ALETHEIA
Alpha Chi's Journal of Undergraduate Scholarship
Volume 10 | 2025

Modelling and Filtering Trojan Asteroids in the Solar System: Implementing Dynamical Properties of Trojans into N-Body Integrator "Rebound"

Lina Laghmam

Yale University

***Aletheia* Vol. 10, 2025**
Title: Modelling and Filtering Trojan Asteroids in the Solar System: Implementing Dynamical Properties of Trojans into N-Body Integrator "Rebound"
DOI: 10.21081/ax0415
ISSN: 2381-800X
Keywords: astrodynamics, Trojan, Rebound, integration

Volume 10 | 2025

Title: Modelling and Filtering Trojan Asteroids in the Solar System: Implementing Dynamical Properties of Trojans into N-Body Integrator "Rebound"
DOI: 10.21081/ax0415
ISSN: 2381-800X

Modelling and Filtering Trojan Asteroids in the Solar System: Implementing Dynamical Properties of Trojans into N-Body Integrator "Rebound"

Lina Laghmam

Yale University

Abstract

Trojan asteroids are a significant feature of our Solar System. They represent a large group of asteroids that orbit around Lagrange Points L4 and L5, sharing their planet's orbit around the Sun. They orbit at around 60 degrees ahead or behind the planet, and their relative stability is a key topic of research in astrodynamics. Most Trojan asteroids are recorded around Jupiter, the largest planet of our solar system. However, similar groups of small bodies are known to exist around other planets such as Mars, Neptune, Uranus, and Earth. In this paper, we offer a new way to model Solar System Trojans using the N-Body integration software "Rebound." We also discuss the motion of Jupiter Trojans analytically, in order to establish the necessary criteria to recognize such groups of asteroids in simulations.

We start by determining key starting conditions for the simulation. Then, we pick an accurate integrator as well as determine pertinent time to set the simulation running. At last, we collect data and classify it given known and explained dynamical properties of Trojan asteroids.

Keywords: *astrodynamics, Trojan, Rebound, integration*

Introduction

Asteroids that orbit around Lagrange Points L4 and L5 (60 degrees ahead or behind the planet) are known as Trojans. Lagrange Point L4 and L5 are two of the five known solutions to the restricted three body problem. In other words, they are points of equilibrium for a small body trapped between two massive ones. In our case, the two massive bodies are the Sun and another planet (mostly Jupiter), while the small body is the asteroid. The asteroid is considered to be in equilibrium when it has no velocity or acceleration in the rotating frame. The location of L4 and L5 is therefore found by solving for zero velocity and zero acceleration in the rotating frame (see analytical section). Trojans' original capture into L4 and L5 is still unknown and submitted to four hypotheses (Jian Li, 2023) of which we cite the following: the trojans were captured by a growing Jupiter (not moving outside of Jupiter's orbits after its full formation) or the trojans' orbits decayed into L4 and L5 (being attracted to Jupiter's orbit potentially after its full formation).

Most importantly, Trojan asteroids' location gives them special properties. Indeed, a small body stuck at L4 or L5 will move in what is called a "tadpole" orbit or in "horseshoe" orbit. Their names are pretty representative of their motion: a tadpole orbit looks like an elongated oval, while a horseshoe orbit looks like an incomplete circle (Fig 7 and 12). A tadpole orbit will encompass points L4 and L5, while a horseshoe embraces point L3 in addition to both. Firstly, Trojan asteroids are in tadpole orbits, due to tadpoles' higher stability. Secondly, due to being in semi-equilibrium, Trojan asteroids have been gravitationally stable for billions of years (NASA). Third, Trojan asteroids exhibit an asymmetrical distribution in L4 and L5. They are thus separated in two groups: leading and trailing (respectively L4 and L5), or Greeks and Troys, with a higher density in the Greek group.

Research on Trojan asteroids is led on multiple aspects. It is a vast field of unknowns: what planets harbor Trojans? What are the properties of such planets? How can we develop observational models to detect Trojan asteroids? Do Trojans form at L4 and L5 or move to the points of equilibrium? What is the timescale for such a movement? Are Trojans a consequence of planet formation? (etc.)

Two famous missions are currently concerned with Trojan asteroids: incoming dust images from ALMA and the future Lucy space mission by NASA. The dust images from ALMA might corroborate important hypotheses on Trojan bodies being a consequence of planetary formation (O. Balsalobre-Ruza, 2023). The Lucy mission would allow us to decipher the formation of the solar system by studying the Trojans, due to their age.

Furthermore, scholarship has been exploring the topic of exo-trojan asteroids—Trojans outside of our solar system. They have been described as "unicorns" by Jorge Lillo Box at the Center for Astrobiology (Surrey Comet), and detecting them is a key challenge in astronomy.

Thus, developing a model to recreate the apparition and motion of such asteroids is of capital importance. Other than understanding their formation's conditions, we could refine our criteria to classify asteroids as Trojans. In that sense, by changing initial conditions for the test-particles, we could track which randomized particles end up as Trojans and which do not. Moreover, once a viable simulation is created for the Sun-Jupiter system, we could change the parameters of the star or the planet to test other systems for Trojans.

In this paper, we seek to offer working simulations of the conglomeration of Jupiter Trojan asteroids and the orbits of single Trojans (at Earth and Jupiter), as well as discussion on their dynamical properties. This paper also discusses different criteria to establish a particle as Trojan, derived from said dynamical considerations once more.

Methods

To run our simulation, we used the N-body integration software "Rebound." Rebound is a software package hosted in C but available in Python. It's highly flexible, easy to install, and open source. We worked on Python. "Rebound" offers modules that allow rotations, orbit plotting, fast integration and energy calculations amongst other commodities. It is mainly with those modules that we have been concerned. After integration, we stored our data in Python and plotted it with the MatPlotLib library.

First, we set up the simulation. We import the Sun and Jupiter as our two massive particles. The units are set so that G=1, meaning that the mass of Jupiter must be given in regard to the Sun as well as its semi-major axis and eccentricity (Jupiter is thus imported with a mass of 0.001, a semi-major axis of 1 and an eccentric-

ity of 0.1). The Sun is imported from the NASA JPL Horizons datasystem made available on Rebound. In this unit system, the orbital period is given as twice pi. After importing our two massive particles, we must add our test particles. Test particles represent the asteroids and are considered massless (in comparison to the Sun and Jupiter). They are initialized to avoid any interaction between each other, by setting the Sun and Jupiter as the only two active particles. The test particles are at first given semi-random orbital elements. Their semi-major axes are chosen in a random distribution between 0.9 and 1.1 AU, eccentricity is given by a Rayleigh distribution[1] of peak 0.03, mean anomaly is randomized, longitude of ascending node is randomized, argument of pericenter is randomized, and no inclination is given. The semi-major axes, eccentricity, and inclination are chosen relatively close to Jupiter's orbital elements to set the test particles close to their desired destination on the planet's orbit.

The simulations are run with different times depending on our desired number of orbits for study. Total running time for the simulation is measured and input in the simulation as an integer number of Jupiter's orbits. However, the time stamps at which the positions of the particles are stored do not necessarily correspond to the end or the beginning of one of Jupiter's orbits.

Generally, Trojans tend to appear at around 1000 orbits (admitting a few impure particles), so that is for how long we ran most of our simulations. Positions of the test particles are recorded about once every ten orbits. In our last simulations, we have set the simulation to record the test particles' positions around the last 10% or last 1% of orbits, once per orbit to a hundred times per orbit.

To calculate the positions of the particles, we used the WHFast Integrator module on Rebound. It stands for Wisdom-Holman Fast integrator, and works especially well with long-term orbit integration (Hanno Rein 2015). We set up the time step at 0.001 of the orbit for complete accuracy, as recommended by the WHFast guidelines.

We calculate the orbital elements of the test particles using Rebound's built-in module of functions- calculate. orbits(). We store semi-major axes and eccentricities for the test particles, and true anomaly for Jupiter.

[1] Continuous probability distribution for non-negative random variables.

We store the particle's positions in two frames: inertial and rotational. To set up the rotational frame, we calculate Jupiter's true anomaly at every point in time of the simulation and store it as "l" for "lambda." Then, we apply a rotational matrix with angle lambda to the x and y positions stored in the inertial frame.

$$\begin{bmatrix} cos(l) & sin(l) \\ -sin(l) & cos(l) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} xrotated \\ yrotated \end{bmatrix} \quad (1)$$

We plot the characteristic tadpole orbits in a rotating frame, but we generally plot larger Trojan conglomerations in an inertial frame.

We use the Rebound OrbitPlot to plot Jupiter's orbit specifically. Note that the Rebound OrbitPlot function plots Jupiter's orbit in an inertial frame only. In the rotational frame, Jupiter does not move which renders its orbit obsolete for study.

We also plot a graph of eccentricity versus semi-major axis values or our particles to show the concentration around our desired Trojan values at relevant points of the simulation.

Finally, to guarantee we only conserve relevant Trojan asteroids, we must input filters in our code. In order to accomplish that, we must allow for a dynamical consideration of Trojan asteroids. The section below describes Trojans' properties studied in order to establish our filters.

Dynamical Consideration of Trojan Asteroids' Characteristics

Our first filter on the particles acts in regard to the geometrical position of the Trojan Asteroids on Jupiter's orbit in relation to Lagrange points 4 and 5. We must understand where the Trojans are situated geometrically in regard to the Sun and Jupiter in order to translate that criteria into our simulation. To that effect, we establish the position of L4 and L5 in the Sun-Jupiter system.

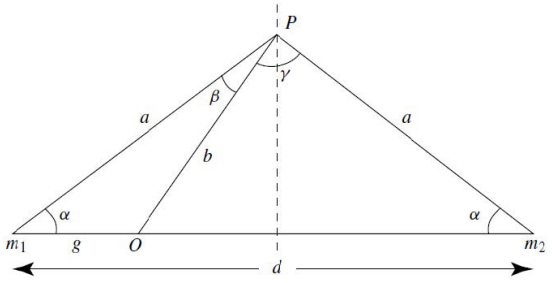

Fig 1.1 (Murray, 1999)

O here represents the center of mass of the system, M_1 is the Sun, m_2 is Jupiter, and P is a test-particle. The triangle is isosceles, as by propriety of O being the center of mass. In order to find the point P of equilibrium, we must balance the centrifugal acceleration of particle P with the force directed towards the center of mass. Thus (Murray 1999):

$$n^2 b = F_1 \cos(\beta) + F_2 \cos(\gamma) \quad (2.1),$$

which, with further trigonometry, gives:

$$n^2 = G(M_1 + m_2) / a^3 \quad (2.2)$$

As we are in the rotating frame, we have another expression for the centrifugal acceleration. We extract it from U, in which the term $(x^2 + y^2)$ is the centrifugal potential and the other term is the gravitational potential. In U, and further in the paper, u_1 and u_2 are expressed as such:

$$u_1 = 1 - u_2 \quad (3.1)$$

$$u_2 = m_2/(M_1 + m_2) \quad (3.2)$$

$$U = \frac{n^2}{2}(x^2 + y^2) + \frac{u_1}{r_1} + \frac{u_2}{r_2} \quad (3.3)$$

Rearranging, we get:

$$n^2 = G(M_1 + m_2)/d^3 \quad (4)$$

which means that $a = d$. In which case, P is a vertex of an equilateral triangle that has M_1 and m_2 as its basis. Ultimately, L4 and L5 are the vertices of the two equilateral triangles that have for basis points the Sun and Jupiter. Trojans orbit around them, meaning that setting up a filter for particles approximately 60 degrees from Jupiter as their last position guarantees that we obtain Trojans. However, as their motion extends to some degree from L4 and L5, we risk eliminating Trojan particles with the filter. It is not optimal, so we move on to another property.

We suggest the implementation of a second filter that acts on the orbital elements of the particle. Indeed, Trojan asteroids share Jupiter's orbit around the sun: their semi-axis must be close to Jupiter's and their eccentricity decently similar. To ensure that our test particles stay close enough to Jupiter's orbit that they remain interesting for study, we can turn to Hill's equations. Hill's equations restrict the study of the motion of a small particle orbiting around the second body of greater mass within a sphere (Fig 1.2) of radius:

$$\Delta H = \left(\frac{u_2}{3}\right)^{1/3} \quad (5.1)$$

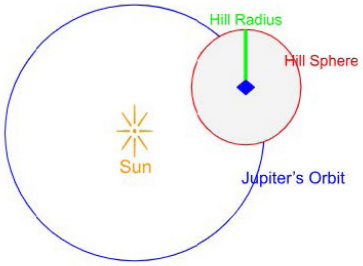

Fig 1.2

In our case, the sphere of interest would be the sphere of radius 0.001 surrounding Jupiter. Beyond it, the test particle would independently orbit the Sun (we consider it to be "kicked out" of the simulation). The formula for the radius of our test particle around the planet (called Hill Radius) is given by:

$$RHill = a(m_2/3M_1)^{1/3} \quad (5.2)$$

We translate this criteria to the particle's semi-major axis by setting it to "a_{Jup} +- *RHill*," or in other words, to Jupiter's semi-major axis give or take the Hill Radius. In our simulation's units with $a_{Jup} = 1$, we get the particle as lodged between semi-major axes 0.97 and 1.03. We add in another criteria for eccentricity (e lodged in between 0.05 and 0.1) to avoid egregiously eccentric orbits. This filter works in theory, and its apparent yield is discussed later in the paper.

Our third filter acts on the motion equations of the particle. A particle affected by two massive bodies admit the following motion equations (Murray 1999) with U equation (3.3):

$$x'' - 2ny' = \frac{dU}{dx} \quad (6.1)$$

$$y'' + 2nx' = \frac{dU}{dy} \quad (6.2)$$

$$z'' = \frac{dU}{dz} \quad (6.3)$$

Multiplying (6.1) by x', (6.2) by y' and (6.3) by z', adding them up and integrating, we obtain:

$$x'^2 + y'^2 + z'^2 = 2U - CJ \quad (7.1),$$

with CJ a constant of integration. This is equivalent to:

$$CJ = n^2(x^2 + y^2) - v^2 + 2\left(\frac{u_1}{r_1} + \frac{u_2}{r_2}\right) \quad (7.2),$$

with CJ called the Jacobi Constant.

Trojan asteroids orbit around points of equilibrium. Thus, the Jacobi constant becomes interesting as we can set the velocity to zero and get the following equation: CJ = 2U. This equation binds the motion of our test particle to desired areas (L4 and L5). For the Sun-Jupiter system for example, we can calculate the desired Jacobi constant at Lagrange L4 and L5 to be 3 - u_2, so about 2.999.

The filter calculates the Jacobi constants of our test particles. If they are approximately equal to the Jacobi constant at L4/L5, then the test particle must be a Trojan, as the Jacobi provides the necessary bounds to the notion.

Implementing the Filters in the Simulation

To implement the first filter, we start by defining an angle function that calculates the angle clockwise between two points given their x and y coordinates. We then create an array containing Jupiter's position. Afterwards, we implement a three-dimensional array that takes in the x and y coordinates of the particles at different times. We set up a list for the angles. We also set up a loop with the number of particles as indexes that calculates the angle between Jupiter and each particle at its last position. We store said angles and set up a filter loop. The filter loop transfers the x and y coordinates of the particle onto new arrays (x_trojan and y_trojan), if the angle between the last position of the particle and Jupiter is approximately 60 degrees (58-63 degrees or 55-65 degrees). See Appendix A for code.

To implement the second filter, we ask Rebound to calculate the semi-major axis and eccentricity of each particle using their calculate.orbit() module. We store the values in lists. As a reminder, for a particle to be a Trojan, it must be situated in Jupiter's Hill Sphere, which in our simulation's units translates to a semi-major axis in the range 0.97 to 1.03 (comparatively to a_jup=1). We calculate the x and y positions for each particle at our given times. Then, we set up a filter loop that transfers the x and y coordinates of a particle onto new arrays (x_trojan and y_trojan) if the semi-major axis of the particle is contained in the Hill criteria's range and its eccentricity is reasonable (0 to 1, or 0.05 to 0.1 if we are critical). See Appendix B for code.

To implement the third filter, we define a function to calculate the Jacobi constant of each particle (Rebound Documentation: Poincare Surface of Section). Then, we set a list for Jacobi constants. We start a loop to calculate the Jacobi Constant for each particle and store it. Then, we start our filter loop that transfers the x and y coordinates of a particle onto new arrays (x_trojan and y_trojan) if the Jacobi Constant of the particle is close to 2.999. We chose a range of 2.95 to 3. See Appendix C for code.

Results

Our results were plotted both in the inertial and rotating frame because, first, there was a large focus on obtaining the Trojan conglomerate—which was feasible in an inertial frame—and because, second, the rotational frame was especially useful in considering the motion of a single particle.

Conglomeration of Trojan—Plotting All Particles at the Last Position

The first plots we propose are a side-by-side comparison of a simulation run at 100 (Fig 2.1) and at 1000 orbits (Fig 2.2). Note that the filters are not implemented in this simulation yet, which means that there are a few impure particles in the last plot, despite the clear conglomerate of Trojans at L4 and L5. In red is Jupiter's orbit, and the blue dots are the test particles.

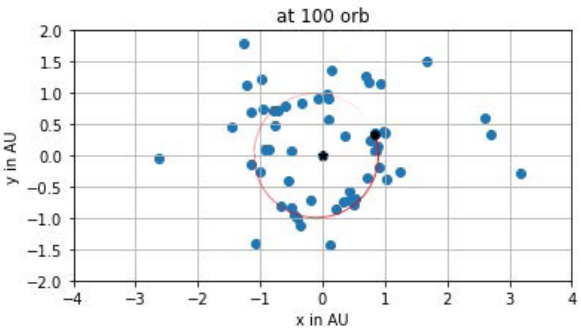

Fig 2.1.1

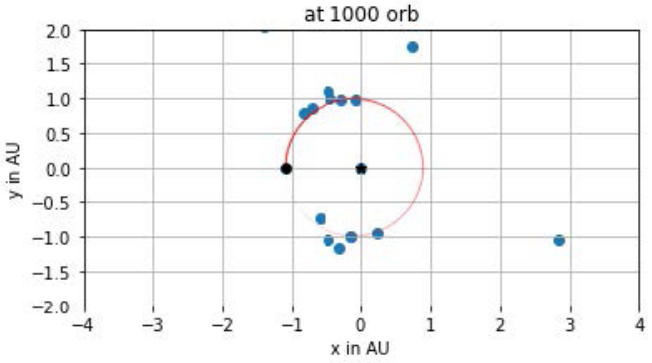

Fig 2.1.2

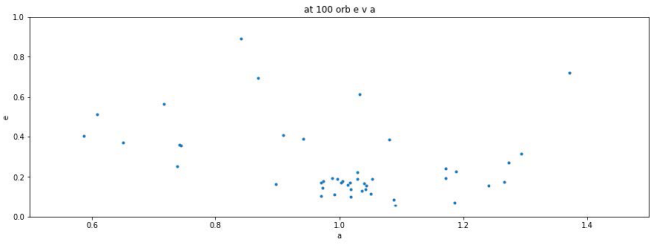

Fig 2.2.1

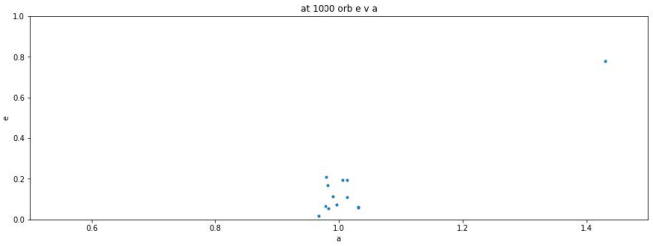

Fig 2.2.2

We have also plotted the eccentricity in function of the semi-major axis at both points of the simulation. As expected, at 1000 orbits (with the Trojan distribution), there is a concentration of particles at the RHill criteria range. In this distribution, there is no particular difference in density between the leading and trailing group. However, another run of the same simulation settings (admittedly not the exact same initial conditions to the randomization of some orbital elements for the test particles), we get the following plot for 1000 orbits (Fig 3):

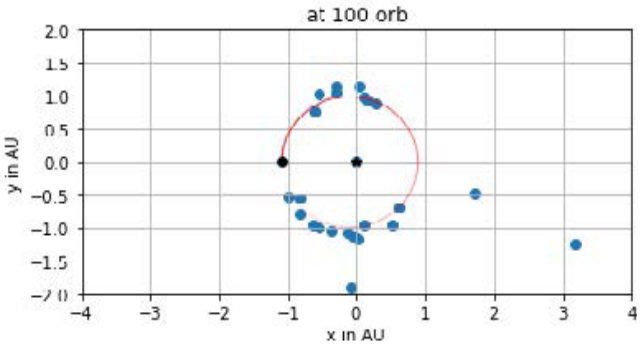

Fig 3

This clearly demonstrates a higher density in one of the groups, which is expected observationally around Jupiter.

Before instoring filters, we have tried to run the simulation for longer periods of time. At 2000 orbits, we obtain the following plots, and it seems safe to assume that non-Trojans are kicked out of the simulation completely (Fig 4.1 and 4.2).

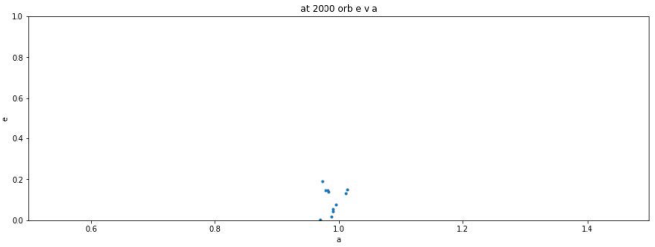

Fig 4.1

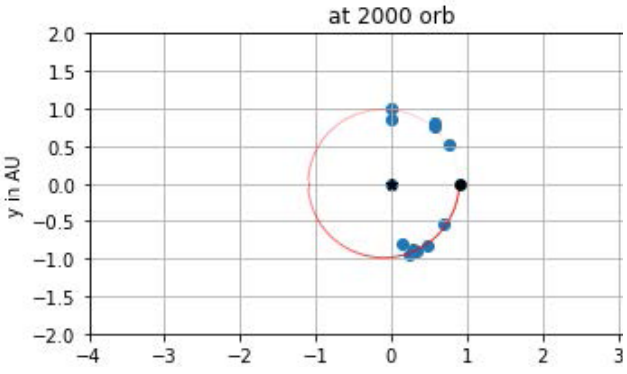

Fig 4.2

Inputting our RHill filter onto the simulation allows us to get rid of impure particles at 1000 orbits to obtain the following Trojan Distribution (Fig 5). (The colors were created with a particle loop; they do not matter beyond aesthetic value.)

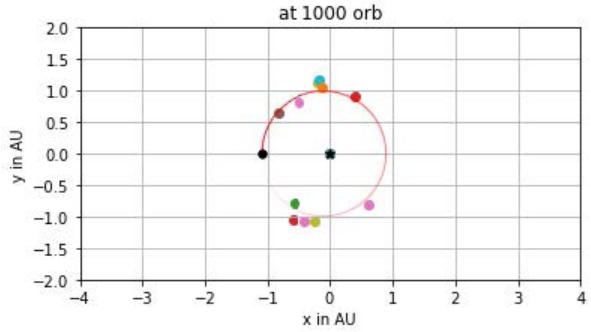

Fig 5

Motion of a Trojan Asteroid—Plotting One Particle at All Times

After plotting conglomerates of Trojans, we have an interest in plotting the motion of a single Trojan asteroid. In the inertial frame, the motion is dominated by the Sun, meaning that the asteroid moves in a semi-constant ellipse around it (Marzazi et al.), although it admits oscillations due to Jupiter. A Trojan's motion in non-rotating frame looks like the plots below (Fig 6):

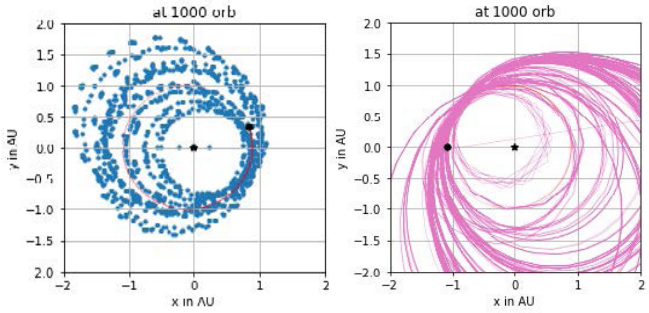

Fig 6

The ellipses of different semi-major axes and eccentricities are created due to the variation of the Trojan's semi-major axes and eccentricity as a function of time due to perturbations by Jupiter. The plots do not show characteristic motion because the motion of the particle is overwhelmed by the Sun's influence in a non rotating frame. These plots are not exploitable, so we switch to the rotating frame in order to obtain the characteristic tadpole orbits (Fig 7.1.1 and 7.2.1 show them around the sun, Fig. 7.1.2 and 7.2.2. are close-ups).

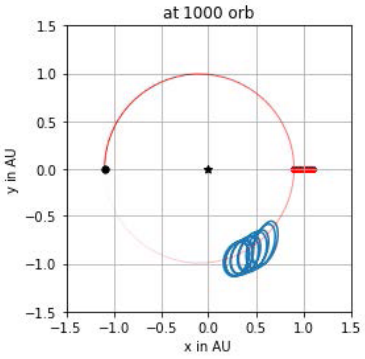

Fig 7.1.1

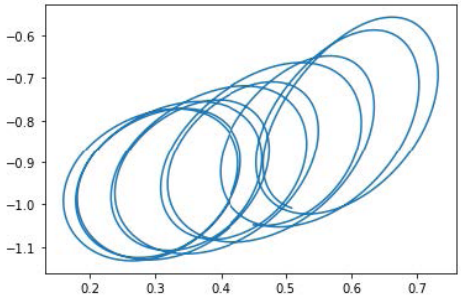

Fig 7.1.2

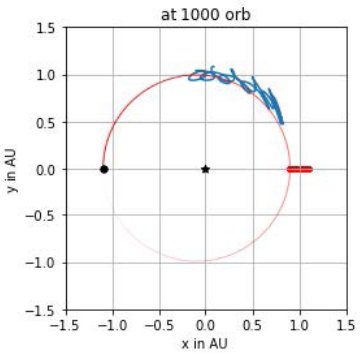

Fig 7.2.1

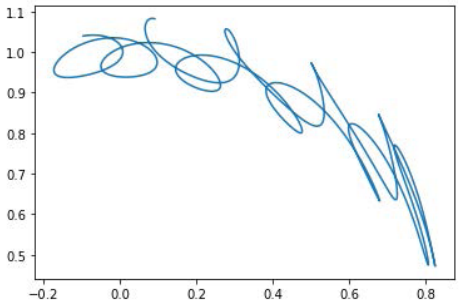

Fig 7.2.2

The two four plots (Fig 7.1 and 7.2) represent the tadpole orbits of Trojans from the Trojan group (at Lagrange point 5). The following two plots are of the tadpole orbit of a Trojan from the Greek camp (at Lagrange point 4). Their equations of motion can be written.

Equation of Motion for Arbitrary Trojan Asteroid

We can start by setting up a small displacement (X, Y) from the Lagrange points 4 and 5. After substituting in our equation of motion (U), we get a set of linear differential equations that involve our position and velocity that we can write in Matrix Form (Murray, 1999):

$$\begin{bmatrix} X' \\ Y' \\ X'' \\ Y'' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ Uxx & Uxy & 0 & 2 \\ Uxy & Uyy & -2 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ X' \\ Y' \end{bmatrix} \quad (8.1)$$

which solves to (with α, β constants and ƛ the eigenvalues of the Matrix equation) (Murray, 1999):

$$X = \sum_{j=1}^{4} \alpha_j e^{\lambda_j t} \quad (8.2)$$

$$X' = \sum_{j=1}^{4} \lambda_j \alpha_j e^{\lambda_j t} \quad (8.3)$$

$$Y = \sum_{j=1}^{4} \beta_j e^{\lambda_j t} \quad (8.4)$$

$$Y' = \sum_{j=1}^{4} \beta_j \lambda_j e^{\lambda_j t} \quad (8.5)$$

The constants are solved with initial conditions boundaries (velocities to zero and initial position); and the eigenvalues are found with the characteristic polynomial. For our Trojan asteroids, we set the initial position to (0.499,-+0.866) and the initial velocity to (0,0). Uxx is 0.75, Uyy is 2.25 and Uxy is roughly +-0.01. We solve for the eigenvalues with the characteristic polynomial on Wolfram Alpha:

$$\lambda^4 + \lambda^2 + 0.0067 = 0 \quad (8.6)$$

$$\lambda_{1,2} = +- \frac{1}{20} i \sqrt{200 - \frac{\sqrt{973027}}{5}} \quad (8.7)$$

$$\lambda_{3,4} = +- \frac{1}{20} i \sqrt{200 + \frac{\sqrt{973027}}{5}} \quad (8.8)$$

The constants give a system of 8 equations with our initial conditions (Appendix D) which give the following solutions (on calculator):

α	Value
α_1	$0.522 - 0.845i$
α_2	$0.52344 + 0.845i$
α_3	$-0.52267 + 0.07014$
α_4	$-0.52279 - 0.06962$

(8.9)

β	Value
β_1	$-0.397 + 0.49i$
β_2	$-0.398 - 0.04i$
β_3	$-0.035 - 0.04i$
β_4	$-0.035 + 0.04i$

(8.10)

We notice that the solutions are in conjugate pairs—barr integration approximations (Murray, 1999)—and that we can establish the following motion equation using Euler's equation[2].

$$X(t) = 1.044cos(0.082t) + 1.044cos(0.966t) - 1.044sin(0.082t) - 1.044sin(0.996t) \quad (9.1)$$

$$Y(t) = 0.794cos(0.082t) + 0.796cos(0.966t) - 0.07sin(0.082t) - 0.07sin(0.082t) \quad (9.2)$$

[2] e^(itheta) = cos(theta)+ isin(theta)

These equations plot to be a tadpole orbit, as we plotted them at a time t=30 and t=100 (Fig 8.1 and Fig 8.2).

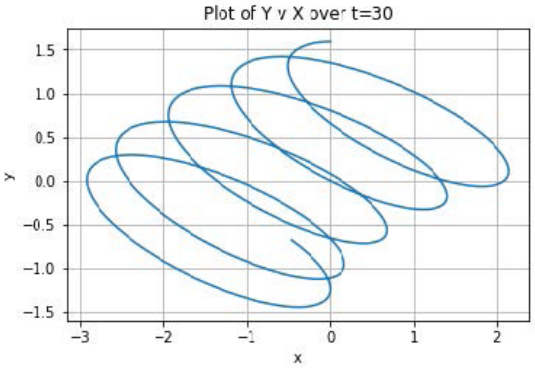

Fig 8.1

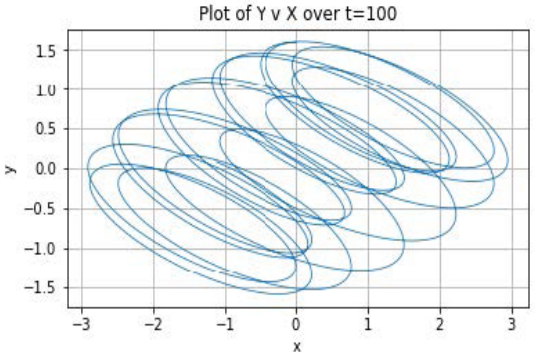

Fig 8.2

A horseshoe orbit has not been observed near Jupiter yet (Marzazi et al.) due to horseshoes' higher instability.

Criteria Filter Yield

Apart from the graphs of conglomeration and single particle Trojan, we offer to discuss the yield of our filters. We note that we have tested out the yield of our filters over one simulation, due to time restraints, thus the results are biased. Figure 9 shows the conglomeration of Trojans (all particles that survive at last time) obtained from different filters. Fig 9.1 was obtained from the Jacobi filter, Fig 9.2 from the angles filter, and Fig 9.3 from the orbital element filter. The red dash is the position of Jupiter at all times during the simulation.

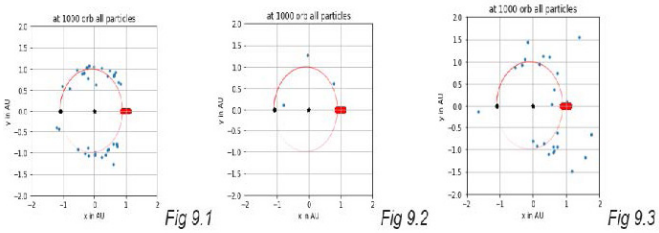

Fig 9.1 *Fig 9.2* *Fig 9.3*

The Jacobi filter gives the most recognizable Trojan distribution, the angle filter fails, and the elliptical filter is not entirely convincing. Furthermore, having tested out manually the single particles that followed (printing their plot through time to check whether or not they have Trojan properties), we obtain the following data (Fig 10):

Filter	Criteria	Initial Number of Particles	Number of Orbits	Particles Surviving the Filter	Trojans	Accuracy of filter (Trojans/ Surviving)
Jacobi	Jacobi constant between 2.99 and 3	1000	1000	132	70	53%
Orbital Elements	A between 0.97 and 1.03, and e between 0.05 and 1	1000	1000	27	2	7.4%
Angles	Angle between 59 and 61 degrees	1000	1000	10	2	20%

Fig 10

As it stands from the tests on the simulation, the Jacobi criteria is by far the most proficient. However, we must stress that the data is heavily skewed, and more tests should be run to reach a viable conclusion. Nevertheless, the Jacobi criteria functioning best goes with our intuition as it is the least empirical.

Next, we move on from the Sun-Jupiter system onto Sun-Earth. On Sun and Earth, the criteria must be recalculated to fit Earth's properties (mass ratio of 10^{-6}), but the idea remains the same. As a matter of fact, to simulate other planets that might have Trojans, one need only recalculate the criteria with the new mass ratio and launch the simulation. We have simulated Earth Trojans: the Hill radius becomes 0.01, the Jacobi constant

remains at 2.999 (NASA JPL), and the angle remains 60 degrees. Fig 11 shows the tadpole orbit of an Earth Trojan: Fig 11.1 shows it on the larger orbit around the Sun (the star symbol), and Fig 11.2 is a close-up. For comparison, Fig 12 shows a near-Earth horseshoe orbit (another kind of orbit present at the co-orbital of planets); 12.1 is the orbit shown on the larger one around the Sun, and 12.2 is a close-up.

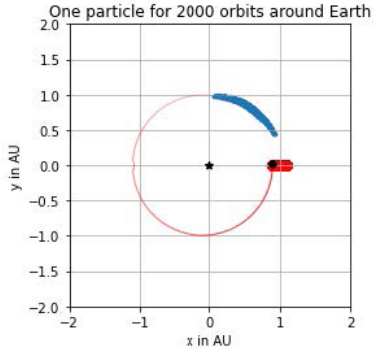

Fig 11.1

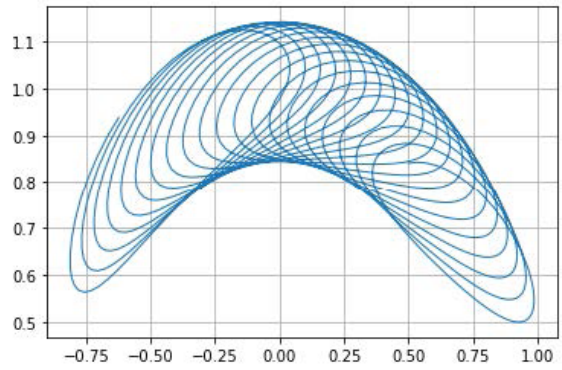

Fig 11.2

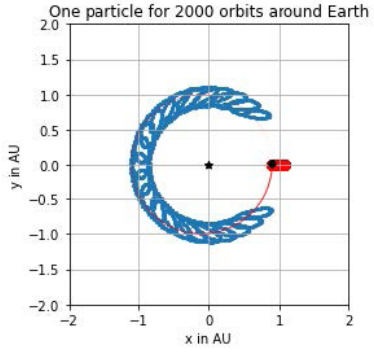

Fig 12.1

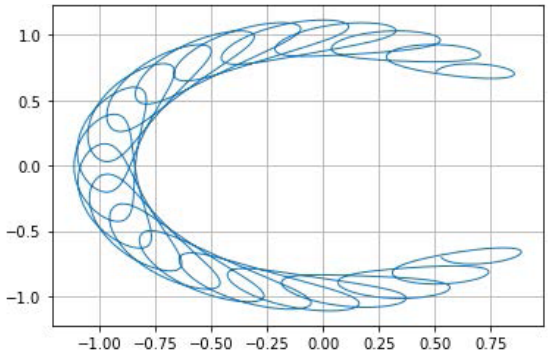

Fig 12.2

Horseshoe orbits have different starting conditions compared to tadpoles; they tend to start further away from Lagrange points, but otherwise their dynamics are pretty similar.

Discussion

We must acknowledge some bias in our data. First, the simulation only takes into account two massive bodies: the Planet and the Sun. Any other interaction with surrounding bodies are neglected. Second, inclination was simplified to zero. Third, the simulations could never run the same way twice, as we randomized the test particles every time. Comparing two simulations or "the same simulation" that have been run separately is always subject to that initial parameter difference. On the assessment of our criteria, this issue is vital. The yield rate for the Jacobi filter might be higher because of the simulation's original setting. Admittedly, the percentage is much higher for Jacobi than others, which could mean that bias is weak. In the future, the filters should be tested on at least tens of different simulations to make sure that the Jacobi one is the most performant. Fourth, a flaw might reside with the choice to run the simulations for 1000 orbits most times. Running them for shorter or longer might be judicious in determining the exact apparition moment of Trojan conglomerations (admittedly, plotting the test particles at different orbits works as well). Finally, we have not tested the simulation for other planets than Jupiter or Earth in our Solar System or outside of it. In the future, running the simulation with other mass ratios or stars would be the best way to discover new potential Trojans in other systems.

Conclusions

By exploring models of Trojan asteroids in our Solar System, we obtained and recreated the following information:

1. that Trojans conglomerate at Lagrange points 4 and 5;
2. that Jupiter Trojans conglomerate at around 1000 orbits;
3. that Trojans remain within Hill's Sphere of their planet, which translates to their orbital properties;
4. that Trojans have a Jacobi constant that remains close to the Jacobi constant at the Lagrange points 4 and 5;
5. that Trojans' singular motion only appears in rotating frame;
6. that Jupiter Trojans orbit in tadpole motion, for which we recreated a typical motion equation;
7. that a Jacobi filter works best to classify particles as Trojans;
8. that Earth Trojans appear in tadpole orbits; and
9. that Earth Trojans mostly appear at around 2000 orbits in the simulation.

This paper offered a basis to run simulations on Trojan asteroids in the hope of it remaining consistent with any planet within and outside the solar system. In the future, we would expect to narrow down the criteria to increase the yield rate and test the simulation with other planetary properties.

Acknowledgements

This work was conducted thanks to the Institute of Astronomy at Cambridge University under the guidance of Professor M. Wyatt and Mr. S. Young.

References

AstroVaporia. (n.d.). Hill radius. http://astro.vaporia.com/start/hillr adius.html

Hanno Rein, Daniel Tamayo, WHFAST: a fast and unbiased implementation of a symplectic wisdom–Holman integrator for long-term gravitational simulations, Monthly Notices of the Royal Astronomical Society, Volume 452, Issue 1, 01 September 2015, Pages 376–388. https://doi.org/10.1093/mnras/stv1257

Jian Li, Zhihong Jeff Xia, Fumi Yoshida, Nikolaos Georgakarakos and Xin Li, Asymmetry in the number of L4 and L5 Jupiter Trojans driven by jumping Jupiter, Astronomy & Astrophysics, 669 (2023) A68, DOI: https://doi.org/10.1051/0004-6361/2 02244443

Marzari, F., Scholl, H., Murray, C. D., & Lagerkvist, C. (2002). Origin and evolution of Trojan asteroids. In W. F. Bottke Jr., A. Cellino, P. Paolicchi, & R. P. Binzel (Eds.), Asteroids III (pp. 725–738). University of Arizona Press, https://www.researchgate.net/pu blication/241538277_ Origin_and_Evolution_of_Trojan_Asteroids

Murray, C. D., & Dermott, S. F. (1999). Solar System Dynamics, Cambridge University Press. https://doi.org/10.1017/CBO9781139174817

O. Balsalobre-Ruza, I. de Gregorio-Monsalvo, J. Lillo-Box, N. Huélamo, Á. Ribas, M. Benisty, J. Bae, S. Facchini and R. Teague, Tentative co-orbital submillimeter emission within the Lagrangian region L5 of the protoplanet PDS 70 b, Astronomy & Astrophysics, 675 (2023) A172, DOI: https://doi.org/10.1051/0004-6361/202346493

P.A. News Agency, Astronomers discover exoplanet may have sibling sharing orbit, Surrey Comet, July 2023. https://www.surreycomet.co.uk/news/national/23665972.astronomers-discover-exoplanet-may-sibling-sharing-orbit/

Parker, J. S., & Anderson, R. L. (2014). Low-Energy Lunar Trajectory Design, JPL Deep Space Communications and Navigation Series, Vol. 12. Wiley– Chapter 2, p.27-117. https://doi.org/10.1002/9781118855065

Rashid, Tasneem, "Lagrangian Points and Jacobi Constants for a Class of Asteroids" (2015). Doctoral Dissertations and Master's Theses. 236. https://commons.erau.edu/edt/236

Rein, H., & Liu, S.-F. (2012). Rebound: an open source multi-purpose N-body code for collisional dynamics. Astronomy & Astrophysics, 537, A128. https://doi.org/10.1051/0004-6361/201118085

Souchay, J. J., & Dvorak, R. (Eds.). (2010). Dynamics of small solar system bodies and exoplanets (Lecture Notes in Physics, Vol. 790). Springer Nature. https://doi.org/10.1007/978-3-642-04458-8

Van Anderlecht, Alexandre G., "Tadpole orbits in the L4/L5 region: Construction and links to other families of periodic orbits" (2016). Open Access Theses. 823. https://docs.lib.purdue.edu/open_access_theses/823

Xiaodong Liu and Jürgen Schmidt, Dust arcs in the region of Jupiter's Trojan asteroids, Astronomy & Astrophysics, 609 (2018) A57, DOI: https://doi.org/10.1051/0004-6361/2 01730951

Xiaodong Liu and Jürgen Schmidt, Comparison of the orbital properties of Jupiter Trojan asteroids and Trojan dust, Astronomy & Astrophysics, 614 (2018) A97, DOI: https://doi.org/10.1051/0004-6361/2 01832806

Appendix A
Angle Criteria Function and Application Code

Function (from python blog –not written by Author):

```
def angle_between(p1, p2):
      ang1= np.arctan2(*p1[::-1])
      ang2= np.arctan2(*p2[::-1])
      return np.rad2deg((ang1-ang2)%2*np.pi)

xy_test = np.zeros((sim.N, N_out, 2)) #array for test of coordinates
B=[x[1][1],y[1][1]]# position of Jupiter at beginning
angles=[] #list of angles

for j in range(1, N_testparticles):
  angles.append(angle_between(xy_test[j][N_out-1], B)) #calculating angles and adding them to list for each par-
ticle

for i, t in enumerate(times):
  for j in range(N_testparticles-1):
                    if 58 <= angles[j] <= 63 #filter
                      x_trojan[j, i] = xy_test[j][i, 0]
                      y_trojan[j, i] = xy_test[j][i, 1]
```

Appendix B
Complete Code for Setting up the Simulation, Integrating, and Plotting (example of elliptical criteria)

```
import rebound
import matplotlib.pyplot as plt
import numpy as np
import math as m

#create a sim object

sim = rebound.Simulation()

#change integrator
sim.integrator=”whfast”
sim.dt=0.001

#add your planets
sim.add(‘Sun’)
sim.add(m=1e-3, a=1, e=0.1, hash=”Jupiter”, primary=sim.particles[0])

#add your test particles
N_testparticles = 100
```

```
for i in range (N_testparticles):
   sim.add(a=np.random.uniform(low=0.9,  high=1.1),  e=np.random.rayleigh(scale=0.03),  M=np.random.
rand()*2*np.pi, Omega=np.random.rand()*2*np.pi, omega=np.random.rand()*2*np.pi, primary=sim.particles[0])

#move to COM
sim.move_to_com()

#set only 2 active planets
sim.N_active = 2

#set up your number of orbits and arrays
orbit_number = int(input('What is the orbit number?'))
N_out =int(input('How many times should I keep store of the position of the test particles?'))
t_max= orbit_number *2* np.pi #max time
times = np.linspace ((99*t_max)/100, t_max, N_out)
x = np. zeros ((sim.N, N_out)) # coordinates for x
y = np. zeros_like (x)
x_rotated = np. zeros ((sim.N, N_out)) # coordinates for x
y_rotated = np. zeros_like (x)
a_final = []
e_final =[]
x_trojan = np.zeros((sim.N, N_out))
y_trojan = np.zeros_like (x_trojan)

# now integrate and rotate
for i, t in enumerate(times):
   sim.integrate (t, exact_finish_time =0)
   l = sim.particles[1].calculate_orbit(primary=sim.particles[0]).f
   for j in range (sim.N) :
      x[j,i] = sim.particles[j].x
      y[j,i] = sim.particles[j].y
      x_rotated[j, i]= (x[j, i] * m.cos(l)) + (y[j, i]*m.sin(l))
      y_rotated[j, i]= (-x[j, i] * m.sin(l)) + (y[j, i]*m.cos(l))

#calculate orbital elements

for j in range (1, N_testparticles+1):
   orbits= (sim.particles[j].calculate_orbit(primary=sim.particles[0]))
   a_final.append(orbits.a)
   e_final.append(orbits.e)

for i, t in enumerate(times):
   for j in range(N_testparticles):
      if 0<=e_final[j]<=0.1:
         if 0.97 <= a_final[j] <= 1.03: elliptical filter, changes given criteria
            x_trojan[j, i] = x_rotated[j,i]
            y_trojan[j, i] = y_rotated[j,i]
```

```
x_trojan = x_trojan[(~np.all(x_trojan == 0, axis=1))]
y_trojan = y_trojan[(~np.all(y_trojan == 0, axis=1))]

#plot one particle at all times
fig, ax = plt.subplots()
plt.grid (True)
plt.plot(x_trojan[0], y_trojan[0])
plt.scatter(x_rotated[1], y_rotated[1], s=10, color=”red”) #plot Jupiter’s position
ax.set_title (“at 1000 orb”)
ax. set_aspect (‘equal’)
ax. set_xlabel (“x in AU”)
ax. set_ylabel (“y in AU”)
rebound.OrbitPlot(sim, particles=[1], primary=0, show_primary=True, fig=fig, ax= ax, color=True)
 #Jupiter’s orbit
ax.set_xlim([-1.5, 1.5])
ax.set_ylim([-1.5,1.5])

#plot position of all particles at final time
fig1, ax1 = plt.subplots()
plt.grid (True)
rebound.OrbitPlot(sim, particles=[1], primary=0, show_primary=True, fig=fig1, ax= ax1, color=True)
ax1.set_xlim([-2, 2])
ax1.set_ylim([-2,2])
plt.scatter(x_trojan[:, N_out-1], y_trojan[:, N_out-1], s=10)
plt.scatter(x_rotated[1], y_rotated[1], s=50, color=”red”)
ax1.set_title (“at 1000 orb all particles”)
ax1. set_aspect (‘equal’)
ax1. set_xlabel (“x in AU”)
ax1. set_ylabel (“y in AU”)

#plot e v a graph
fig2 = plt.figure(figsize=(15,5))
ax2 = plt.subplot(111)
ax2.set_xlabel(“a”)
ax2.set_ylabel(“e”)
ax2.set_xlim([-20, 20])
ax2.set_ylim([0, 1])
ax2. set_title (“at 1000 orb e v a”)
plt.scatter(a_final,e_final,s=10)
```

Appendix C
Jacobi Criteria Function and Application Code

```
def get_jacobi_const(): #Jacobi Function from Poincare Maps Rebound documentation—not written by Author
   star = sim.particles[0]
   planet = sim.particles[1]
   particle = sim.particles[j]
```

```
    rstar = np.array(star.xyz)
    rplanet = np.array(planet.xyz)
    r = np.array(particle.xyz)
    v = np.array(particle.vxyz)

    KE = 0.5 * v@v # test particle kinetic energy
    mu1 = sim.G * star.m
    mu2 = sim.G * planet.m
    r1 = r-rstar
    r2 = r-rplanet
    PE = -1*mu1/np.sqrt(r1@r1) - mu2/np.sqrt(r2@r2) # test particle potential energy

    lz = np.cross(r,v)[-1]

    CJ = 2 * planet.n * lz - 2 * (KE + PE) # Jacobi constant
    return CJ

CJ=[] #list for jacobi constants

for j in range (sim.N): #calculating constants
    C = get_jacobi_const()
    CJ.append(C)

for i, t in enumerate(times):
    for j in range(N_testparticles-1):
        if 2.95<= CJ[j] <= 3:
                        x_trojan[j, i] = x[j, i]
                          y_trojan[j, i] = y[j, i]
```

Appendix D
Equation System to Solve for Arbitrary Trojan Motion

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 0.499 \quad (1)$$

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 = +- \frac{\sqrt{3}}{2} \quad (2)$$

$$\alpha_1\lambda_1 + \alpha_2\lambda_2 + \alpha_3\lambda_3 + \alpha_4\lambda_4 = 0 \quad (3)$$

$$\beta_1\lambda_1 + \beta_2\lambda_2 + \beta_3\lambda_3 + \beta_4\lambda_4 = 0 \quad (4)$$

$$\alpha_1 = \frac{\lambda_1^2 - Uxx}{2\lambda_1 + Uxy} \quad (5)$$

$$\alpha_2 = \frac{\lambda_2^2 - Uxx}{2\lambda_2 + Uxy} \quad (6)$$

$$\alpha_3 = \frac{\lambda_3^2 - Uxx}{2\lambda_3 + Uxy} \quad (7)$$

$$\alpha_4 = \frac{\lambda_4^2 - Uxx}{2\lambda_4 + Uxy} \quad (8)$$