the program slowed down when testing the time. It could be the overhead of creating parallel.

Another efficiency improvement could be done is the code where we used to analyze mass segregation. The `minimum_spanning_tree` used in `scipy` is implemented with Kruskal algorithm. However, since the n-body is a dense graph with the number of edges significantly larger than that of vertices, Prim's algorithm would provide a better performance. This is because in n-body simulation, $E = V^2$, where E denotes the number of edges and V denotes the number of vertices. Kruskal's requires $O(ElogV) = O(V^2logV)$ while Prim's provides an amortized $O(E + VlogV) = O(V^2)$ case when implementing Fibonacci Heap. But after all, since the amount of stars is small, the time to run the program is still acceptable.

5.2.3. *Computation Accuracy*

In the Results section, we found out that the energies in the binaries often exploded into great values and then the binary system broke down. I do not believe this is the result of some close encounters, but some numerical error. I should have done a more general test on the collisional system rather than plotting a random value and judge the performance by subjective observation. Also, I would like to implement individual time time steps if I have more time. The two collision scheme I conduct both contain several problems. For the normal one, it do not affect the behavior of single stars, and thus assume that the stars are static during regularization. However, this is totally run, as I found the time step of the binaries is often similar to that of a single star. And for the `AFFECT_SINGLE_STAR`, since we view the binary as a single star, it would recompute the force when conducting normal test. Therefore, the force added on single star would doubled.

The other realistic condition we do not consider is the existence of primordial binaries. The problem is significant because the formation of the binary only relies on close encounter and that when the amount of particles is low, we do not have enough binaries. This problem could further extend to when should I conduct two-body regularization. If the amount of particles is low, then I shall set larger range for (or loosen the judgement of) close encounter. There is no objective way of the decision making.

Overall, I think the greatest change I could make is to focus on a more specific topic, and try to figure out the physical meanings to verify and facilitate my project. To validate the correctness of initial condition, fitting it with a model should be a way better choice then subjective judgement. For example, fitting it with the cumulative mass and velocity dispersion. And I should also estimate the relaxation time in advance, to have a better knowledge of how long should be a reasonable time for core collapse.

However, since the goal of n-body simulation is its overall statistical outcomes, I believe the result still remain some degrees of consistency with reality.

I have to admit I do not know the difficulty to conduct a collisional test in the beginning, but I did learn a lot from it and had a lot of fun.

REFERENCES

Aarseth, S. J. 2003, Gravitational N-Body Simulations

Aarseth, S. J., Tout, C. A., & Mardling, R. A. 2008, The Cambridge N-Body Lectures, Vol. 760, doi: 10.1007/978-1-4020-8431-7

Allison, R. J., Goodwin, S. P., Parker, R. J., et al. 2009, Monthly Notices of the Royal Astronomical Society, 395, 1449–1454, doi: 10.1111/j.1365-2966.2009.14508.x

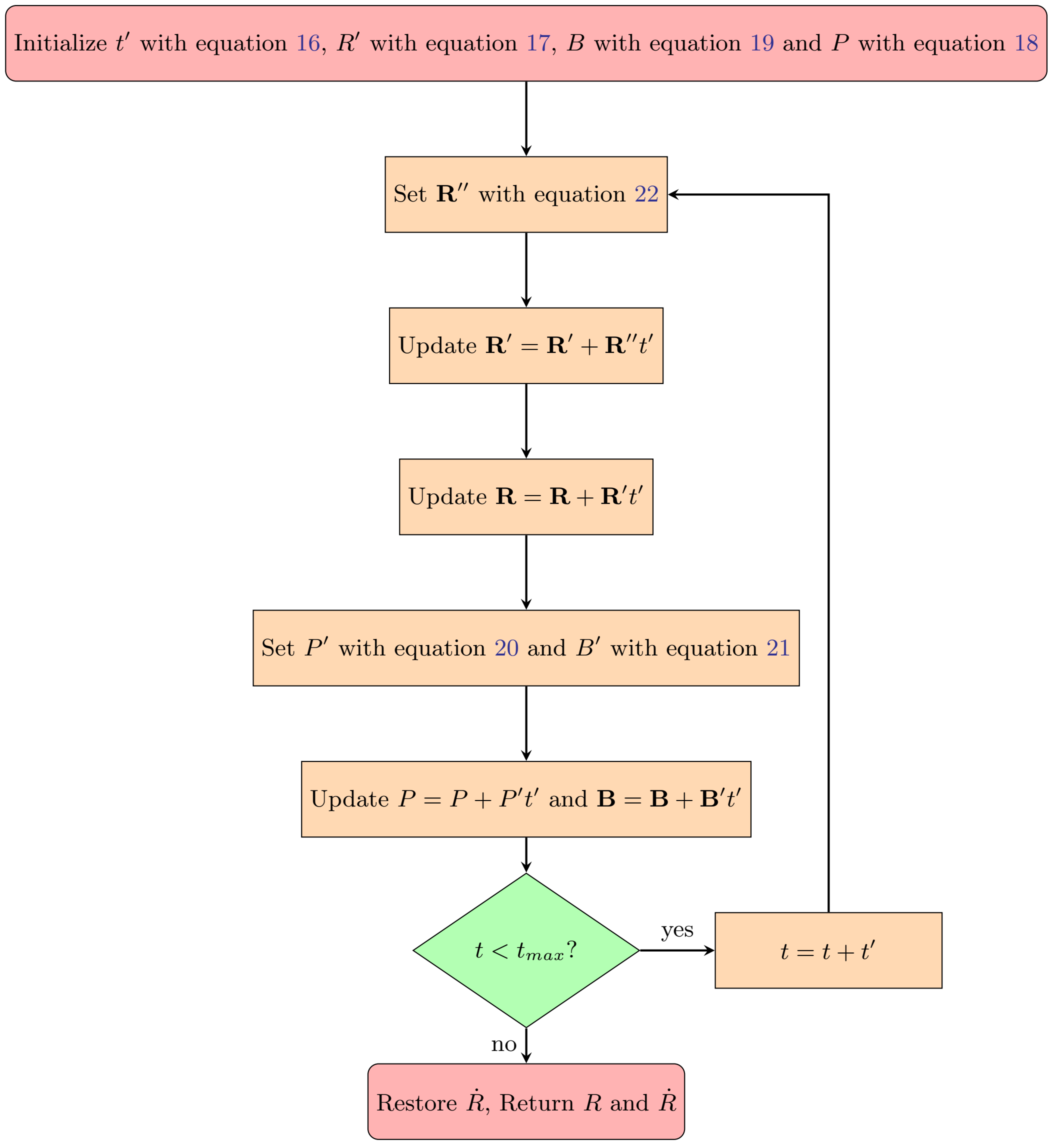
Heggie's lecture slide. 2003, Heggie's lecture slide on Computation and astrophysics of the N-body problem

Kroupa, P. 2001, MNRAS, 322, 231, doi: 10.1046/j.1365-8711.2001.04022.x

Plummer, H. C. 1911, MNRAS, 71, 460, doi: 10.1093/mnras/71.5.460

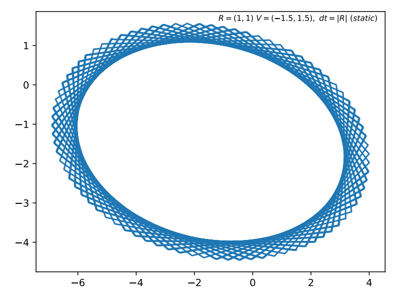
von Hoerner, S. 1963, ZA, 57, 47

APPENDIX

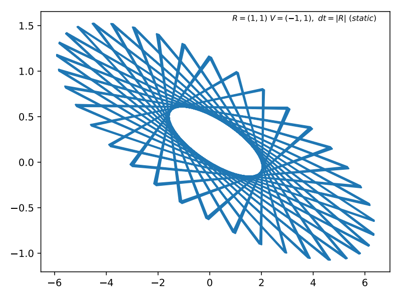


Initialize t' with equation 16, R' with equation 17, B with equation 19 and P with equation 18
Set $\mathbf{R}''$ with equation 22
Update $\mathbf{R}' = \mathbf{R}' + \mathbf{R}''t'$
Update $\mathbf{R} = \mathbf{R} + \mathbf{R}'t'$
Set P' with equation 20 and B' with equation 21
Update $P = P + P't'$ and $\mathbf{B} = \mathbf{B} + \mathbf{B}'t'$
$t < t_{max}$?
yes
$t = t + t'$
no
Restore $\dot{R}$, Return R and $\dot{R}$

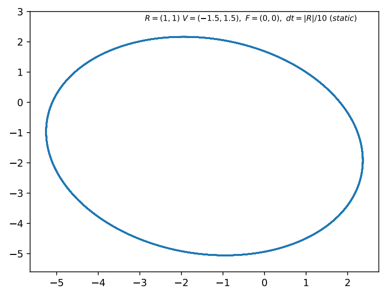
Figure 12. Flow chart of Collisional Treatment



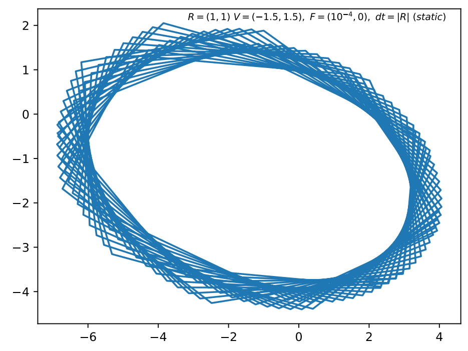
(a) $R = (1, 1),\ V = (-1.5, 1.5),\ dt = |R|$ (static)



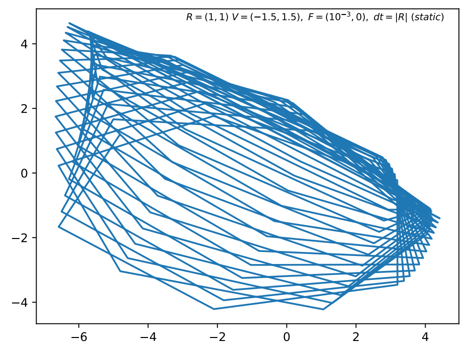
(b) $R = (1, 1),\ V = (-1, 1),\ dt = |R|$ (static)



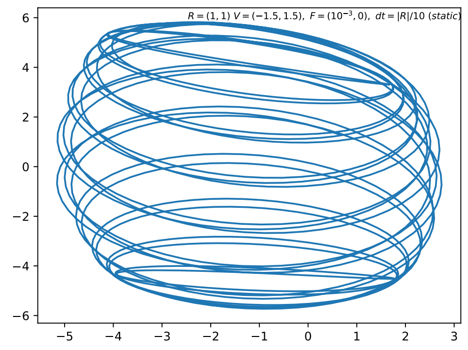
(c) $R = (1, 1),\ V = (-1.5, 1.5),\ dt = |R|/10$ (static)



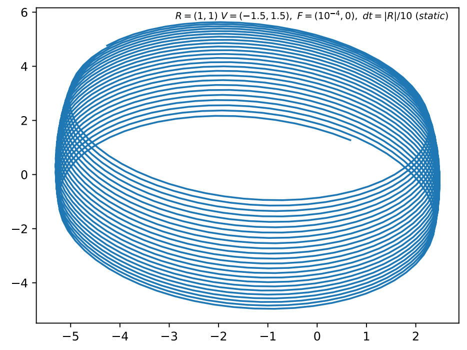
(d)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-4}, 0),\ dt = |R|$
(static)



(e)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-3}, 0),\ dt = |R|$
(static)

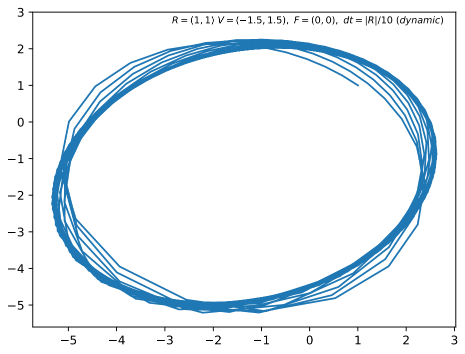


(f)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-3}, 0),\ dt = |R|/10$
(static)

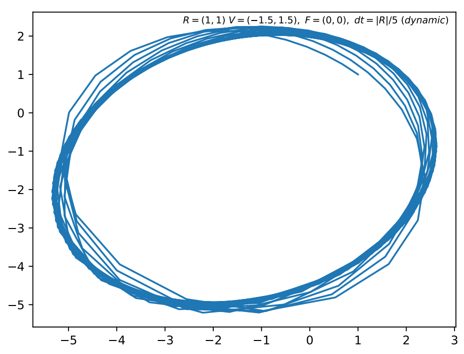


(g)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-4}, 0),\ dt = |R|/10$
(static)

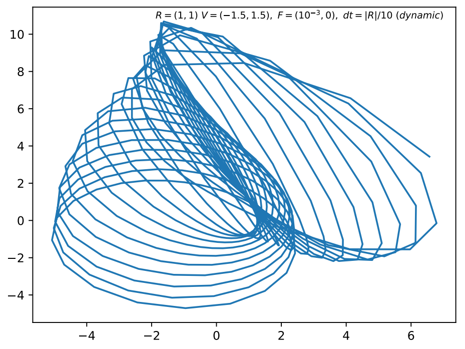
Figure 13. Plots of $\mathbf{R}$ in 2-dimension regularization test using Burdet-Heggie's alternative with static dt. Comparing (a), (b) and (c), we may find out (c), whos has $dt = |R|/10$ has the best stability. Similarly, under perturbation F, we may also observe that (f), (g) with $dt = |R|/10$ behave more stable then (d) and (e) who have $dt = |R|$



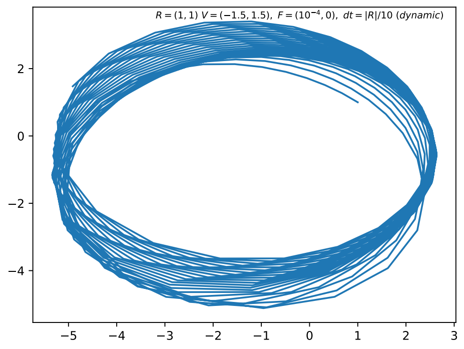
(a) $R = (1, 1),\ V = (-1, 1),\ dt = |R|/10$ (dynamic)



(b) $R = (1, 1),\ V = (-1.5, 1.5),\ dt = |R|/5$ (dynamic)



(c)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-3}, 0),\ dt = |R|/10$
(dynamic)



(d)
$R = (1, 1),\ V = (-1.5, 1.5),\ F = (10^{-4}, 0),\ dt = |R|/10$
(dynamic)

Figure 14. Plots of $\mathbf{R}$ in 2-dimension regularization test using Burdet-Heggie's alternative with dynamic dt. We may observe that when dt is dynamic, it wouldn't be as stable as when it is static. We may compare (a) with (c) in Figure 13. We do not show the plots of $dt = |R|$ since it would blow up and did not show a periodic orbit.