ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА

(повне найменування вищого навчального закладу)

ННІ «ФІЗИКО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ»

(назва факультету)

КАФЕДРА ФІЗИКИ ЯДРА ТА ВИСОКИХ ЕНЕРГІЙ ІМЕНІ О.І.АХІЄЗЕРА

(повна назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
БАКАЛАВРА
(освітньо-кваліфікаційний рівень)

на тему
укр. «Фізичне моделювання відхилення потенційно небезпечних астероїдів»
англ. «Physical modeling of deflection of potentially dangerous asteroids»

Виконав: студент 4 курсу навчання
за ОПП бакалавр
напрям підготовки 6.040204 «Прикладна фізика»
Панасюк А. С.
(прізвище та ініціали) (особистий підпис)
Керівник: д.ф.-м.н. Голубов О. А.
(прізвище та ініціали) (особистий підпис)
Консультант: Кириленко І. І.
(прізвище та ініціали) (особистий підпис)
Рецензент: к.ф.-м.н., доц. Станкевич Д. Г.
(прізвище та ініціали) (особистий підпис)

Харків – 2024 рік

АНОТАЦІЯ

Панасюк Андрій. – Фізичне моделювання відхилення потенційно небезпечних астероїдів. – рукопис. Дипломна робота бакалавра з прикладної фізики (напрям підготовки «Прикладна фізика») – Харківський національний університет імені В. Н. Каразіна, Харків, 2024.

37 с., 16 рис., 1 табл., 8 джерел

Метою дипломної роботи є розв'язання однієї із задач в галузі астероїдного захисту – обчислення величини та напрямку збурення Δv швидкості потенційно небезпечного астероїда необхідного для уникнення зіткнення з Землею. В першому розділі описано програму для чисельного розв'язку задачі й наведено обчислені в цій програмі Δv для різних проміжків часу до зіткнення. Другий розділ присвячено виведенню аналітичних наближень отриманої чисельно залежності $\Delta v(t)$ в різних часових областях. Результати роботи можуть лягти в основу подальших досліджень переспрямування астероїдних орбіт з метою захисту планети від можливих зіткнень.

ABSTRACT

Panasiuk Andrii. – Physical modeling of deflection of potentially dangerous asteroids. Bachelor's thesis in applied physics (specialty – «Applied physics») – V.N. Karazin Kharkiv National University, Kharkiv, 2024.

37 p., 16 fig., 1 tabl., 8 ref.

The purpose of this thesis is to solve one of the problems in the field of asteroid defense - to calculate the magnitude and direction of the perturbation Δv of the velocity of a potentially hazardous asteroid necessary to avoid a collision with the Earth. The first section describes the program for solving the problem numerically and presents the Δv calculated in this program for different time intervals before the collision. The second section is devoted to the derivation of analytical approximations of the numerically obtained dependence $\Delta v(t)$ in different time domains. The results can be used in further studies of redirecting asteroid orbits to protect the planet from possible collisions.

ЗМІСТ

ВСТУП

Сучасний стан астероїдних досліджень

Астероїдами називають малі (порівняно із планетами) кам’яні або металеві небесні тіла. За розташуванням у Сонячній системі вони поділяються на такі основні види (рис.1.):

- Астероїди головного поясу розташовані між Марсом та Юпітером. До них належить більшість відомих астероїдів.

- Троянці – астероїди, що знаходяться в орбітальному резонансі 1:1 з планетою. Найчисленнішими представниками класу є троянці Юпітера, проте інші планети (навіть Земля) теж мають своїх троянців.

- Група Гільди – група астероїдів, що рухаються поза головним поясом та перебувають в орбітальному резонансі 3:2 з Юпітером.

Навколоземні астероїди – астероїди з перигелієм не більше 1.3 астрономічні одиниці (1 а. о. дорівнює відстані між Землею та Сонцем), тобто з орбітою так чи інакше близькою до земної. На сьогоднішній день налічують понад 30000 таких об’єктів різних розмірів і продовжують відкривати нові [https://neo.ssa.esa.int/risk-list]

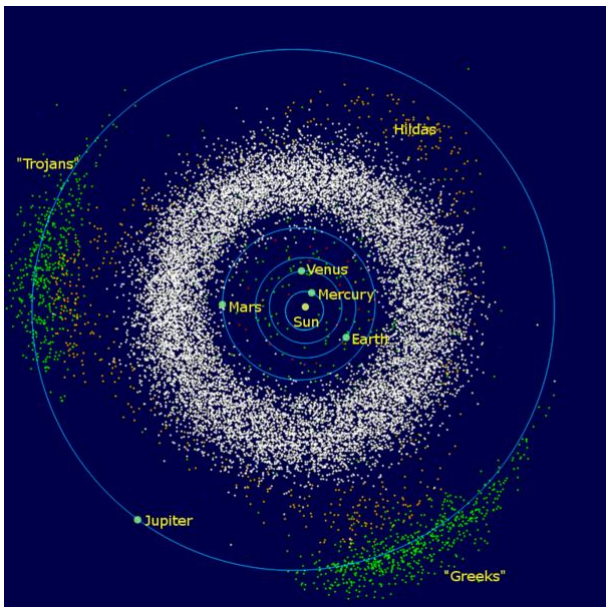

Рис.1. Розташування астероїдів у Сонячній системі

[https://uk.wikipedia.org/wiki/Пояс_астероїдів#/media/Файл:InnerSolarSystem-uk.png]

Зважаючи на ризики, які представляють людству астероїди, було створено профільні установи зі спостереження за навколоземними астероїдами, а також проведено низку космічних місій для їх вивчення. До таких місій належать [1]:

- NEAR Shoemaker (1996-2001) – перший космічний апарат, що вийшов на орбіту довкола астероїда;
- Hayabusa (2003-2005) та Hayabusa 2 (2019-2020) – космічні апарати, призначені для доставки дослідних зразків астероїдів на Землю;
- Dawn (2007-2018)– космічний апарат, що відвідав два найбільші об’єкти поясу астероїдів: Цереру і Весту.
- OSIRIS-REx – відвідав астероїд 101955 Бенну в 2018 році, станом на сьогодні летить до астероїда 99942 Апофіс – одного із найнебезпечніших навколоземних об’єктів;
- DART (2021-2022) – космічний зонд, відправлений з метою дослідження зміни руху астероїда під дією спланованого зіткнення.
- Lucy (запущено в 2021) - космічний зонд, відправлений з метою дослідження 8 різних астероїдів, переважно троянців Юпітера;
- Psyche (запущено в 2023) – місія з дослідження однойменного астероїда, підозрюваного на металевий склад.

Під час виконання роботи були використані каталоги навколоземних об’єктів від CNEOS (Center for Near-Earth Object Studies) та NEOCC (Near-Earth Object Coordination Centre).

Астероїдна небезпека

Деякі навколоземні астероїди є потенційно небезпечними. Зіткнення з небесними тілами залишили геологічний слід та суттєвий вплив на біологічне різноманіття планети [6]. За даними Earth Impact Database, зареєстровано щонайменше 190 кратерів космічного походження.

Хоча загальноприйнятим параметрами потенційно небезпечного астероїда є відстань зближення щонайменше 0.05 а. о та діаметр щонайменше

140 м, варто зауважити, що тіла з меншими характерними розмірами також здатні заподіяти суттєвої локальної шкоди. Найвідомішими прикладами виступають Тунгуський (1908 р.), Сіхоте-Алінський (1947 р.) та Челябінський метеорити (2013 р.).

В таблиці нижче [1] наведено оцінки шкоди залежно від розмірів падаючого тіла.

Таблиця 1

Оцінки шкоди та частота ударів

Розмір об'єкта	Енергія удару (МТ)	Величина шкоди	Середня частота падіння (роки)	Розмір кратера, км
3-4 м	0.001	Жодної	1	n/a
25 м	1	Вибух у повітрі	100	n/a
50 м	10	Локальна	1000	1
140 м	100	Регіональна	10 000	3
500 м	5000	Континентальна	100 000	7
1 км	50 000	Глобальна	500 000	14
5 км	$6\ 10^6$	Глобальна	$20\ 10^6$	60
10 км	$50\ 10^6$	Глобальна/ Вимирання	$100\ 10^6$	100

Існує два способи для оцінки рівня потенційної небезпеки від певного астероїда - шкала Торіно та шкала Палермо [4].

Шкала Торіно (рис. 2.) приймає цілі значення від 0 до 10, де найменше значення присвоюють об'єкту з нехтовно малою ймовірністю зіткнення або ж тілам, що згорять в атмосфері, а найбільше – об'єкту, що становить глобальну небезпеку.

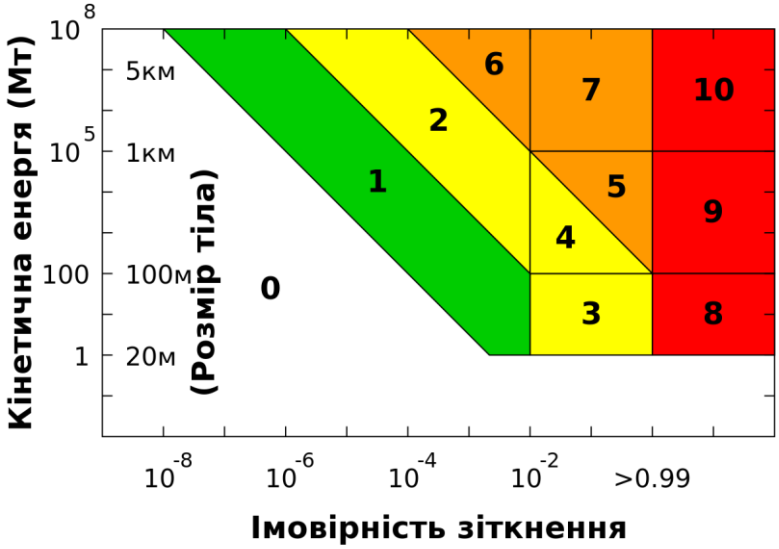

Рис. 2. Візуалізація шкали Торіно

[https://uk.wikipedia.org/wiki/Шкала_Торіно #/media/Файл:Torino_scale_uk.svg]

Шкала Палермо пропонує дещо інший підхід до цього питання. Вона порівнює ймовірність падіння потенційно небезпечного тіла із середнім ризиком, що створюється астероїдним фоном протягом визначеного проміжку часу. Оцінка за шкалою здійснюється за наступною формулою:

$$P = \log_{10} \frac{P_I}{f_B \Delta T}$$

Де $P_I -$ ймовірність зіткнення з досліджуваним тілом, $f_B -$ фонова частота зіткнень, $\Delta T -$ час до зіткнення з досліджуваним тілом.

В статті [4] звертають увагу на те що шкала Торіно, попри свої недоліки, чудово підходить для оповіщення населення про загрозу, але для більш фахової її оцінки зручніше використовувати шкалу Палермо.

Методи відхилення астероїдів

Наступним логічним кроком після виявлення та оцінки загрози є її знешкодження. Станом на сьогодні, вченими запропоновано різні методи зміни траєкторії астероїдів [2]:

- Зміна траєкторії небезпечного тіла за допомогою тривалої дії встановлених на поверхні рушіїв;

- Іонна або лазерна абляція – спосіб, який передбачає вплив на пучком іонів або лазерним променем на астероїд з спеціально призначеного космічного апарату. Відділена від поверхні порода створюватиме реактивну тягу, яка скеровуватиме небесне тіло в бажаному напрямку;
- Гравітаційний трактор - космічний апарат, який може змінити напрямок руху потенційно небезпечного астероїда, без фізичного контакту з ним, завдяки використанню гравітаційного поля.
- Кінетичний вплив – зміщення орбіти астероїда за допомогою зіткнення космічного апарату із ним.

Зосередимось на останньому методі. У випадку удару (короткодіючої механічної взаємодії), величину переданого астероїду збурення швидкості можна оцінити з закону збереження імпульсу.

$$M_a \Delta v = \beta p_{sc} \Longrightarrow \Delta v = \frac{\beta p_{sc}}{M_a}$$

де $M_a -$ маса цільового астероїда; $p_{sc} -$ імпульс космічного апарата; $\beta -$ коефіцієнт передачі імпульсу.

Місія DART показала можливість випадків, коли $\beta > 1$, що пов’язано з вильотом уламків астероїда внаслідок удару. Величина коефіцієнту передачі імпульсу залежить від багатьох факторів: рельєф та склад астероїда, форма космічного апарату, швидкість зіткнення тощо. Дослідження β є важливими для астероїдного захисту і досі тривають [5].

Метод кінетичного впливу, на нашу думку, є найбільш досяжним через наявні та випробувані людством технічні засоби. Саме тому в роботі проведено розрахунки виключно для миттєвих приростів швидкості.

РОЗДІЛ 1. ЧИСЕЛЬНИЙ РОЗРАХУНОК ВІДХИЛЕННЯ НЕБЕЗПЕЧНОГО АСТЕРОЇДА

На даний момент не існує аналітичного способу розв'язку задачі N-тіл в загальному вигляді. Тому для моделювання руху тіла в Сонячній системі доцільним є використання чисельних методів.

Під час виконання дипломної роботи було створено програму для пошуку оптимального миттєвого збурення швидкості астероїда, яке б запобігло потенційному зіткненню. В цьому розділі описано основні структурні елементи даної програми та отримані нею результати .

Більш розгорнутий опис програми наведено в Додатку А.

1.1. Бібліотеки

Існує безліч програмних бібліотек створених для полегшення написання та оптимізації коду. Під час виконання дипломної роботи окрім вже стандартних для мови програмування Python бібліотек для візуалізації, та обробки даних стали в нагоді наступні:

- astroquery – інструмент для доступу до різноманітних астрономічних баз даних та архівів, забезпечує зручний інтерфейс для виконання запитів до різних астрономічних служб, таких як NASA, ESA, CDS тощо.

- pyorb – інструмент для роботи з орбітальною механікою та аналізом орбіт. Дозволяє моделювати траєкторії небесних тіл та обчислювати їх орбітальні елементи.

- Rebound – бібліотека для моделювання руху та взаємодії астрономічних систем N-тіл. Вона підтримує найсучасніші методи чисельного інтегрування, що гарантує ефективне використання обчислювальних ресурсів та точність отриманих результатів. Серед запропонованих у бібліотеці інтеграторів, для чисельного моделювання поставленої задачі було обрано IAS15 (англ. Integrator for Adaptive Step-size 15th order). Його перевагами є адаптивний крок інтегрування, висока точність та стабільність у тривалих симуляціях.

1.2. Функції

Комплексні задачі зручно розбивати на більш прості. В програмуванні цей підхід реалізовують застосуванням користувацьких функцій. Нижче наведено основні функції створені для дослідження збурень астероїдів.

create_collider – створює штучний астероїд з заданими параметрами зустрічі з Землею та моделює його базову орбіту від моменту зіткнення до обраного моменту часу в минулому. В якості аргументів приймає: дату зіткнення, тривалість руху до зіткнення, часовий крок моделювання, метод чисельного інтегрування, швидкість та координати зіткнення. Результатом виконання функції є кеплерівські та декартові орбітальні параметри астероїда, а також вектори стану астероїда та Землі в геліоцентричній системі для кожного окремого часового кроку.

reverse_sim – перевірка оборотності моделювання орбіт (перевірка пов’язаних із моделюванням невизначеностей координат та швидкостей). Після create_collider симуляція орбіти проходить від заданого раніше моменту часу в минулому до дати зіткнення та фіксуються нові вектори стану.

perturbation – додає до базової орбіти збурення заданої величини у заданому напрямку та повертає нову орбіту.

simulation_mp – рахує відносну відстань до Землі для кожної збуреної орбіти та дозволяє встановити оптимальний напрямок прикладення збурення.

min_dist_func - рахує відносну відстань до Землі для одного астероїда з відомою базовою орбітою та вектором збурення.

bisection – знаходить корінь нелінійної функції методом половинного ділення (також відомий як метод бісекції).

1.3. Алгоритм

Система тіл для чисельного розрахунку складається з Сонця, усіх планет Сонячної системи та 400 невзаємодіючих між собою безмасових екземплярів синтетичного астероїда. Для спрощення подальшого аналізу, базова орбіта компланарна до орбіти Землі та єдина для всіх проведених в роботі обчислень.

В момент передбаченого зіткнення, швидкість астероїда становить 20 км/с та спрямована нормально до поверхні планети (типова величина для потенційно небезпечних навколоземних тіл). Шукане значення збурення отримуємо з умови дотичності зміненої траєкторії до Землі.

За допомогою бібліотеки astroquery, з бази даних JPL Horizons було отримано вектори стану для планет Сонячної системи, які в подальшому використовуються у чисельному моделюванні.

Наступний крок – введення в систему синтетичного астероїда із вказаними вище параметрами. Для цього застосовується функція *create_collider*, яка й відповідає за створення базової орбіти.

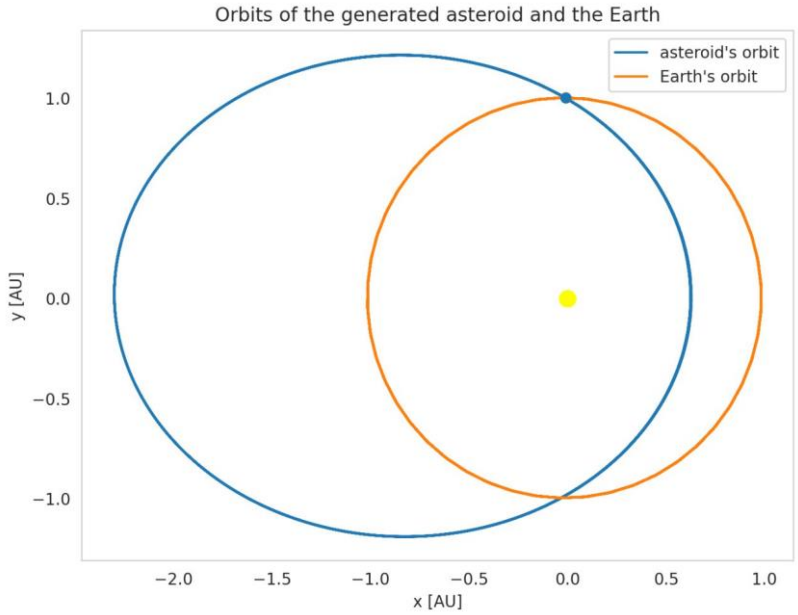

Рис. 1.3.1. Схематичне зображення геліоцентричних орбіт Землі (помаранчева лінія) та базової згенерованої орбіти (блакитна лінія)

Параметри базової орбіти отримані в *create_collider* передано в *reverse_sim.* Різниця між векторами стану астероїда на базовій та оберненій орбітах показує рівень похибки чисельного моделювання в кожній точці симуляції. Збурення Δv, яке не перевищує цей рівень немає сенсу розглядати в рамках детермінованої задачі;

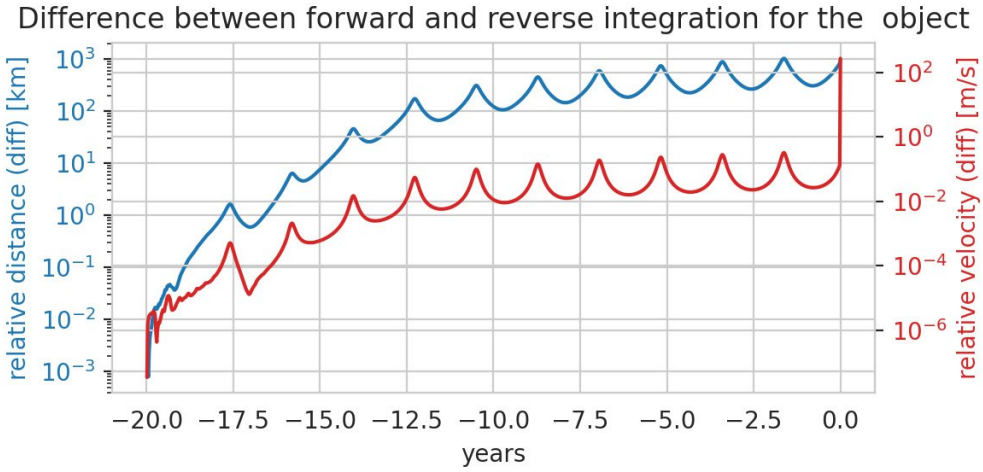

Рис. 1.3.2. Приклад графіку різниць координат та швидкостей для базової та оберненої орбіт в симуляції тривалістю 20 років

Після створення та перевірки базової орбіти створюємо набір збурених орбіт. До екземплярів синтетичного астероїда прикладаються збурення, які розподілені рівномірно та сферично-симетрично (див. рис. 1.3.). Кожному екземпляру відповідає єдиний напрямок збурення з фіксованою величиною Δv. На цьому етапі, засоби бібліотеки *pyorb* використовувались для розрахунку дії заданого Δv на кеплерівські орбітальні елементи орбіти астероїда.

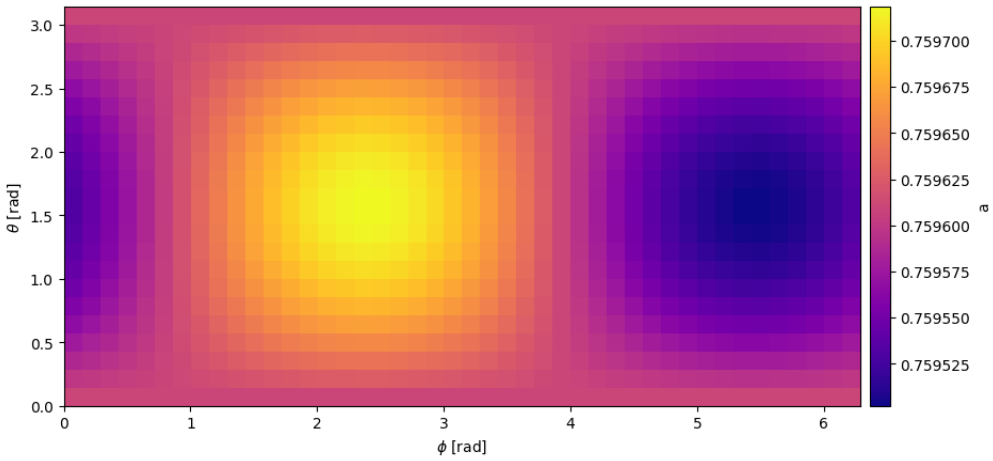

Рис. 1.3.3. Залежність великої півосі збуреної орбіти залежно від напрямку прикладення Δv в симуляції тривалістю 1 рік

Далі *simulation_mp* обчислює відстані прольоту повз Землю для кожного збуреного екземпляру. На основі цих даних будується відповідний графік (рис. 1.4.) та записується напрям, який дає максимальну відстань.

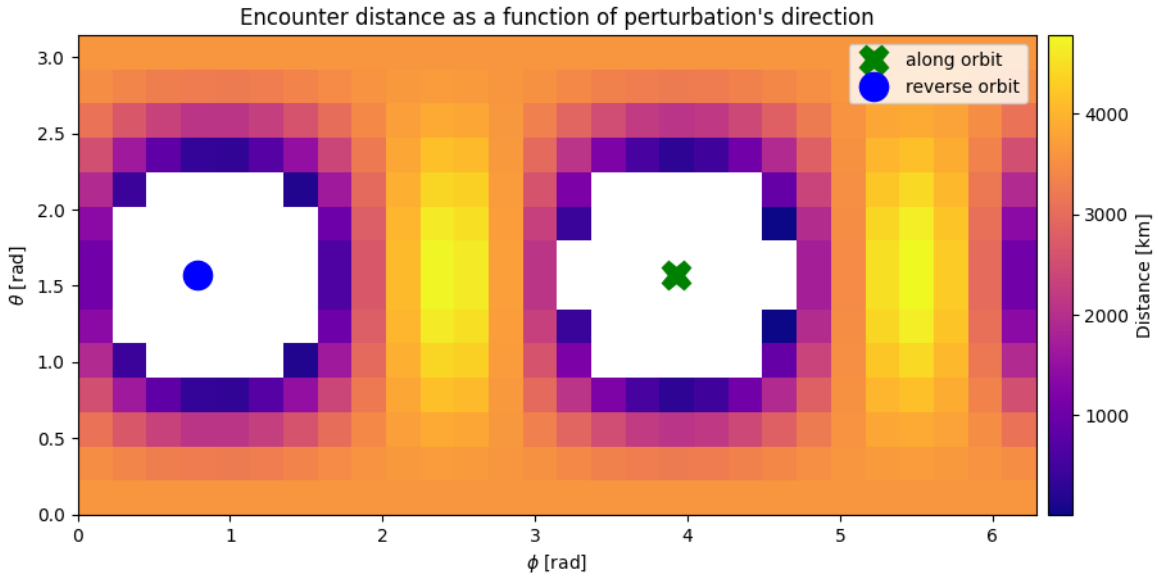

Рис. 1.3.4. Залежність відстані прольоту збуреного тіла залежно від напрямку прикладення Δv в симуляції тривалістю 1 рік

Після визначення оптимального напрямку збурення, обчислюється необхідна його величина. Програмно це реалізовано поєднанням функцій *min_dist_func* та *bisection*, за допомогою яких підбирається найменша величина збурення, якої достатньо для того, щоб переспрямувати траєкторію астероїда по дотичній до поверхні Землі.

Результати

Результатом успішного виконання алгоритму є набір збурень Δv необхідних для відхилення тіла, що впорядковані за часом їх прикладення до синтетичного астероїда.

Обчислені Δv можуть як відрізнятися на декілька порядків, так і слабко змінюватись впродовж певного часового проміжку. Тому задля зручнішої візуалізації чисельно отриманої залежності введемо знерозмірене збурення:

$$\Delta\tilde{v}(t) = \Delta v \cdot \frac{t}{R} \qquad (*)$$

Щоб вмістити на одному зображенні всі знайдені $\Delta\tilde{v}(t)$ для кожного t, ці величини прологарифмовано. Проаналізуємо отриманий графік (рис. 1.2.1.)

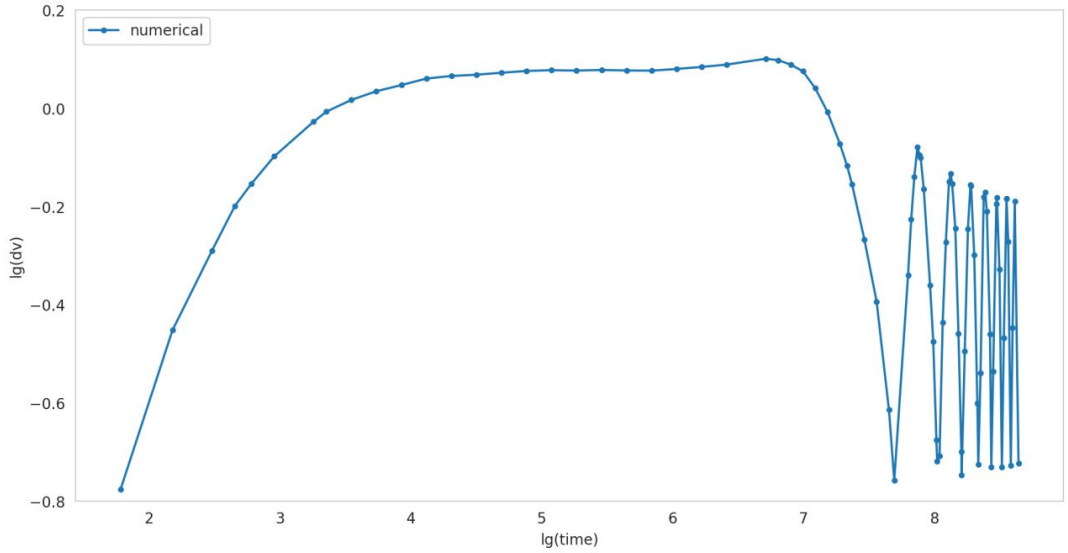

Рис. 1.4.1. Залежність знерозміреного збурення від часу протягом 14 років.

Обмеження в 14 років виникає через неминуче накопичення похибки при тривалих обчисленнях. Можемо розбити цей графік на декілька умовних областей:

- Область лінійного зростання (до години). Коли час до зіткнення прямує до нуля, швидкість необхідна для переведення астероїда на навколоземну орбіту прямує до константи. У запропонованому нормуванні проведено множення на t, що й дає лінійний ріст $\Delta\tilde{v}$. Таким чином видиме в цій області зростання є артефактом обраного нормування;

- Плато (від доби до місяця). Ця ділянка покриває відстані, як за порядком величини значно перевищують радіус Землі та, водночас, значно менші від півосі астероїдної орбіти. Треба надати збурення, яке поверне астероїд на певний кут. Очікувано, що воно буде обернено пропорційне відстані до Землі та прямо пропорційне її розміру. Враховуючи знерозмірення, на рис. 1.4.1. отримуємо константу;

- Область осциляцій (роки). На таких проміжках часу важливим є вплив Сонця, на рис. 1.4.1. спостерігаються коливання. Очікувано, що такі коливання мають період близький до періоду обертання астероїда довкола Сонця, що показано на рис. 1.4.2.

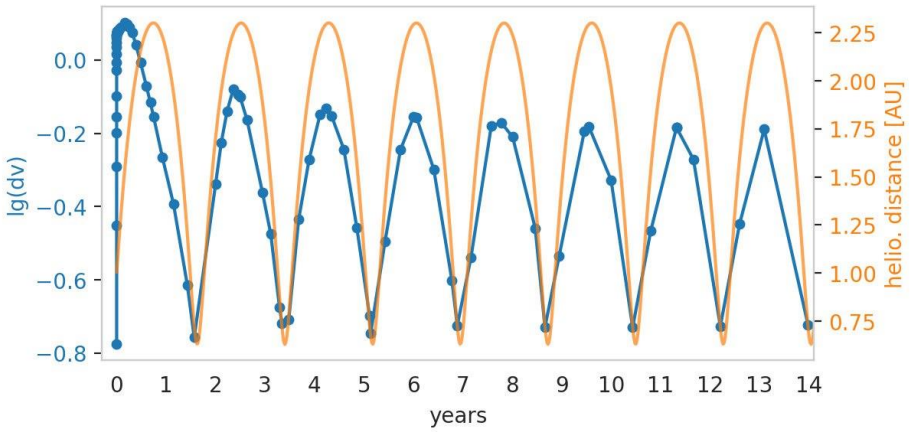

Рис. 1.4.2. Залежність знерозміреного збурення від часу (синя лінія) та геліоцентрична відстань астероїда (оранжева лінія)

З рис. 1.4.2 помітно, що мінімумам відповідають перицентричні збурення, а максимумам – апоцентричні.

Підсумки: Графік знерозміреного збурення не сильно відхиляється від одиниці, що, в свою чергу, обґрунтовує доцільність знерозмірення (*). Таким чином, в нульовому наближенні потрібно прикласти збурення R/t. Наступні наближення – поправки до цього закону.

Зі зрозумілих причин область лінійного зростання не відповідає правилу R/t. Проте, ця область є нецікавою з практичних міркувань астероїдного захисту: складно довести космічний апарат до об’єкту, коли до зіткнення лічені хвилини.

РОЗДІЛ 2. АНАЛІТИЧНІ ОЦІНКИ ВІДХИЛЕННЯ АСТЕРОЇДА

В цьому розділі наведено аналітичні оцінки поведінки залежності $\Delta\tilde{v}(t)$ в різних її областях. Загальна постановка задачі лишається такою ж як і в минулому розділі.

2.1. Наближення 1

Розглянемо найпростіший випадок, де тяжіння від планети та Сонця – відсутні, і, як наслідок, астероїд рухається рівномірно прямолінійно.

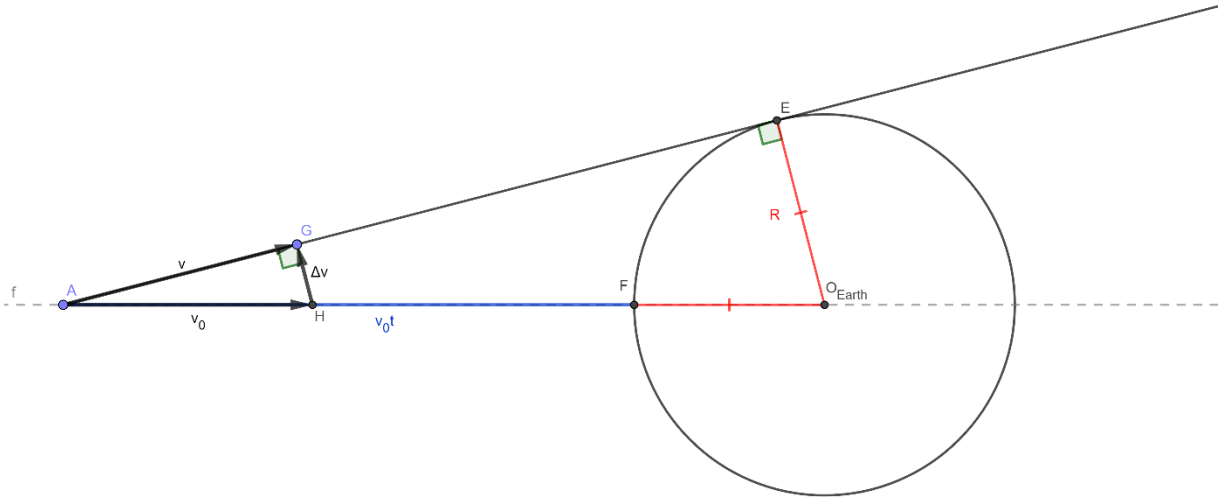

Рис. 2.1.1. Збурення потенційно небезпечного астероїда у 1- му наближенні

Необхідно змінити траєкторію так, щоб замість зіткнення у точці F астероїд пройшов повз Землю по дотичній у точці E. Для цього у певній точці A на відстані $AO = v_0 t + R$ прикладемо збурення швидкості Δv, так щоб $\Delta\vec{v} \parallel \overrightarrow{EO}$. З подібності трикутників ΔAGH та ΔAEO отримуємо наступну пропорцію:

$$\frac{\Delta v}{v_0} = \frac{R}{R + v_0 t} \qquad (2.1.1)$$

$$\Delta\tilde{v} \equiv \frac{\Delta v t}{R} \Longrightarrow \Delta\tilde{v}_1 = \frac{v_0 t}{R + v_0 t} = \frac{1}{1 + \frac{R}{v_0 t}} \qquad (2.1.2)$$

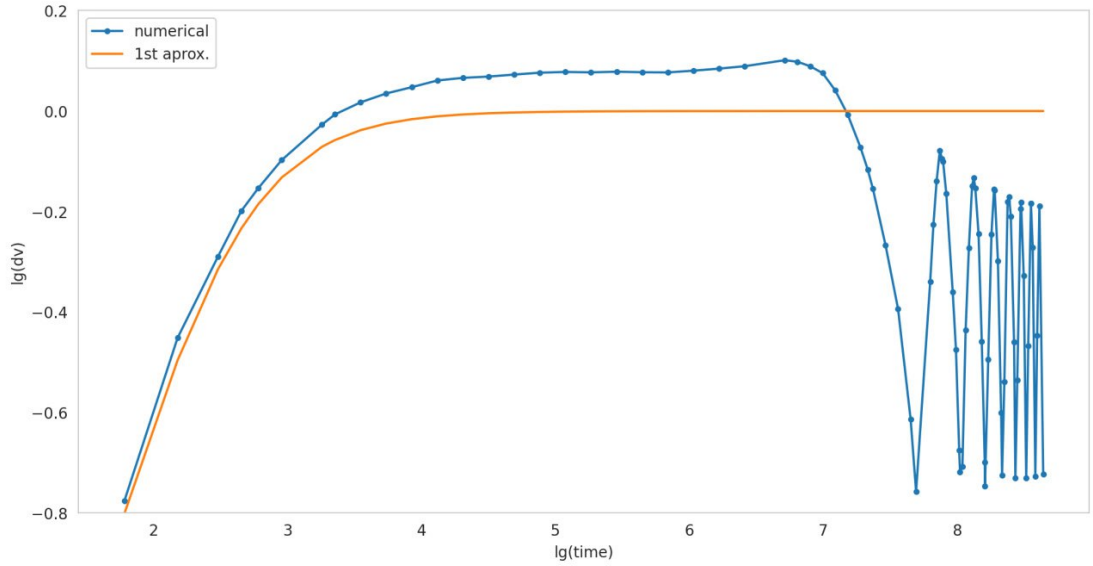

Рис. 2.1.2. Порівняння чисельного моделювання та першого наближення

Висновок: Поведінка графіку Наближення 1 відображає загальний тренд лінійної області та плато. Проте, неврахування зміни швидкості по мірі зближення з планетою призводить до суттєвих розбіжностей між отриманими чисельно та аналітично величинами збурень вже на відстанях порядку радіуса Землі (на часах порядку години).

2.2. Наближення 2

Астероїд рухається під дією сили тяжіння Землі, без врахування тяжіння Сонця. Допоки астероїд не зблизиться із Землею, його рух буде близьким до рівномірного прямолінійного, але, по мірі зближення із планетою, її тяжіння додаватиме збуреній траєкторії кривизни та змінюватиме швидкість падаючого тіла (див. рис. 2.2.1.). Відповідно, прицільний параметр підльоту астероїда слід збільшити, щоб він пройшов повз планету.

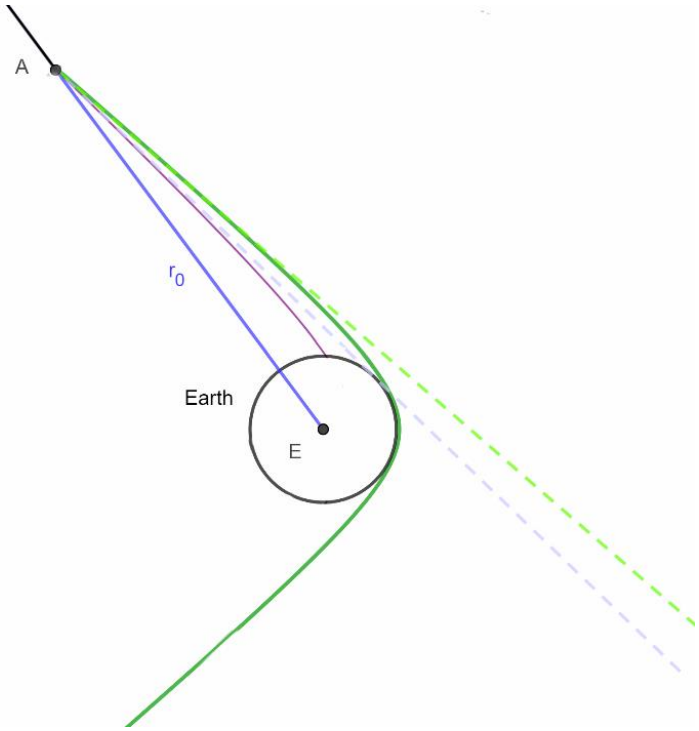

Рис. 2.2.1. Демонстрація впливу тяжіння на траєкторію у 1-му (фіолетова лінія) та у 2-му наближеннях (зелена лінія). Штриховані лінії відповідають напрямкам відхилення

В рамках цього наближення знехтуємо радіальним прискоренням. Розрахуємо відстань, на якій прикладено збурення, за законом рівномірного руху, беручи до уваги скінченні розміри планети:

$$r_0 \approx v_0 t + R \tag{2.2.1}$$

В подальших обчисленнях використано умову малості збурення:

$$v_0 \gg \Delta v \Longrightarrow v_0^2 + (\Delta v)^2 \approx v_0^2 \tag{2.2.2}$$

Запишемо закони збереження енергії та моменту імпульсу для досліджуваної системи:

$$\begin{cases} \dfrac{v_0^2}{2} = \dfrac{v_{max}^2}{2} - \dfrac{GM}{R} \\ b v_0 = R v_{max} \end{cases} \tag{2.2.3}$$

Виразимо v_{max} з першого рівняння системи (2.2.3):

$$v_0^2 = v_{max}^2 - v_2^2 \Longrightarrow v_{max} = \sqrt{v_0^2 + v_2^2} \tag{2.2.4}$$

де $v_2 = \sqrt{\frac{2GM}{R}}$ – друга космічна швидкість.

Отримаємо вираз для прицільного параметра:

$$b = R\sqrt{1 + \frac{v_2^2}{v_0^2}} \tag{2.2.5}$$

Модифікуємо формулу (2.1.1) виконавши в чисельнику заміну $R \to b$ та підставимо у неї отриману в (2.2.5) величину:

$$\frac{\Delta v}{v_0} = \frac{b}{r_0} \approx \frac{R\sqrt{1 + \frac{v_2^2}{v_0^2}}}{v_0 t + R} \Longrightarrow \Delta\tilde{v}_2 = \frac{\Delta v t}{R} = \left(1 + \frac{R}{v_0 t}\right)^{-1}\sqrt{1 + \frac{v_2^2}{v_0^2}} \Longrightarrow$$

$$\Longrightarrow \Delta\tilde{v}_2 = \sqrt{1 + \frac{v_2^2}{v_0^2}}\,\Delta\tilde{v}_1 \tag{2.2.6}$$

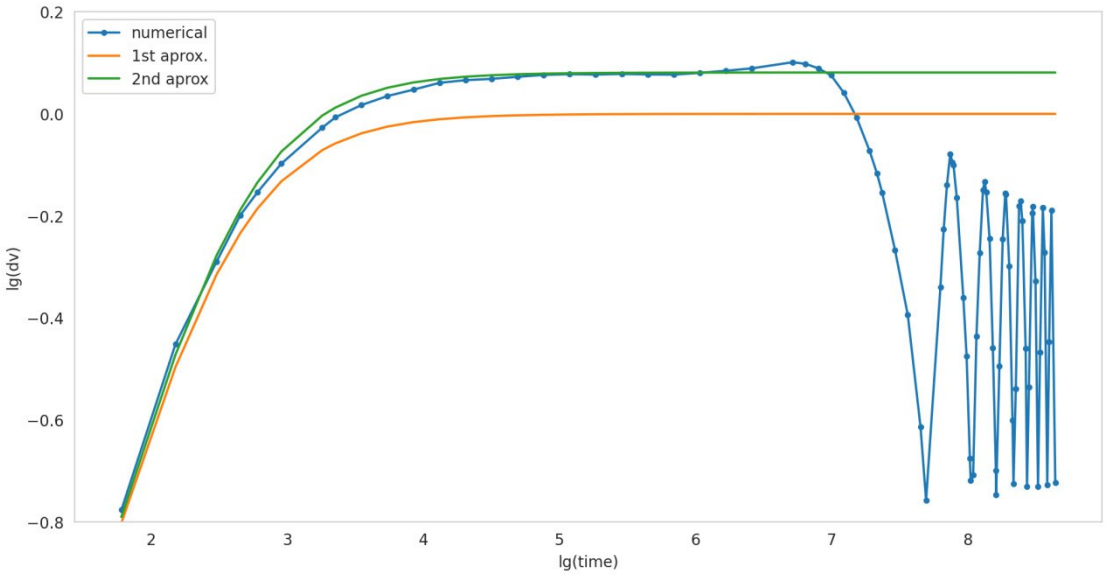

Рис. 2.2.2. Порівняння чисельного моделювання, 1-го та 2-го наближень

Висновок: Врахування тангенціального відхилення траєкторії внаслідок тяжіння Землі суттєво покращило збіжність з комп'ютерною моделлю порівняно із минулим наближенням. Друге наближення є близьким до чисельних розрахунків аж до області осциляцій.

Крім цього, з формули (2.2.6.) можна прийти до висновку, що чим більша початкова швидкість астероїда, то менший вплив на характер його руху має планета. В граничному випадку, коли $v_0 \gg v_2$, вираз для мінімального необхідного відхилення прямуватиме до обчисленого у Наближенні 1.

На рис. 2.2.2 помітне відхилення $\approx 5\%$ від чисельного моделювання при переході між плато та областю лінійного росту. В наступному наближенні усунемо цю розбіжність врахуванням радіального прискорення астероїда гравітацією Землі.

2.3. Наближення 3

Тепер оцінимо необхідну величину збурення, за умови, що астероїд постійно відчуває земне тяжіння. Отримаємо наступні закони збереження:

$$\begin{cases} \dfrac{v^2}{2} - \dfrac{2MG}{r} = \dfrac{v_{max}^2}{2} - \dfrac{2MG}{R} \\ [(\vec{v} + \Delta\vec{v}) \times \vec{r}] = v_{max} R \end{cases} \tag{2.3.1}$$

З закону збереження енергії знайдемо поправку до початкової швидкості пов'язану з тяжінням

$$\frac{v^2}{2} - \frac{2MG}{r} = \frac{v_{max}^2}{2} - \frac{2MG}{R} \Longrightarrow$$

$$\Longrightarrow v^2 = v_{max}^2 - v_2^2 + v_2^2 \frac{R}{r} = v_0^2 + v_2^2 \frac{R}{r} \tag{2.3.2}$$

З формули вище помітно, що при $r \gg R$ ми повернемось до попереднього наближення.

Оскільки в момент збурення $\vec{v} \parallel \vec{r}$, то очевидно, що вклад в момент імпульсу створюється виключно збуренням:

$$[(\vec{v} + \Delta\vec{v}) \times \vec{r}] = [\vec{v} \times \vec{r}] + [\Delta\vec{v} \times \vec{r}] = \Delta v r = v_{max} R \tag{2.3.3}$$

В цьому наближенні ми не можемо знехтувати радіальним прискоренням тіла, тому справедливо наступне:

$$r = v_0 t + s \Longrightarrow s = r - v_0 t \tag{2.3.4}$$

де $\mathrm{s} -$ поправка до відстані на прискорений рух та скінченні розміри планети;

Раніше було отримано вираз для швидкості. З нього знайдемо тривалість незбуреного руху:

$$v(r) = \frac{dr}{dt} = \sqrt{v_0^2 + v_2^2 \frac{R}{r}} \Longrightarrow t = \int_R^r \frac{dr}{\sqrt{v_0^2 + v_2^2 \frac{R}{r}}} \tag{2.3.5}$$

Обчислимо s:

$$\mathrm{s} = r - v_0 \int_R^r \frac{dr}{\sqrt{v_0^2 + v_2^2 \frac{R}{r}}} = r - v_0 \int_R^r \frac{dr}{\sqrt{v_0^2 + v_2^2 \frac{R}{r}}} + \int_R^r dr - \int_R^r dr =$$

$$= R - v_0 \int_R^r dr \left(\frac{1}{v_0} - \frac{1}{\sqrt{v_0^2 + v_2^2 \frac{R}{r}}} \right) =$$

$$= R - R \int_1^{\frac{r}{R}} dx \left(1 - \left(1 + \frac{v_2^2}{v_0^2 x} \right)^{-\frac{1}{2}} \right) \approx$$

$$\approx R + R \int_1^{\frac{r}{R}} dx \frac{v_2^2}{2 v_0^2 x} \approx R + R \frac{v_2^2}{2 v_0^2} \ln \frac{r}{R} \approx$$

$$\approx R + R \frac{v_2^2}{2 v_0^2} \ln \frac{R + v_0 t + f(\ln(R))}{R} \approx R + R \frac{v_2^2}{2 v_0^2} \ln \frac{R + v_0 t}{R} \tag{2.3.6}$$

Підставимо (2.3.4) та (2.3.6) в (2.3.3), звідки отримаємо наступні вирази для збурення Δv та його знерозміреної величини $\Delta \tilde{v}_3$:

$$\Delta v = \frac{v_{max} R}{R + v_0 t + R \frac{v_2^2}{2 v_0^2} \ln \frac{R + v_0 t}{R}} \Longrightarrow$$

$$\Longrightarrow \Delta\tilde{v}_3 = \frac{\sqrt{1+\frac{v_2^2}{v_0^2}}}{\left(1+\frac{R}{v_0 t}+\frac{R v_2^2}{2v_0^3 t}\ln\frac{R+v_0 t}{R}\right)} \qquad (2.3.7)$$

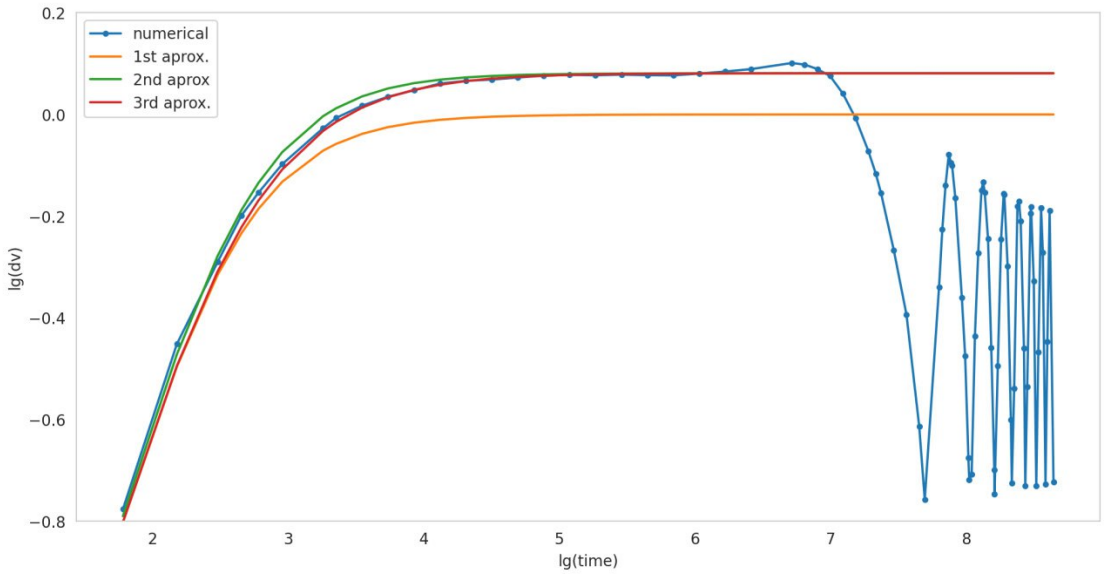

Рис. 2.3.2. Порівняння чисельного моделювання, 1-го, 2-го та 3-го наближень

Висновок: Врахування радіального прискорення астероїда Землею усунуло розбіжність 2-го наближення із моделюванням на часах порядку години. Наближення 3 є справедливим для часів від години до місяця з відсотковою точністю. Понад цей часовий проміжок спостерігаємо суттєві відхилення від чисельних розрахунків, пов'язані з гравітацією Сонця.

2.4. Наближення 4

Для області осциляцій в чисельному моделювання справедливо наступне:

- Відносна зміна орбітальних елементів – мала величина (рис. 1.3.3);
- Для компланарних орбіт найбільш ефективним є збурення, що не змінює схилення астероїда (рис.1.3.4);
- Максимальні та мінімальні необхідні збурення корелюють із проходженням астероїдом своїх апсид (рис. 1.4.2).

Врахуємо тяжіння Сонця (в нашому випадку – фінітність орбіт небесних тіл сонячної системи). Розглянемо збурення в апоцентрі та перицентрі й оцінимо таким чином граничні збурення в області осциляцій. Для цієї задачі введемо наступні обмеження, які узгоджуються з результатами комп'ютерного моделювання:

- Земля та астероїд рухаються в одній площині;
- збурення відбуваються в апоцентрі або перицентрі астероїда без зміни орбітального схилення (такий маневр відомий як гоманівський перехід [3]);
- збурена орбіта близька до початкової, тобто зміна орбітальних параметрів мала в порівнянні з їх початковими значеннями.

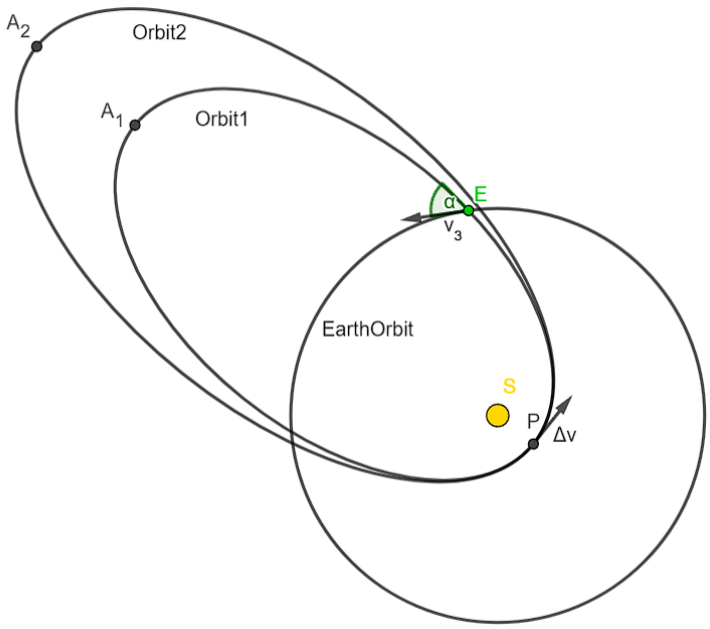

Рис. 2.4.1. Схематичне зображення перицентричного гоманівського переходу

Згідно із законом збереження енергії, для будь-якої кеплерівської орбіти справедливе наступне рівняння:

$$v = \sqrt{GM\left(\frac{2}{r} - \frac{1}{a}\right)} = \sqrt{\frac{2GM}{r}}\sqrt{1 - \frac{r}{2a}} \qquad (2.4.1)$$

Перицентрична швидкість початкової та кінцевої орбіти:

$$V_{1p} = \sqrt{\frac{2GM}{r_1}}\sqrt{1-\frac{r_1}{2a}}\,; V_{2p} = \sqrt{\frac{2GM}{r_1}}\sqrt{1-\frac{r_1}{2a_{\text{H}}}} \tag{2.4.2}$$

Знайдемо величину збурення Δv_p в перицентрі, необхідну для переходу:

$$\Delta v_p = V_{2p} - V_{1p} = \sqrt{\frac{2GM}{r_1}}\left(\sqrt{1-\frac{r_1}{2a_{\text{H}}}} - \sqrt{1-\frac{r_1}{2a}}\right) =$$

$$= \sqrt{\frac{GM}{r_1}}\left(\sqrt{\frac{2r_{2\text{H}}}{r_{2\text{H}}+r_1}} - \sqrt{\frac{2r_2}{r_2+r_1}}\right) = V_1\Delta\left(\sqrt{\frac{2r_2}{r_2+r_1}}\right) \tag{2.4.3}$$

Обчислимо різницю квадратних коренів у (2.4.3) через формулу скінченних приростів $f(r_{2\text{H}}) - f(r_2) \approx f'(r_2)\Delta r_2$

$$\Delta v_p = V_1\Delta r_2 \cdot \left(\sqrt{\frac{2r_2}{r_2+r_1}}\right)'_{r_2} = V_1\Delta r_2\left(\frac{r_2+r_1}{2r_2}\right)^{-\frac{1}{2}} \cdot \frac{2(r_2+r_1)-1\cdot 2r_2}{(r_2+r_1)^2} =$$

$$= V_1\Delta r_2 \frac{r_1}{\sqrt{2r_2(r_2+r_1)^3}} \Longrightarrow \tag{2.4.4}$$

$$\Longrightarrow \Delta r_2 = \Delta v_p \frac{\sqrt{2r_2(r_2+r_1)^3}}{V_1 r_1} \tag{2.4.5}$$

Отримали вираз, який пов’язує приріст апоцентричної відстані Δr_2 та приріст швидкості Δv_p; перицентрична відстань r_1 при цьому лишається незмінною. Перейдемо до приросту великої півосі Δa:

$$\Delta a = \frac{\Delta r_2}{2} = \frac{\Delta v_p}{V_1 r_1}\sqrt{\frac{r_2(r_2+r_1)^3}{2}} \tag{2.4.6}$$

Третій закон Кеплера встановлює зв’язок між відношенням великих півосей та періодів обертання для різних орбіт сонячної системи.

Таким чином, за відомою відносною зміною великої півосі, знаходимо відносну зміну періоду обертання астероїда.

$$\frac{T_{\text{H}}}{T} = \left(\frac{a_{\text{H}}}{a}\right)^{\frac{3}{2}} \Rightarrow 1 + \frac{T_{\text{H}} - T}{T} = \left(1 + \frac{a_{\text{H}} - a}{a}\right)^{\frac{3}{2}} \Rightarrow$$

$$\Rightarrow \frac{\Delta T}{T} = \frac{3}{2}\frac{\Delta a}{a} = \frac{3\Delta v_p}{V_1 r_1}\sqrt{\frac{r_2(r_2 + r_1)}{2}} \qquad (2.4.7)$$

Якщо зіткнення із Землею мало відбутись в момент t, то через наявність ΔT астероїд перетне траєкторію планети в точці, яку вона вже пройшла. Запізнення астероїда внаслідок збурення обчислимо наступним чином:

$$\Delta t = t\frac{\Delta T}{T} = \frac{3t\Delta v_p}{V_1 r_1}\sqrt{\frac{r_2(r_2 + r_1)}{2}} \qquad (2.4.8)$$

Тепер обчислимо достатню затримку, щоб астероїд пройшов повз планету.

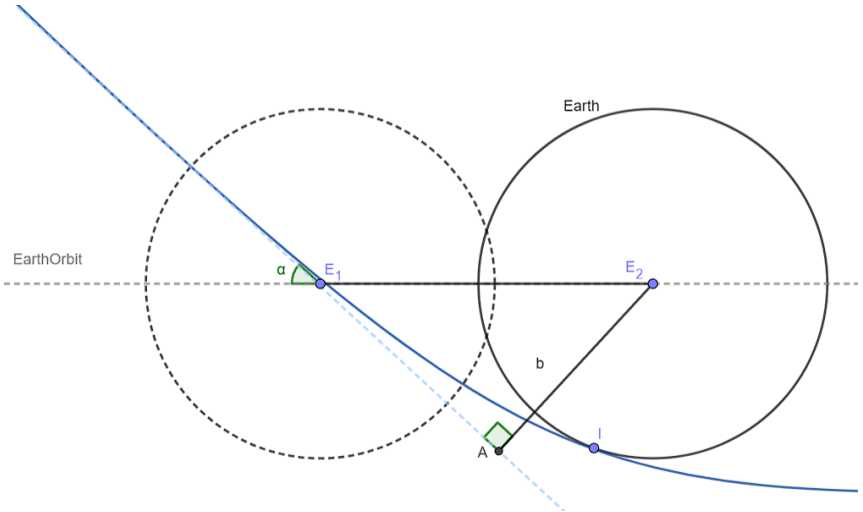

Рис. 2.4.2. Схематичне зображення проходження астероїда, внаслідок спричиненої збуренням затримки

Величину прицільного параметру b встановлюємо з геометричних міркувань (див. рис. 2.4.2.) та прирівнюємо до мінімального безпечного його значення, отриманого в Наближенні 2.

$$b = E_1E_2 \sin\alpha = v_{Earth}\Delta t \sin\alpha = R\frac{v_{max}}{v_0} \Longrightarrow \quad (2.4.9)$$

$$\Longrightarrow \Delta t = \frac{Rv_{max}}{v_0 v_{Earth} \sin\alpha} \quad (2.4.10)$$

Сталу V_1 можна переписати в термінах швидкості руху Землі довкола Сонця:

$$V_1 \equiv \sqrt{\frac{GM}{r_1}} = \sqrt{\frac{GM}{r_{sun}}} \cdot \sqrt{\frac{r_{sun}}{r_1}} = v_{Earth}\sqrt{\frac{r_{sun}}{r_1}} \quad (2.4.11)$$

де r_{sun} – радіус орбіти Землі, тобто 1 астрономічна одиниця.

Підставимо (2.4.10) та (2.4.11) та в (2.4.8) й отримаємо шукане збурення в перицентрі

$$\Delta v_p = \frac{Rv_{max}r_1V_1}{3tv_0v_{Earth}\sin\alpha}\sqrt{\frac{2}{r_2(r_2+r_1)}} = \frac{Rv_{max}}{3v_0t\sin\alpha}\sqrt{\frac{2r_1r_{sun}}{r_2(r_2+r_1)}} \quad (2.4.12)$$

Аналогічно можна отримати формулу для апоцентричного збурення:

$$\Delta v_a = \frac{Rv_{max}}{3v_0t\sin\alpha}\sqrt{\frac{2r_2r_{sun}}{r_1(r_2+r_1)}} \quad (2.4.12)$$

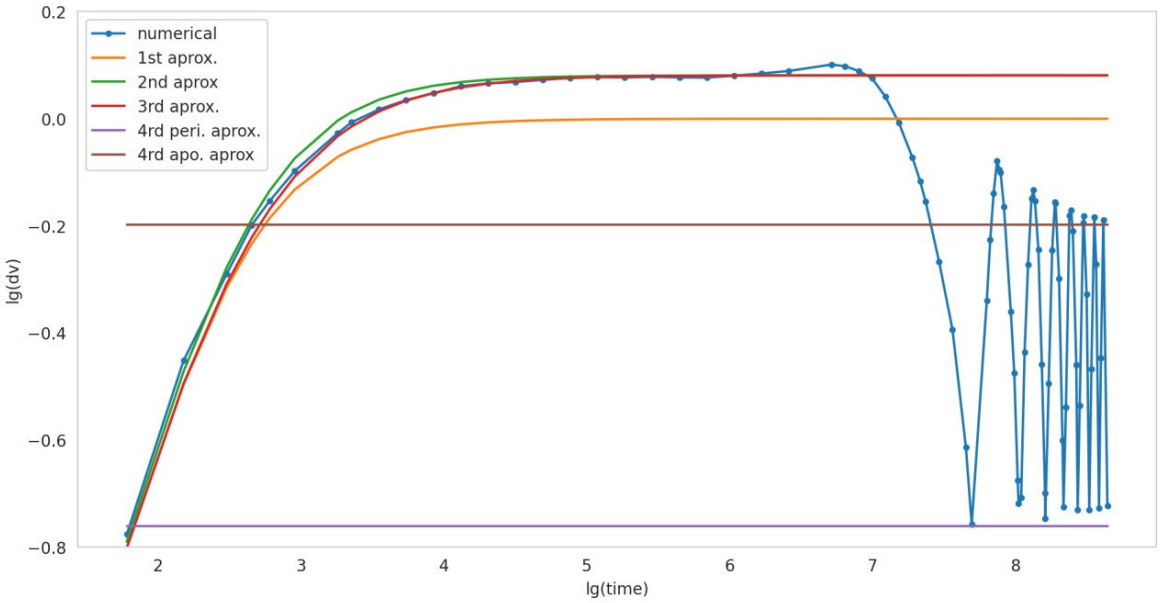

Рис. 2.4.3. Порівняння області осциляцій та
знайдених аналітично амлітудних значень Δv

Висновок: Знайдені величини для апоцентричного та перецентричного збурень є близькими до максимумів та мінімумів чисельного моделювання в області осциляцій. Перехід до області осциляцій (точки близько року) узгоджується з теорією гірше, бо на таких часах суттєвою є не лише зміна періоду обертання, а й зміщення точки перетину орбіт астероїда та Землі.

ВИСНОВКИ

В даній роботі розглянуто задачу про оптимальне відхилення астероїда миттєвим збуренням його швидкості з метою уникнення потенційного зіткнення з Землею.

Для чисельного моделювання задачі було створено пакет допоміжних функцій на основі бібліотеки Rebound. Створені функцій генерували орбіту штучного потенційно небезпечного астероїда й розрахували оптимальний напрямок та мінімальну необхідну величину збурення для переспрямування астероїда. Симуляцію проведено для різних проміжків часу до зіткнення: від 5 хвилин до 14 років. За результатами симуляцій, оптимальний напрямок збурення на малих часах – перпендикулярний до початкової швидкості відносно Землі, а на великих – вздовж або протилежно вектору початкової швидкості відносно Сонця. Величина достатнього збурення в оптимальному напрямку в нульовому наближенні $\approx R/t$ ($R-$ радіус Землі, $t-$ час до зіткнення).

Для опису результатів чисельного моделювання було проведено аналітичні оцінки для різних проміжків часу. Кожне з наближень узгоджується з комп’ютерними розрахунками в межах своєї області застосовності.

Результати роботи можуть лягти в основу подальших досліджень астероїдної небезпеки, які включатимуть базові орбіти з різними параметрами та більші проміжки часу до зіткнення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 .Shepard M. Asteroids: Relics of Ancient Times / Michael K. Shepard. – Pennsylvania: Cambridge University Press, 2015. – 368 с.

2. Morrison D. Overview of Active Planetary Defense Methods / David Morrison // Planetary Defense: Global Collaboration for Defending Earth from Asteroids and Comets / David Morrison., 2019. – (Space and Society). – С. 113– 121.

3. George L. Introduction to Orbital Mechanics. [Електронний ресурс] / Lynnane George. – 2023. – Режим доступу до ресурсу: https://colorado.pressbooks.pub/introorbitalmechanics/.

4. Quantifying the Risk Posed by Potential Earth Impacts / [S. Chesley, P. Chodas, A. Milani та ін.]. // Icarus. – 2002. – №159. – С. 423–432.

5. Influence of the projectile geometry on the momentum transfer from a kinetic impactor and implications for the DART mission / [S. Raducan, M. Jutzi, T. Davison та ін.]. // International Journal of Impact Engineering. – 2022. – №162. – С. 104–147.

6. Dealing with the Impact Hazard / [D. Morrison, A. Harris, G. Sommer та ін.] // Asteroids III / [D. Morrison, A. Harris, G. Sommer та ін.]. – Tucson: University of Arizona Press, 2003. – (The University of Arizona Space Science Series).

7. Mazanek D. Mission Functionality for Deflecting Earth-Crossing Asteroids/Comets / D. Mazanek, S. Park. // Journal of guidance, control, and dynamics. – 2003. – №26. – С. 734–742.

8. Winter O. Gravitational perturbations correlated with the asteroid kinetic impact deflection technique / O. Winter, A. Prado, B. Chagas. // Scientific Reports. – 2022. – №12. – С. 117–121.

ДОДАТОК А

```
def create_collider(enc_time = 2460310.5, sim_time = 0.0, sim_step=3.65,
                    out_step=0.01, integrator='ias15', dv=(0, 0, 0),
                    distance=(6400, 0, 0)):
    """
    Creates body which has a encounter with the Earth at the given encounter time with specified parameters
    Args:
        enc_time: time when the encounter occurs [JD]
        sim_time: time to simulate the orbit (also the epoch of the resulting orbit) [days]
        sim_step: simulation step [days]
        out_step: step for the output data (should be >= sim_step) [days]
        integrator: integrator for the simulations (default: IAS15)
        dv: amount for delta v (perturbation) to apply to a created body
        phi: azimutal angle of perturbation [rad]
        theta: polar angle of perturbation [rad]
        distance: [x,y,z] components of distance fromm the Earth's center (6400 km is the surface) [km]

    Returns:
        pert_orbit: ['a', 'e', 'i', 'om', 'w', 'ma', 'epoch'] of the collider's resulting orbit
        pert_orbit_cart: ['x', 'y', 'z', 'vx', 'vy', 'vz', 'epoch'] of the collider's resulting orbit
        state_vecs: pd.DataFrame(['x', 'y', 'z', 'vx', 'vy', 'vz']) state vectors of the collider for every
        out_step moment of the simulation
        earth_vecs: pd.DataFrame(['x', 'y', 'z', 'vx', 'vy', 'vz']) state vectors of the Earth for every
        out_step moment of the simulation

    """
    # set the sign of the integration step (and the direction of the simulation)
    sim_step = -sim_step if sim_time < 0 else sim_step
    # path to simulation archive
    sa = rebound.Simulationarchive(path_sim_archive)
    sim = sa.getSimulation(t=enc_time, mode="exact", keep_unsynchronized=0)
    sim.testparticle_type = 0
    sim.integrator = integrator
    sim.dt = sim_step
    sim.ri_ias15.min_dt = 0
    # number of bodies except bodies that will be added later
    sim.move_to_hel()
    # get Earth's orbit as a base orbit (heliocentric)
    base_orbit = sim.particles[3] - sim.particles[0]
    base_orbit = pd.Series([base_orbit.x, base_orbit.y, base_orbit.z, base_orbit.vx,
                    base_orbit.vy, base_orbit.vz],
                      index=['x', 'y', 'z', 'vx', 'vy', 'vz'])
    dvx, dvy, dvz = dv
    # add perturbed body to the simulation
    sim.add(x=base_orbit.x + distance[0] / AU_TO_KM, \
            y=base_orbit.y + distance[1] / AU_TO_KM,
            z=base_orbit.z + distance[2] / AU_TO_KM,
            vx=base_orbit.vx + dvx / AU_TO_M * DAY_SEC,  ascending node
          vy=base_orbit.vy + dvy / AU_TO_M * DAY_SEC,
          vz=base_orbit.vz + dvz / AU_TO_M * DAY_SEC,
          m=0.0)

    t_max = sim.t + sim_time
```

```
    Nout = int(abs(t_max-sim.t) / out_step)
    # simulation times and stops
    times = np.linspace(sim.t, t_max, Nout)
    # array to store particle's parameters
    state_vecs = np.zeros((Nout, 6))
    earth_vecs = np.zeros((Nout, 6))
    ps = sim.particles
    #======== simulation cycle =================
    #cycle for integrating till the possible encounter
    # move center to solar system barycenter
    sim.move_to_com()
    for i, time in tqdm(enumerate(times),
                        total=len(times), leave=False):
        sim.integrate(time, exact_finish_time=1)
        h = ps[-1] - ps[0]
        state_vecs[i, :] = h.x, h.y, h.z, h.vx, h.vy, h.vz
        earth_vecs[i, :] = ps[3].x, ps[3].y, ps[3].z, ps[3].vx, ps[3].vy, ps[3].vz

    final_epoch = sim.t
    # get object's orbit after integration
    orbit = ps[-1].orbit(primary=sim.particles[0])

    # return the object's orbit
    pert_orbit = pd.Series([orbit.a, orbit.e, orbit.inc, orbit.Omega, orbit.omega, orbit.M, final_epoch],
                           index=['a', 'e', 'i', 'om', 'w', 'ma', 'epoch'], name='perturbed_orbit')
    # get cartesian heliocentric orbit
    orbit = ps[-1] - ps[0]
    pert_orbit_cart = pd.Series([orbit.x, orbit.y, orbit.z, orbit.vx, orbit.vy, orbit.vz, final_epoch],
                                index=['x', 'y', 'z', 'vx', 'vy', 'vz', 'epoch'], name='perturbed_orbit')
    return pert_orbit, pert_orbit_cart, state_vecs, earth_vecs


def reverse_sim(clones: pd.DataFrame, sim_time=1.0, sim_step=3.65, out_step=10,  integrator='WHFast',
input_type="cart"):
    """
    Simulates the dynamics of the clones in time (old simulation function that is now used only in reverse)
    Args:
        clones: pd.DataFrame of clones initial parameters (cartesian state vectors or keplerian elements)
        sim_time: amount of time to simulate orbits [days]
        sim_step: simulation step [days]
        out_step: step for the output data (should be >= sim_step) [days]
        integrator: integrator for the simulations (default: IAS15)
        input_type: ["cart", "kep"] type of initial parameters to use

    Returns:
        times: np.ndarray with output times of simulation
        state_vecs: np.ndarray with state vectors of bodies for every output time
        earth: state vectors of the Earth for every output time
    """
    # set the sign of the integration step (and the direction of the simulation)
    def _record_encounters(sim_pointer, collided_particle_index):
        """custom function to record encounters between test particles"""
        sim = sim_pointer.contents
        # get the indexes of the collided particles
        idx_p1 = collided_particle_index.p1
        idx_p2 = collided_particle_index.p2
```

```
    print("Collision between particle {} and particle {} at time {}\n".format(idx_p1, idx_p2, sim.t))
    print(f'{len(sim.particles)}')
    return 2

sim_step = -sim_step if sim_time < 0 else sim_step
# path to simulation archive
sa = rebound.Simulationarchive(path_sim_archive)
# start epoch of the simulation is the epoch of the first body
start_epoch = clones['epoch'].iloc[0]
# load solar system from the simulation archive
sim = sa.getSimulation(t=start_epoch, mode="exact", keep_unsynchronized=0)
# for IAS15
sim.ri_ias15.min_dt = 0
# for WHFast
# sim.ri_whfast.corrector = 17
# sim.ri_whfast.safe_mode = 1
sim.collision = "none"
sim.collision_resolve = _record_encounters
sim.testparticle_type = 0
# number of bodies except bodies that will be added later
part_num = sim.N_real
sim.move_to_hel()
if input_type == "cart":
    # adding other bodies to the simulation
    for idx, body in enumerate(clones.iloc):
        sim.add(x=body['x'], # semi-major axis
                y=body['y'], # eccentricity
                z=body['z'], # inclination
                vx=body['vx'], # right ascension of the ascending node
                vy=body['vy'], # argument of perihelion
                m=0.0, # mass
                r=1e-10 / AU_TO_M, # radius
                vz=body['vz'],
                hash=f'clone{idx}') # mean anomaly)
elif input_type == "kep":
    for idx, body in enumerate(clones.iloc):
        sim.add(primary=sim.particles[0], # heliocentric coordinates
                a=body['a'], # semi-major axis
                e=body['e'], # eccentricity
                inc=body['i'], # inclination
                Omega=body['om'], # right ascension of the ascending node
                omega=body['w'], # argument of perihelion
                r=1e-10 / AU_TO_M, # radius
                m=0.0, # mass
                M=body['ma'],
                hash=f'clone{idx}') # mean anomaly)
#===== Important Simulation Parameters ============
sim.integrator = integrator          #
sim.dt = sim_step                    #
ps = sim.particles
#========simulation parametes and arrays init ============
# max time for simulation (days)
t_max = start_epoch + sim_time
# number of checkpoints for data_output
Nout = int(abs(t_max-sim.t) / out_step)
# simulation times and stops
```

```
    times = np.linspace(sim.t, t_max, Nout)
    # array to store particle's parameters
    state_vecs = np.zeros((Nout, 6, len(clones)))
    kep_elems = np.zeros((Nout, 6, len(clones)))
    earth = np.zeros((Nout, 6))
    # move the center of the system to the center of mass
    sim.move_to_com()
    #======== simulation cycle ================
    #cycle for integrating till the possible encounter
    for i, time in tqdm(enumerate(times),
                        desc=f"Integrating {len(clones)} clones for {sim_time} days",
                        total=len(times), leave=False):
        sim.integrate(time, exact_finish_time=1)
        earth[i, :] = ps[3].x, ps[3].y, ps[3].z, ps[3].vx, ps[3].vy, ps[3].vz
        for j, p in enumerate(sim.particles[part_num:]):
            # get object's orbit after integration
            orbit = ps[-1].orbit(primary=sim.particles[0])
            h = p - sim.particles[0]
            state_vecs[i,:, j] = h.x, h.y, h.z, h.vx, h.vy, h.vz
            # return the object's orbit
            kep_elems[i,:, j] = orbit.a, orbit.e, orbit.inc, orbit.Omega, orbit.omega, orbit.M
    print("Done!")
    return times, state_vecs, kep_elems, earth


def simulation(clones: pd.DataFrame, sim_time=1.0, write_time=None, min_step=1e-4, input_type="cart",
disable_tqdm=False):
    """
    Simulates the dynamics of the clones in time
    Args:
        clones: pd.DataFrame of clones initial parameters (cartesian state vectors or keplerian elements)
        sim_time: amount of time to simulate orbits [days]
        sim_step: simulation step [days]
        out_step: step for the output data (should be >= sim_step) [days]
        integrator: integrator for the simulations (default: IAS15)
        input_type: ["cart", "kep"] type of initial parameters to use

    Returns:
        times: np.ndarray with output times of simulation
        state_vecs: np.ndarray with state vectors of bodies for every output time
        earth: state vectors of the Earth for every output time
    """

    def _record_encounters(sim_pointer, collided_particle_index):
        """custom function to record encounters between test particles"""
        sim = sim_pointer.contents
        # get the indexes of the collided particles
        idx_p1 = collided_particle_index.p1
        idx_p2 = collided_particle_index.p2
        p = sim.particles[idx_p2 - sim.N_real]
        return 2
    # set the sign of the integration step (and the direction of the simulation)
    # path to simulation archive
    sa = rebound.Simulationarchive(path_sim_archive)
    # start epoch of the simulation is the epoch of the first body
    start_epoch = clones['epoch'].iloc[0]
```

```
    # load solar system from the simulation archive
    sim = sa.getSimulation(t=start_epoch, mode="exact", keep_unsynchronized=0)
    # for IAS15
    sim.ri_ias15.min_dt = min_step
    sim.integrator = 'ias15'
    sim.collision = "line"
    sim.collision_resolve = _record_encounters
    sim.testparticle_type = 0
    # number of bodies except bodies that will be added later
    part_num = sim.N_real
    sim.move_to_hel()
    clone_hashes = [f'clone{i}' for i in range(len(clones))]
    # adding other bodies to the simulation
    if input_type == "cart":
      for idx, body in enumerate(clones.iloc):
        sim.add(x=body['x'],
                y=body['y'],
                z=body['z'],
                vx=body['vx'],
                vy=body['vy'],
                m=0.0,
                r=1e-10 / AU_TO_M,
                vz=body['vz'],
                hash=f'clone{idx}')
    elif input_type == "kep":
       for idx, body in enumerate(clones.iloc):
        sim.add(primary=sim.particles[0], # heliocentric coordinates
                a=body['a'], # semi-major axis
                e=body['e'], # eccentricity
                inc=body['i'], # inclination
                Omega=body['om'], # right ascension of the ascending node
                omega=body['w'], # argument of perihelion
                r=1e-10 / AU_TO_M, # radius
                m=0.0, # mass
                M=body['ma'],
                hash=f'clone{idx}') # mean anomaly)
    #===== Important Simulation Parameters ==========
    sim.dt = min_step
    ps = sim.particles
    #========simulation parametes and arrays init ============
    # max time for simulation (days)
    t_max = start_epoch + sim_time
    # simulation times and stops
    times = []
    # array to store particle's parameters
    state_vecs = [[] for _ in range(len(clones))]
    earth = []
    # move the center of the system to the center of mass
    sim.move_to_com()
    sim_time = round(sim_time, 3)
    col_time_left, col_time_right = write_time[0] + start_epoch, write_time[1] + start_epoch

    pbar = tqdm(total=abs(sim_time),
            desc=f"Integrating {len(clones)} clones for {sim_time} days",
mininterval=1,
            disable=disable_tqdm)
```

```
    while sim.t <= t_max:
        pbar.update(abs(sim.dt))
        times.append(sim.t)
        sim.integrate(sim.t + sim.dt, exact_finish_time=1)
        if  col_time_left <= abs(sim.t) <= col_time_right:
            earth.append([ps[3].x, ps[3].y, ps[3].z, ps[3].vx, ps[3].vy, ps[3].vz])
            for j, hash in enumerate(clone_hashes):
                try:
                    p = sim.particles[hash]
                    state_vecs[j].append([p.x, p.y, p.z, p.vx, p.vy, p.vz])
                except rebound.ParticleNotFound:
                    state_vecs[j].append([ps[3].x, ps[3].y, ps[3].z, ps[3].vx, ps[3].vy, ps[3].vz])
        if len(sim.particles) == part_num:
            if not disable_tqdm:
                print('No test particles left. Stopping simulation...')
            break

    state_vecs = np.array(list(itertools.zip_longest(*state_vecs,
                        fillvalue=[0,0,0,0,0,0]))).T
    state_vecs = np.moveaxis(state_vecs, 0, 2)
    state_vecs = np.moveaxis(state_vecs, 0, 2)
    earth = np.array(earth)
    times = np.array(times)
    if not disable_tqdm:
        print("Done!")
    return times, state_vecs, earth

def chunk_dataframe(df:pd.DataFrame, n: int):
    chunk_size = len(df) // n
    remainder = len(df) % n
    chunks = [df.iloc[i * chunk_size:(i + 1) * chunk_size] for i in range(n)]
    # Add remainder rows to the last chunk
    if remainder:
        chunks[-1] = pd.concat([chunks[-1], df.iloc[n * chunk_size:]])
    return chunks

def simulation_mp(clones: pd.DataFrame,
                  sim_time=1.0,
                  write_time=None,
                  min_step=1e-4,
                  input_type="kep",
                  n_procs=4):
    if write_time is None:
        write_time = (sim_time * 0.9, sim_time * 1.1)
    clones_chunks = chunk_dataframe(clones, n_procs)
    with Pool(processes=n_procs) as pool:
        results = pool.starmap(simulation, [(clones_chunk, sim_time, write_time,      min_step, input_type) for
clones_chunk in clones_chunks])

    min_dists_all = []
    for result in results:
        times, state_vecs, earth = result
        earth = earth[:len(state_vecs)]
        dists = np.sum(((earth[:, :3, np.newaxis] - state_vecs[:, :3, :]) ** 2), axis=1) ** 0.5 * AU_TO_KM
        min_dists = np.min(dists, axis=0)
        min_dists_all.extend(min_dists)
```

```
    print("Simulations finished.")
    return np.array(min_dists_all)


def min_dist_funct(base_orbit, dv_init, phi, theta, sim_time, write_time=None,
            min_step=-1e-3):
    # create collider with initial parameters
    clone = pd.DataFrame([base_orbit.add(perturbation(base_orbit, dv_init, phi=phi,
            theta=theta), fill_value=0)])
    times, state_vecs, earth = simulation(clones=clone,
                                sim_time=sim_time, # years
                                write_time=write_time,
                                min_step=min_step,
                                input_type='kep', disable_tqdm=True)
    earth = earth[:len(state_vecs)]
    dists = np.sum(((earth[:, :3, np.newaxis] - state_vecs[:, :3, :]) ** 2), axis=1) ** 0.5 *                AU_TO_KM
    dist = np.min(dists)
    return dist - 6400


def bisection(func: Callable, a, b, tol=1):
    """Find root of equation f(x) = 0, x in [a, b]"""
    try:
        res_a = func(a)
        res_b = func(b)
    except KeyboardInterrupt:
        return -1000
    if np.sign(res_a) == np.sign(res_b):
        raise Exception(
            f"root a={res_a}, b={res_b} must be bracketed in the bisection method")
    # get midpoint
    m = (a + b) / 2
    # result
    res_m = func(m)
    if np.abs(res_m) < tol:
        # stopping condition, report m as root
        print(f'\nFound solution: dist={res_m:.3f}, dv={m:.3f} m/s')
        return m
    else:
        print(f'Current dist= {res_m:.3f}, dv={m:.3f} m/s ', end='\r')

    if sign(res_a) == sign(res_m):
        # if m is an improvement on a, we call bisection
        return bisection(func, m, b, tol)
    elif sign(res_b) == sign(res_m):
        # case where m is an improvement on b.
        # Make recursive call with b = m
        return bisection(func, a, m, tol)
```