El coste energético y tiempo de vuelo de una maniobra orbital

Número de palabras: 3876

Índice

1. Introducción

El espacio siempre ha generado una sensación especial en mí, y ha hecho que sea mi tema de mayor interés, y seguramente, mi futura profesión. El estudio del universo hace que me sienta increíblemente pequeño. Todas nuestras vidas, nuestras familias, nuestras guerras, nuestros problemas; todo lo que conocemos, o llegaremos a conocer, se reducirá a la nada con el paso de unos pocos siglos, un suspiro para el universo. Nada de lo que hagamos afectará a las grandes hipergigantes azules de la galaxia de Andrómeda, al colosal agujero negro en el centro de nuestra propia galaxia; pero eso es lo hace que me guste tanto la astrofísica, el ser capaz de dejar atrás mi condición de ser temporal, vulnerable y engreído, abrazar la cruda realidad de nuestra existencia, y poder estudiar las maravillas que esconde nuestro cosmos, impasibles ante el paso de nuestro tiempo, esperando a ser descubiertas y estudiadas por nosotros, el más insignificante de los entes en el universo.

Dentro de mis temas de interés se encuentra la astrodinámica, el estudio de la mecánica celeste. En concreto, uno de los subtemas que más me atraen son los puntos de Lagrange, unos lugares en el espacio próximo a dos cuerpos orbitando el uno al otro en los que uno se puede colocar, y permanecer estacionario respecto al resto de cuerpos. La primera vez que oí acerca de ellos, hace unos tres años, no tenía ni idea de cómo podía existir tal cosa. ¿Cómo puede quedarse un cuerpo estacionario sin caerse? ¿Por qué no se usan más a menudo? La realidad es que sí se usan. Al conjunto de posibles maniobras orbitales realizadas con la ayuda de los puntos de Lagrange se les conoce como Red de Transporte Interplanetaria (RTI), pero debido a la dificultad que supone el estudio de las ecuaciones diferenciales que explican su naturaleza, no ha habido tiempo suficiente como para hacerla accesible para las misiones espaciales actuales.

Decidido a estudiar la naturaleza de los puntos de Lagrange, busqué más tipos de desplazamientos por el espacio, con el fin de tener algo con lo que compararlos. Pronto encontré lo que buscaba: la transferencia de Hohmann. La transferencia de Hohmann es probablemente el método más utilizado de todos, debido principalmente a su eficiencia a la hora de alcanzar órbitas en tiempos reducido con gastos de combustible mínimos.

Para este estudio se utilizará software desarrollado por mí junto con software de terceros, que únicamente tendrán en cuenta para hacer sus predicciones la Teoría de la Gravitación Universal. El objetivo de este trabajo es poder comparar los gastos energéticos, el desplazamiento total, y el tiempo empleado por ambos métodos para llegar de un punto a otro, y remarcar sus beneficios y flaquezas.

Por lo tanto, mi pregunta de investigación será: **¿Cómo influye el tipo de maniobra orbital en el gasto**

energético, tiempo de vuelo, y desplazamiento?

2. Marco Teórico

2.1. La Transferencia de Hohmann

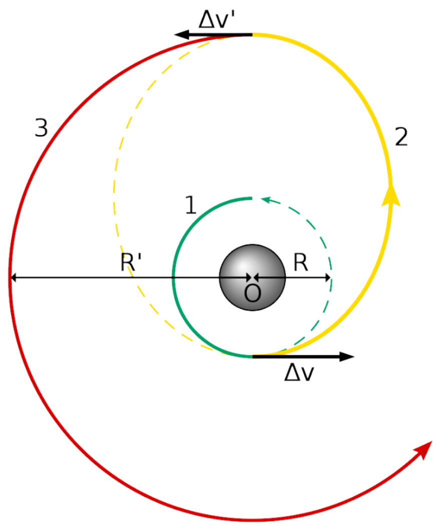


Figura 1: Transferencia de Hohmann[1]

La transferencia de Hohmann es el método más utilizado para moverse por el sistema solar. Como se ve en la figura de abajo, la transferencia de Hohmann se suele utilizar para desplazar un objeto que está orbitando alrededor de un único astro en una órbita circular, a otra órbita circular más alejada (o más cercana). Consiste en realizar una serie de propulsiones que produzcan cambios de velocidad instantáneos en la nave. El primer cambio de velocidad es necesario para mover al objeto a una órbita elíptica con la amplitud deseada, y se deberá realizar perpendicular al radio de la órbita en el sentido del movimiento del cuerpo (prógrado). El segundo se realiza en el apogeo de la órbita elíptica previamente adquirida (en su punto más alto), y también deberá ser prógrado.

2.2. El Problema de los 3 Cuerpos

El problema de los 3 cuerpos busca predecir el comportamiento, inducido únicamente por la gravedad, de dichos cuerpos. Para resolverlo, se empezará con la teoría de la Gravitación Universal:

[1] https://en.wikipedia.org/wiki/Hohmann_transfer_orbit

$$m\frac{d^2\vec{R}}{d\tau^2} = -Gm_1m\frac{\vec{R}_{13}}{R_{13}^3} - Gm_2m\frac{\vec{R}_{23}}{R_{23}^3} \tag{1}$$

Donde $\vec{R}$ es la distancia respecto al origen del tercer cuerpo, $\vec{R}_{13}$ es la distancia desde el primer cuerpo hasta el tercero, y $\vec{R}_{23}$ es la distancia desde el segundo cuerpo hasta el tercero.

Para poder simular la RTI de una forma clara y fácilmente analizable, se utilizará una modificación del problema de los 3 cuerpos: el problema de los 3 cuerpos circular reducido (CR3BP, según su traducción del inglés). Las diferencias entre ambos se hallan en el nombre. "Circular" significa que las órbitas de los dos cuerpos más grandes se consideran perfectamente circulares. "Reducido" significa que que la masa del tercer objeto (en este caso un satélite) se considera tan pequeña en comparación con los otros dos cuerpos, que no se tiene en cuenta a la hora de desarrollar las ecuaciones de movimiento.

Primero, se deberá simplificar el caso concreto del sistema Tierra-Luna a uno general.

Se definirá una masa característica m', que será igual a la suma de las masas de los dos cuerpos más grandes en el sistema (m_1 y m_2). Ahora se definirá el parámetro μ (la proporción entre las 2 masas) con la siguiente igualdad: $\mu = \frac{m_2}{m_1+m_2}$. Para este sistema, será igual a 0.01215.

En segundo lugar, se definirá una distancia característica l' que será igual a la distancia entre los dos cuerpos más grandes, $3{,}850 * 10^5$ en este caso. Se hará lo mismo con la velocidad, de forma que $v' = V * v$, donde V será igual a 1.025.

En tercer lugar se definirá un tiempo propio t' de una forma en la que la Constante Gravitacional sea igual a 1. Utilizando la tercera ley de Kepler, si se establece de antemano que el periodo orbital de los dos cuerpos más masivos es 2, entonces $t' = \frac{T}{2\pi}t$, donde T será $2{,}361 * 10^6$.

Así conseguimos simplificar la ecuación (1) a la expresión:

$$m\frac{d^2\vec{r}}{dt'^2} = -\frac{(1-\mu)}{r_{13}^3}\vec{r}_{13} - \frac{\mu}{r_{23}^3}\vec{r}_{23} \tag{2}$$

Si se aplican las transformaciones necesarias para convertir el sistema de referencia en uno rotatorio, se obtendrá un sistema de 3 ecuaciones diferenciales paramétricas que permitirán calcular la aceleración ejercida en el satélite en todo momento:

$$
\begin{cases}
\ddot{x} = 2\dot{y} + x - \frac{1-\mu}{r_{13}^3}(x+\mu) - \frac{\mu}{r_{23}^3}(x-1+\mu) \\
\ddot{y} = -2\dot{x} + y - \frac{1-\mu}{r_{13}^3}y - \frac{\mu}{r_{23}^3}y \\
\ddot{x} = -\frac{1-\mu}{r_{13}^3}z - \frac{\mu}{r_{23}^3}z
\end{cases}
\tag{3}
$$

2.3. La constante de Jacobi

El sistema de ecuaciones anterior posee una constante independiente del tiempo. Se trata de la constante de Jacobi:

$$
C_J = n^2(x^2+y^2) + 2\left(\frac{\mu_1}{r_1} + \frac{\mu_2}{r_2}\right) - (\dot{x}^2 + \dot{y}^2 + \dot{z}^2) \tag{4}
$$

Se considera $n = 1$ en el sistema generalizado.

Si se calculan las constantes de Jacobi para una serie de satélites con velocidad nula, repartidos por el espacio cercano a los 2 cuerpos más masivos, y luego se traza una serie de líneas equipotenciales a lo largo de los valores devueltos, se obtendrá una gráfica como la de la figura 2.

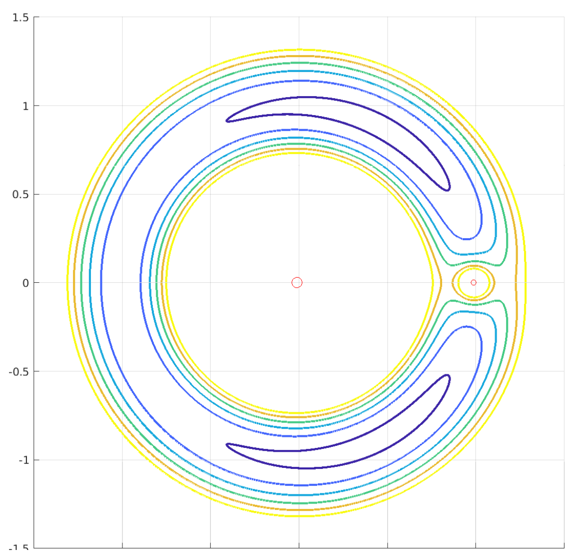


Figura 2: Representación con líneas equipotenciales de la constante de Jacobi[2]

Lo que representan las líneas equipotenciales es lo que se conoce en astrodinámica cómo superficies de velocidad relativa nula. Lo que eso significa es que un satélite cualquiera nunca podrá superar la línea

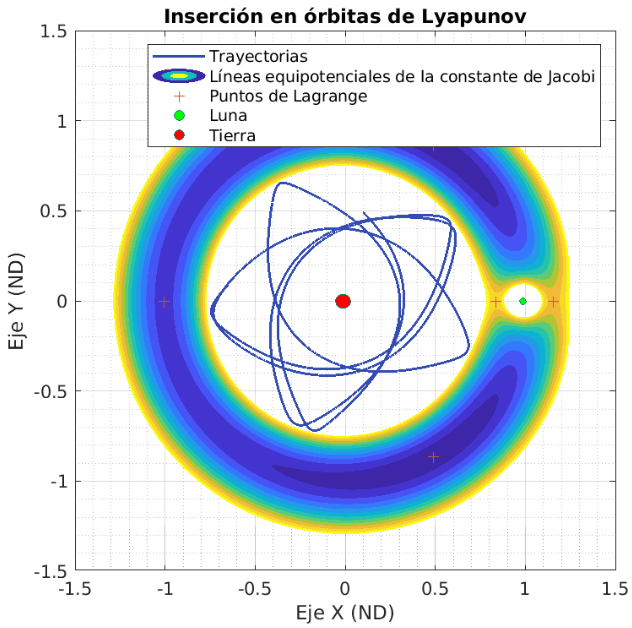
[2]Generado con Matlab. Códigó por Gereshes: https://github.com/gereshes/Matlab-Astrodynamics-Library

correspondiente al valor de su constante de Jacobi. Esto será muy importante a la hora de estudiar la RTI.

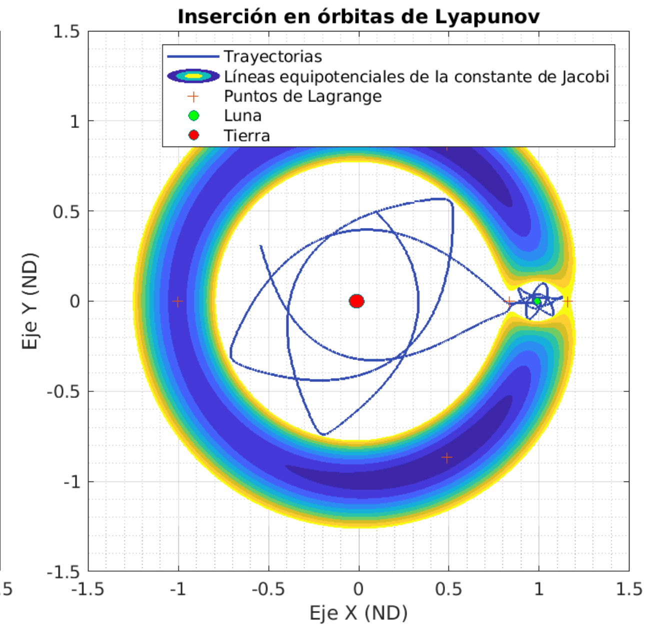
2.4. Alcance de un satélite dependiendo de su constante de Jacobi

Como se ha mencionado antes, la constante de Jacobi de un satélite se puede utilizar para calcular la distancia máxima a la que puede llegar dicho objeto, o en otras palabras, su superficie de velocidad relativa nula. A continuación se muestran unas gráficas de diferentes satélites, que han partido de la misma posición, pero con velocidades diferentes. Cada uno de ellos tiene una constante de Jacobi diferente, y por lo tanto, posibles alcances distintos.

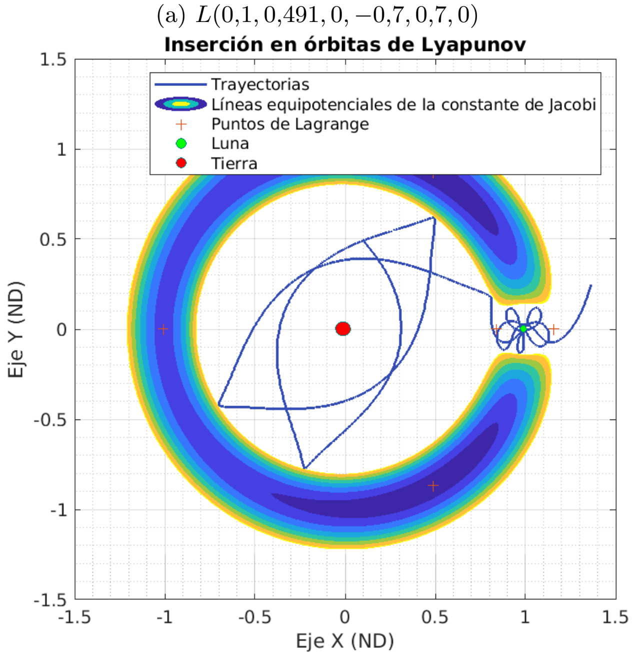
Se difinen las características del satélite en la forma: $L(x, y, z, x', y', z')$. Para los 3 satélites $L(0{,}1, 0{,}491, 0, -0{,}7, 0{,}7, 0)$, $L0(0{,}1, 0{,}491, 0, -0{,}71, 0{,}72, 0)$ y $L(0{,}1, 0{,}491, 0, -0{,}68, 0{,}78, 0)$, se obtienen las gráficas correspondientes (Figura 3).



(a) $L(0{,}1, 0{,}491, 0, -0{,}7, 0{,}7, 0)$



(b) $L(0{,}1, 0{,}491, 0, -0{,}71, 0{,}72, 0)$



(c) $L(0{,}1, 0{,}491, 0, -0{,}68, 0{,}78, 0)$

Figura 3: Órbitas en un sistema de referencia rotatorio, junto con representación de diferentes valores de la constante de Jacobi[3]

La franja amarilla clara en cada una de las gráficas representa el valor inicial de la constante de Jacobi de los satélites, o en otras palabras, las superficies de velocidad relativa nula. En la primera gráfica, el satélite está confinado dentro del sistema. Sin embargo, en la segunda gráfica se ve como el satélite gana acceso al segundo

[3]Generado con Matlab. Códigó por Gereshes: https://github.com/gereshes/Matlab-Astrodynamics-Library

cuerpo gracias a una diferencia de velocidad inicial mínima. Por último, el tercer satélite es capaz de escapar por completo del sistema, de nuevo, con una diferencia de velocidad inicial muy pequeña. En las tres gráficas se aprecia que las puertas que surgen entre el primer cuerpo, el segundo, y el espacio exterior, son en realidad los puntos de Lagrange L1y L2. Con esta información se puede deducir que los puntos de Lagrange son una parte clave del funcionamiento de la RTI, y funcionan como ventanas a una "superautopista interplanetaria".

2.5. Órbitas de Halo

La mayoría de órbitas convencionales giran alrededor de un cuerpo, o de un baricentro, de un sistema. Sin embargo, al estudiar el CR3BP, los físicos se encontraron con un nuevo tipo de órbitas que incumplían esta norma general. Las llamadas órbitas de halo giran alrededor de los diferentes puntos de Lagrange de un sistema. Existen muchos tipos, y solo las llamadas órbitas de Lyapunov son de interés para este estudio. Estas están contenidas en el mismo plano sobre el que rotan los 2 cuerpos más masivos del sistema; o en otras palabras: planas. Tienden a adquirir forma de "judía".

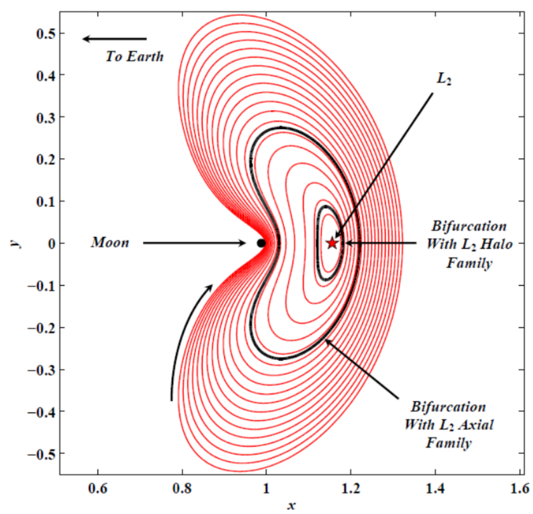


Figura 4: Órbitas de Lyapunov del sistema Tierra-Luna[4]

2.6. Variedades estables e inestables, y su aplicación a la RTI

Es posible crear variedades de 3 dimensiones (un conjunto de planos que forman una superficie) a partir de las trayectorias que seguiría un objeto que alcanza una órbita de halo concreta de forma asintótica ("estable",

[4]Imagen tomada del trabajo de Daniel J. Grebow: "GENERATING PERIODIC ORBITS IN THE CIRCULAR RESTRICTED THREEBODY PROBLEM WITH APPLICATIONS TO LUNAR SOUTH POLE COVERAGE"

en azul), o a partir de las que sigue otro objeto que la escapa ("inestable", rojo). Cualquier objeto que se encuentre en cualquier punto de estas variedades acabará inevitablemente alcanzando (o escapando) una órbita de halo. Las órbitas de halo, producto de las propiedades de los puntos de Lagrange, son las puertas entre las distintas variedades. De esta forma se empieza a ver más claramente el concepto de superautopista interplanetaria.

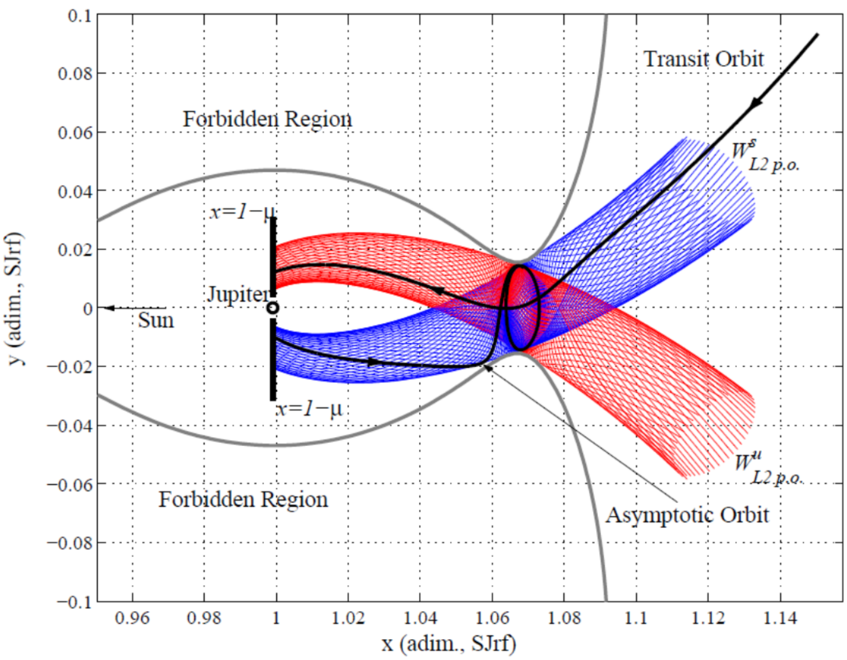


Figura 5: Variedades estable (azul) e inestable (rojo) para una determinada órbita de halo (negro) en el sistema Tierra-Luna[5]

3. Metodología

3.1. Cálculo de la transferencia de Hohmann

Para calcular los cambios de velocidad necesarios para realizar una transferencia de Hohmann, se despeja la velocidad en el teorema de la conservación de la energía:

$$v = \sqrt{GM(\frac{2}{r} - \frac{1}{a})} \tag{5}$$

[5]Imagen sacada del trabajo de Franco Bernelli Zazzera, Francesco Topputo, Mauro Massari: "Assessment of Mission Design Including Utilization of Libration Points and Weak Stability Boundaries"

[6]https://en.wikipedia.org/wiki/Hohmann_transfer_orbit

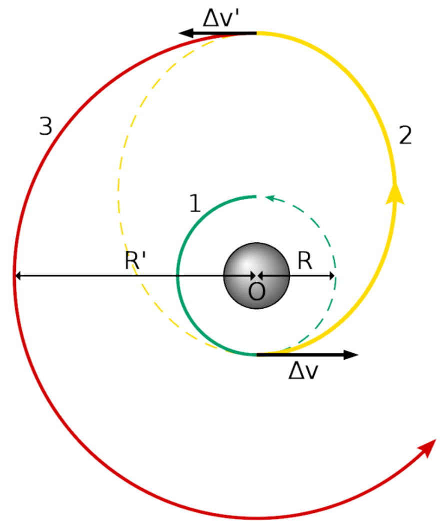


Figura 6: Transferencia de Hohmann[6]

Ahora, para hallar el cambio de velocidad en la primera transferencia desde la órbita inicial a la órbita elíptica de transición, se le resta la expresión correspondiente a la órbita circular inicial a la expresión correspondiente a la órbita elíptica:

$$\Delta v_1 = \sqrt{\frac{GM}{R}} \left(\sqrt{\frac{2R'}{R+R'}} - 1 \right) \tag{6}$$

Donde R es el radio de la órbita circular inicial, y R' el radio de la órbita circular final.

Si se lleva a cabo el mismo proceso para la segunda transferencia desde la órbita elíptica hasta la órbita circular final, se obtiene la siguiente expresión:

$$\Delta v_2 = \sqrt{\frac{GM}{R'}} \left(1 - \sqrt{\frac{2R}{R+R'}} \right) \tag{7}$$

Para calcular el tiempo requerido para llevar a cabo la transferencia se utilizará la tercera ley de Kepler. Se debe calcular la mitad del periodo de la órbita elíptica, como se ve en el diagrama debajo. Para estos cálculos se considererá que la órbita inicial tiene un radio de 1000 Km, más el radio de la Tierra (6.371 km).

Si se depejan y sustituyen los términos necesarios en la tercera ley de Kepler se obtiene:

$$T = \pi\sqrt{\frac{(R+R')^3}{GM}} \tag{8}$$

3.2. Desarrollo de software aplicado a la astrodinámica

Con el fin de ayudarme con el estudio del problema de los 3 cuerpos, desarrollé durante el transcurso de mi trabajo una serie de programas informáticos aplicados a los fenómenos y conceptos de astrodinámica que debía estudiar. Esto me permitió adquirir una mayor comprensión de su funcionamiento, a parte de conseguir unas herramientas para su estudio sobre las que tenía un control completo.

El primer proyecto que inicié fue un simulador gravitatorio. El programa calcula la atracción gravitatoria que siente un grupo de cuerpos por el resto, para luego sumarlas entre ellas y hallar la aceleración resultante. Esta aceleración se suma a su vez a la velocidad del cuerpo en cuestión, y ésta se suma a su vector posición. De esta forma, podemos calcular el comportamiento de todos los objetos que deseemos añadir.

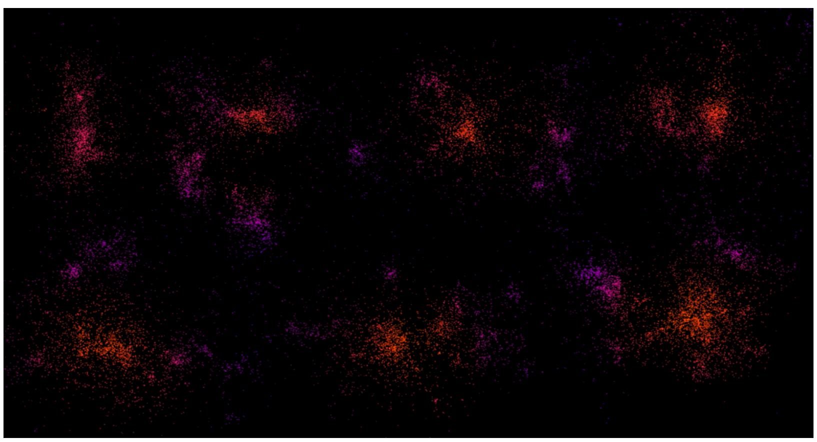


Figura 7: Simulación gravitatoria de 3000 cuerpos

El segundo programa que desarrollé fue un simulador de potencial efectivo. Esto último es la suma de la fuerza gravitatoria y la fuerza centrífuga aplicadas sobre un objeto. Para representarlo, deduje el potencial efectivo correspondiente a cada uno de los puntos en el plano estudiado, considerándolos objetos de masa despreciable, girando alrededor del centro de masas del sistema con una velocidad angular igual a la del segundo cuerpo. La fórmula utilizada es la siguiente:

$$U_{ef}(r) = \frac{L^2}{2mr^2} - \frac{GmM}{r} \tag{9}$$

Los resultados obtenidos se muestran en la figura 8.

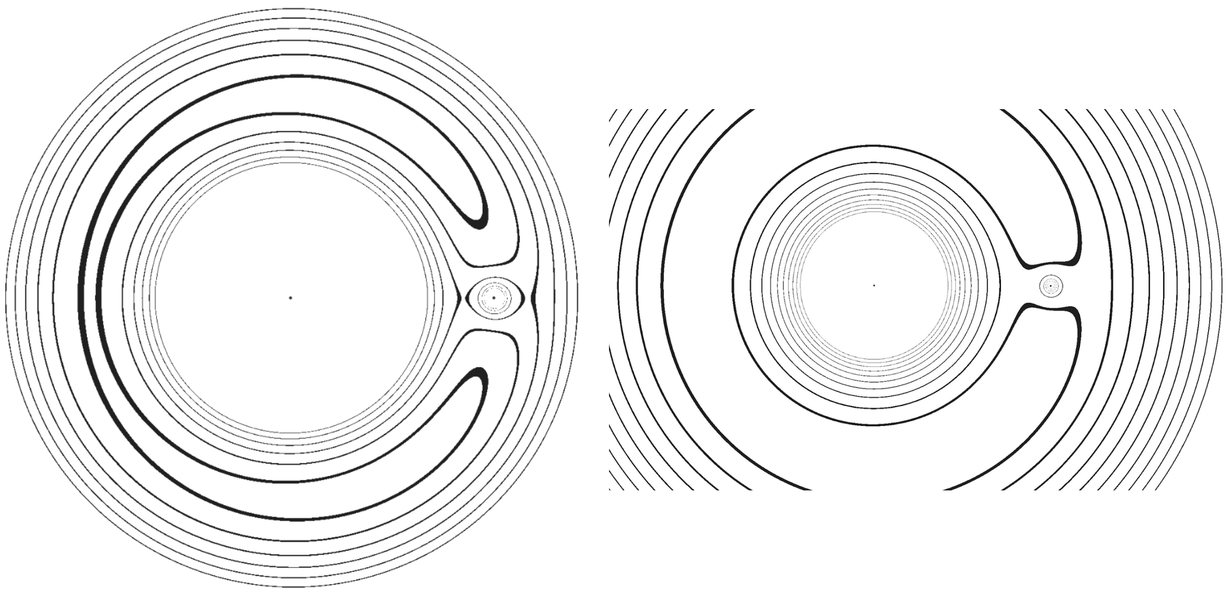


Figura 8: Representaciones del potencial efectivo del sistema Tierra-Luna

En tercer lugar, incluí en los programas anteriores el cómputo de los puntos de Lagrange. Como había predicho con mi programa anterior, estos surgieron en los máximos locales de la función del potencial efectivo.

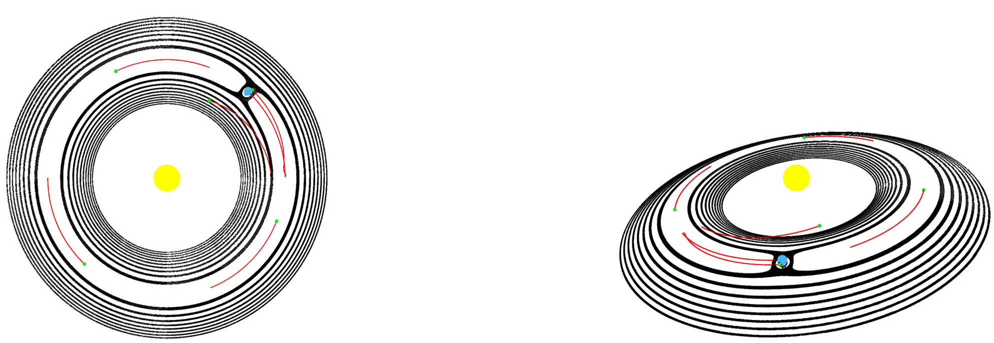


Figura 9: Representaciones del potencial efectivo y de los puntos de Lagrange del sistema Tierra-Luna

Por último, desarrollé un simulador del CR3BP, en el que se representa la órbita recorrida por un satélite de masa nula en un sistema rotatorio no inercial. Este programa en concreto entrará en juego más adelante. Su funcionamiento se basa en aplicar las ecuaciones del CR3BP, calculando la aceleración del satélite, para luego sumarla a su velocidad, que finalmente se suma a su posición.

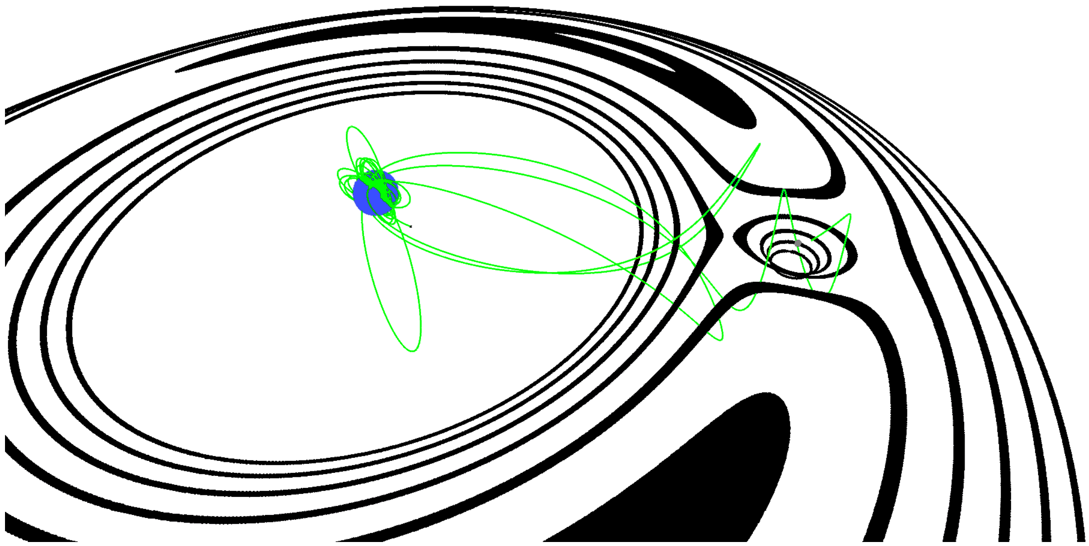


Figura 10: Representación de la órbita descrita por un satélite realizando una transferencia a la Tierra, sobre un mapa de líneas equipotenciales de los valores de la constante de Jacobi, en un sistema rotatorio no-inercial

3.3. Cálculo de las órbitas de halo

Una vez dispuse del software apropiado, para poder hallar órbitas de halo (órbitas de Lyapunov en nuestro caso), tuve en cuenta 2 propiedades extrapolables a todo cuerpo siguiendo dichas órbitas. Consideré que los dos cuerpos de mayor tamaño en el sistema estaban estacionarios sobre el eje X. Las órbitas de Lyapunov, al ser planas, no tienen en ningún momento componente Z. Eso significa que:

- Su velocidad en los ejes X y Z cuando corta el plano XZ es 0.
- Existen 2 puntos en su órbita donde se cumple la condición anterior.

Con estas normas en mente, somos capaces de desarrollar un método para hallar la órbita de Lyapunov en un punto cualquiera:

1. Se elige un punto contenido en el eje X positivo.
2. Se generan una serie de objetos con masa nula que partan desde ese punto, con velocidades en los ejes X y Z nulas. A estos objetos se les dará una velocidad inicial en el eje Y que se encuentre en un rango pertinente previamente seleccionado.

3. Se desarrollan las ecuaciones de movimiento del CR3BP.

4. Si el objeto vuelve a cruzar el plano XZ con velocidades en los ejes X y Z nulas, hemos hallado una órbita de Lyapunov sin necesidad de realizar el cómputo de la órbita entera, ahorrándonos tiempo de procesamiento.

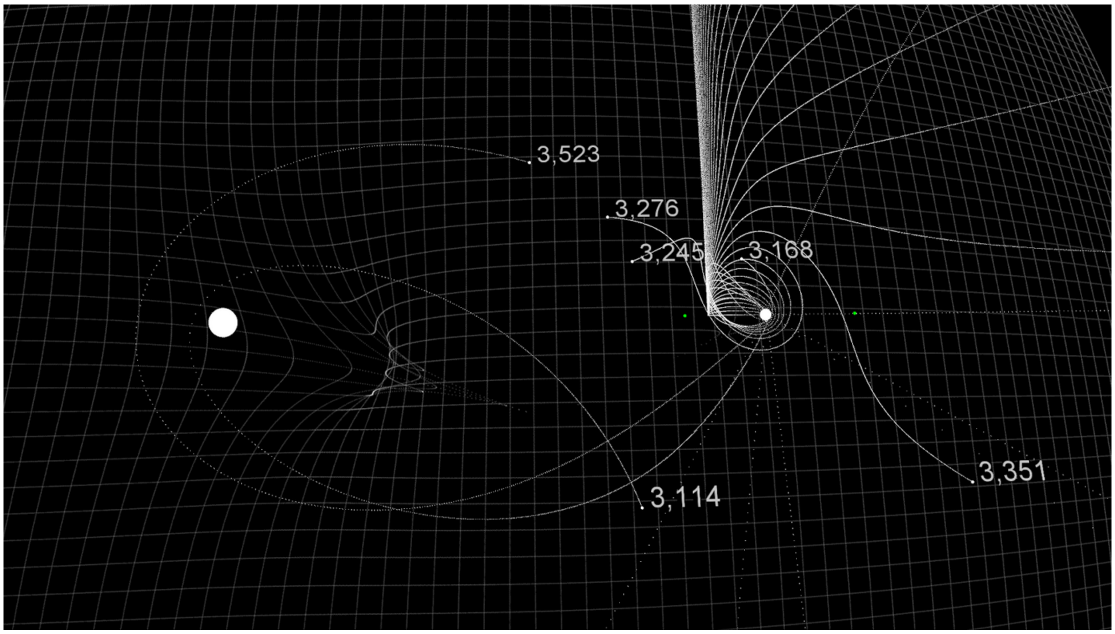


Figura 11: Intento de estimación de órbita de Lyapunov. Los números representan la constante de Jacobi de cada satélite, cuya órbita está dibujada en blanco

3.4. Transferencias desde una órbita geoestacionaria hasta una órbita de halo

Una vez hemos hallado las órbitas de Lyapunov, deberemos calcular la transferencia hasta ellas. El método que utilicé consiste en lanzar desde la Tierra un satélite con impulso variable, para luego realizar una propulsión puntual en el momento adecuado con el objetivo de insertar el objeto en la variedad estable correspondiente a la órbita de halo deseada. Debido a la naturaleza de las variedades estables, el objeto acabará inevitablemente en una órbita de halo.

Al tratarse de sistemas dinámicos caóticos, recurrí a un modelo de algoritmo desarrollado por Dawn Perry Gordon. Dicho algoritmo es en esencia muy parecido al que desarrollé yo para hallar órbitas de Lyapunov, adecuado específicamente a la búsqueda de inserciones en órbitas de halo. Este seleccionará de forma automática las trayectorias que cumplan nuestros requisitos (ángulo, altura de la órbita original), y devolverá a su vez el impulso necesario para llevar a cabo la maniobra, además del tiempo requerido.

En este caso, resulta más sencillo hallar las transferencias al revés: comenzando en la órbita de halo, y terminando en la Tierra, utilizando las inversas de las fórmulas del CR3BP. Se debe recordar que nuestra simulación actual calcula los resultados para un modelo general, por lo que se deberá revertir la transformación realizada en el apartado "El Problema de los 3 Cuerpos" si se pretende obtener los datos específicos para un sistema concreto, como será más adelante mi caso.

El algoritmo funciona de la siguiente manera (es importante recordar que el tiempo está invertido):

1. Se toma una amplitud para la órbita de halo y el ángulo con el que se aplicará la velocidad respecto al eje X (ver dibujo). Se decide un impulso inicial (Δv) de forma aleatoria, o basándose en los resultados de una iteración anterior.
2. Se aplican las ecuaciones del CR3BP de forma invertida, o "atrás en el tiempo".
3. Se mide la distancia desde el satélite hasta la Tierra. Si se encuentra en el rango de 1000 - 2200 Km (más el radio de la Tierra, 6.371 km) se finaliza la simulación y se reitera el proceso, con las mismas condiciones iniciales, pero con una sensibilidad mayor. De esta forma se obtendrá un resultado cada vez más exacto para el impulso requerido.
4. Si el satélite no tiene las condiciones descritas en el paso anterior, se pasa a la siguiente posible velocidad.

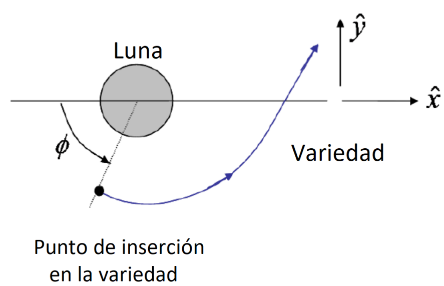


Figura 12: Esquema del modelo de funcionamiento del algoritmo usado para hallar las órbitas de halo[7]

4. Resultados

4.1. Resultados de los cálculos de las transferencias de Hohmann

[7]Imagen obtenida del trabajo de Dawn Perry Gordon: "TRANSFERS TO EARTH-MOON L2 HALO ORBITS USING LUNAR PROXIMITY AND INVARIANT MANIFOLDS"

[8]https://en.wikipedia.org/wiki/Hohmann_transfer_orbit

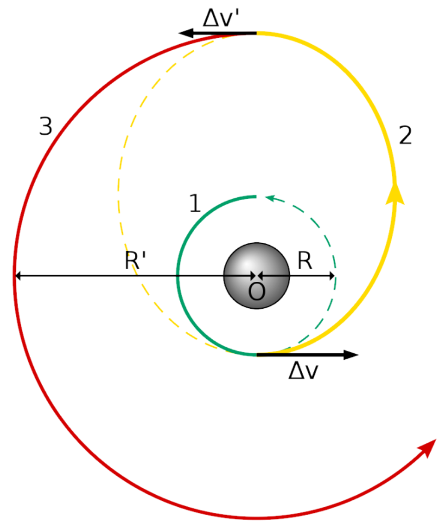
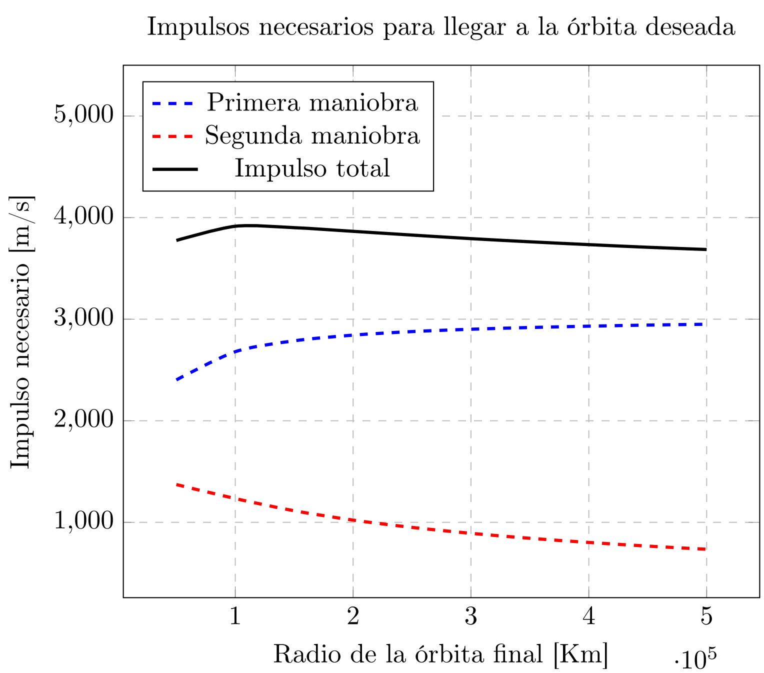
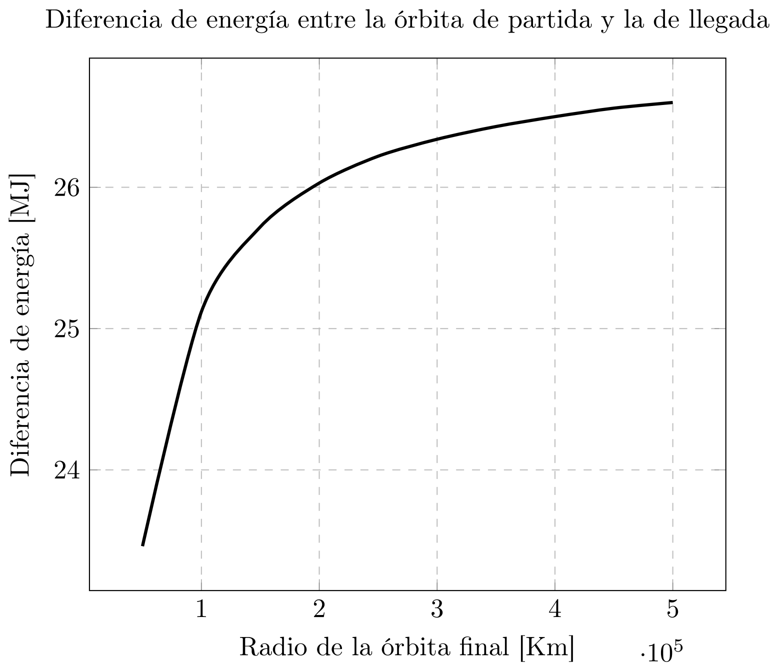


Figura 13: Transferencia de Hohmann[8]

Para calcular las transferencias de Hohmann consideré que la órbita de partida tenía un radio de 1100 Km, además del radio de la Tierra, 6378 Km. Luego calculé Δv_1 y Δv_2, la suma de ambas, y el tiempo requerido para realizar la transferencia.

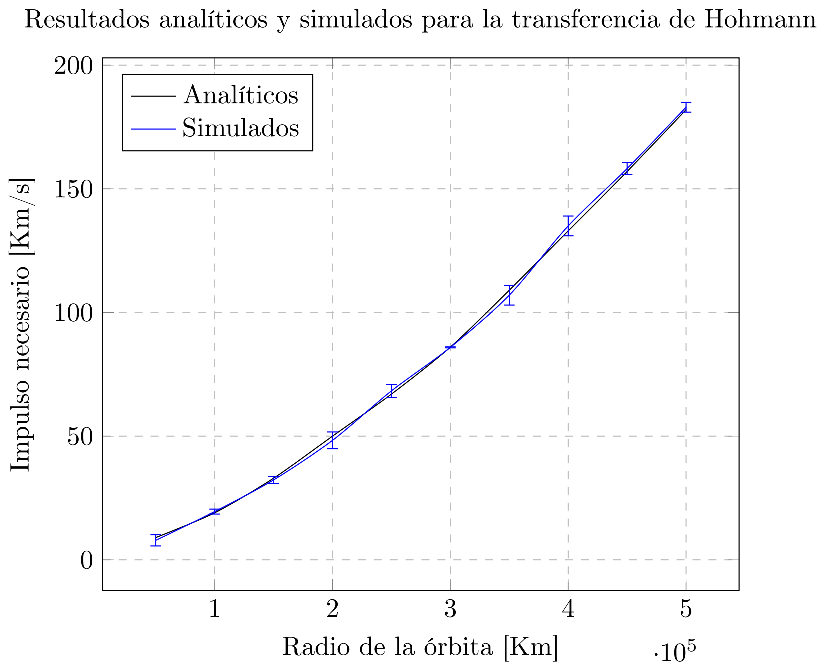




Observamos un claro incremento de la duración de vuelo respecto al radio de la órbita final, y al mismo tiempo una subida en la eficiencia (menos impulso necesario).

Al cotejar los resultados analíticos con los del simulador utilizado[9] las variaciones eran mínimas, debidas probablemente a diferencias en el redondeo. [H]

[9],
[9] https://es.mathworks.com/matlabcentral/fileexchange/38942-the-hohmann-orbit-transfer



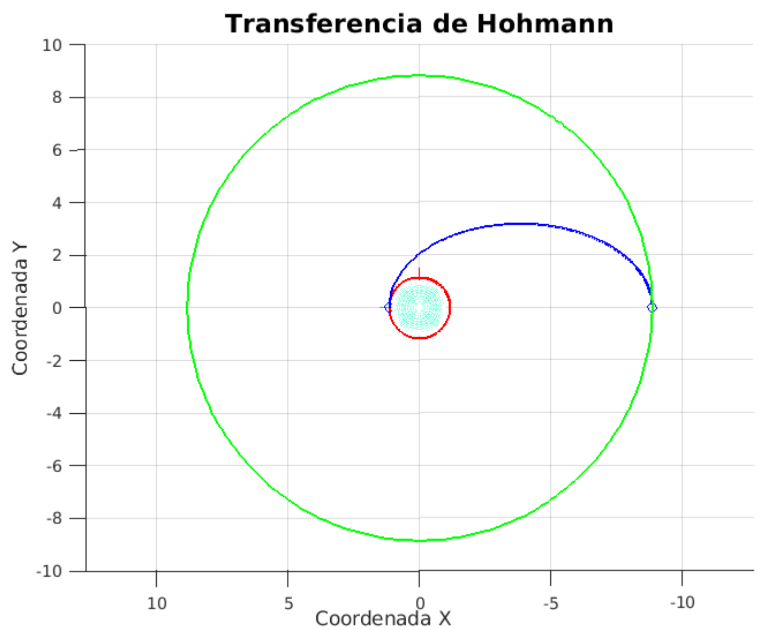


Figura 14: Muestra del resultado del simulador para la transición entre una órbita circular a 1000 Km de la superficie de la Tierra a otra a 50 000 Km[10]

[10]https://es.mathworks.com/matlabcentral/fileexchange/38942-the-hohmann-orbit-transfer

4.2. Resultados de los cálculos de las órbitas de Lyapunov

A la hora de desarrollar el algoritmo que llevase a cabo la búsqueda de órbitas de Lyapunov, encontré varios problemas que me impidieron poder concluir el experimento satisfactoriamente. Al proceder con el método previamente explicado en el apartado de "Metodología", los resultados de las órbitas que obtuve no fueron los esperadas.

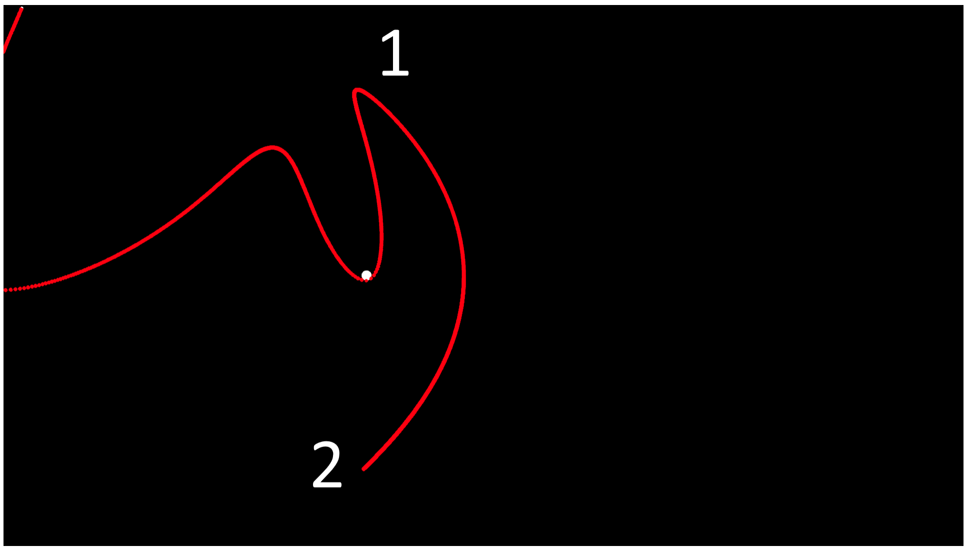


Figura 15: Intento de estimación de órbita de Lyapunov (en rojo). El satélite parte del punto 1, y sigue una trayectoria aparentemente periódica, pero en vez de orbitar, disminuye su velocidad hasta llegar a 0 (en el punto 2), vuelve sobre sus pasos, y finalmente cae hacia la Luna

En vez de seguir órbitas regulares, los objetos generados seguían uno de dos comportamientos: o caían en el pozo gravitatorio más cercano (como en la figura 15), o escapaban hacia el espacio exterior. Lo mismo ocurrió cuando intenté simular otros tipos de órbitas de halo. Para este último caso, desarrollé una inteligencia artificial que detectaba automáticamente el paso de las partículas, y evaluaba si estas se trataban o no de órbitas de halo. Con cada iteración, la IA aumentaba su sensibilidad, hasta que solo quedasen aproximaciones cercanas a las órbitas de halo canónicas.

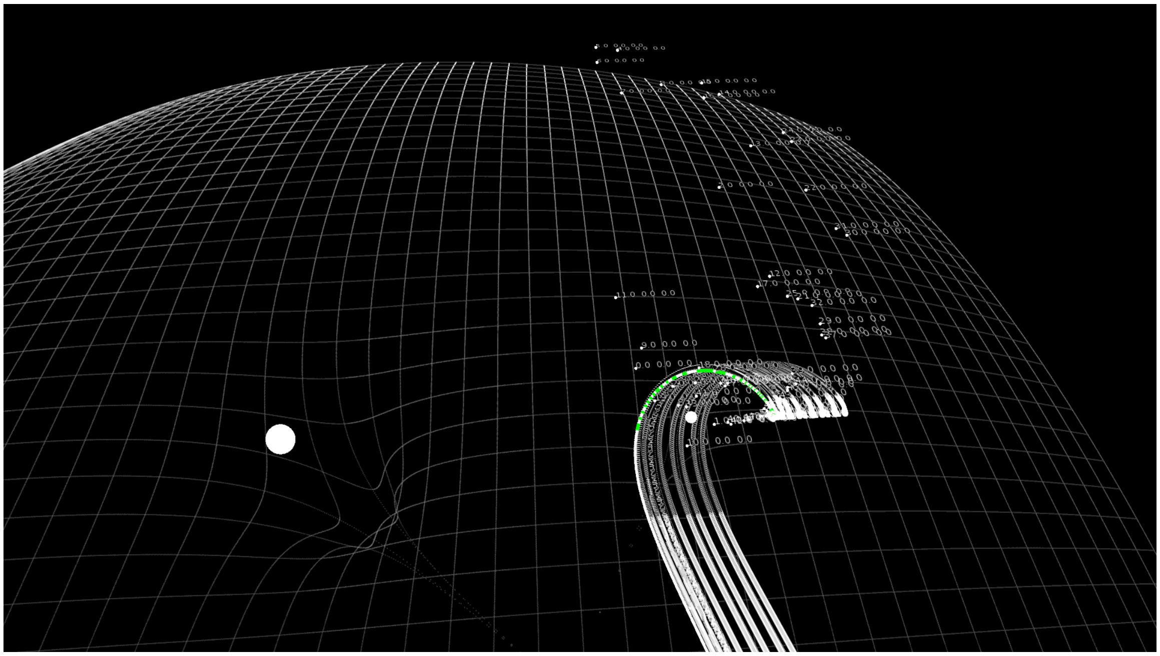


Figura 16: Intento de estimación de órbita de Lyapunov. Los números representan la constante de Jacobi de cada satélite, cuya órbita está dibujada en blanco

Sin embargo, todos mis intentos resultaron en órbitas no periódicas. Algunas de las más refinadas por la inteligencia artificial antes descrita guardaban similitudes con las órbitas de halo esperables, pero estos parecidos eran solo momentáneos (ver figura 16).

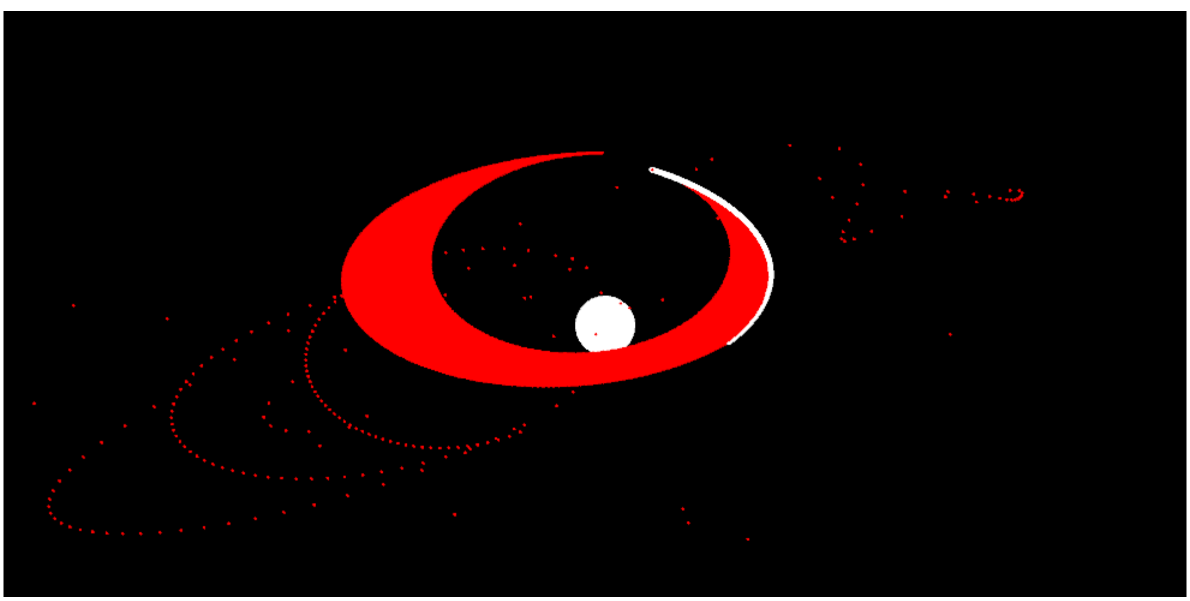


Figura 17: Intento de estimación de órbitas de halo. La superficie roja representa la variedad formada por el conjunto de órbitas simuladas. Las trayectorias parecen ser periódicas, pero terminan evolucionando de forma irregular, y cayendo hacia el cuerpo más cercano

4.3. Resultados de los cálculos de las inserciones en órbitas de Lyapunov

Debido a la complejidad del proceso, y de las dificultades que tuve al desarrollar mi propio algoritmo, me apoyé en los hallazgos de D. P. Gordon para seguir con el estudio. Utilizando sus estimaciones para las inserciones, simulé yo mismo algunas de las posibles órbitas, y recopilé sus datos.

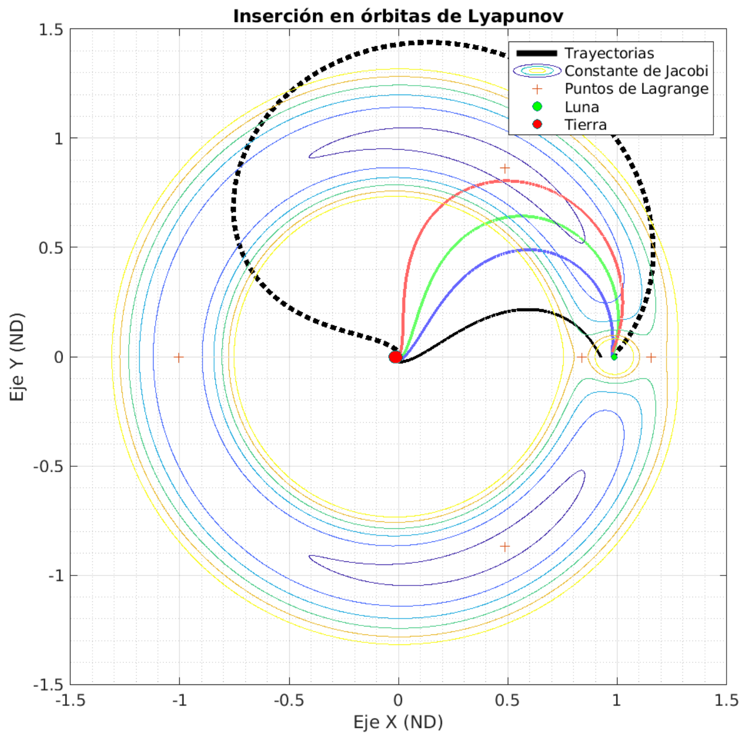
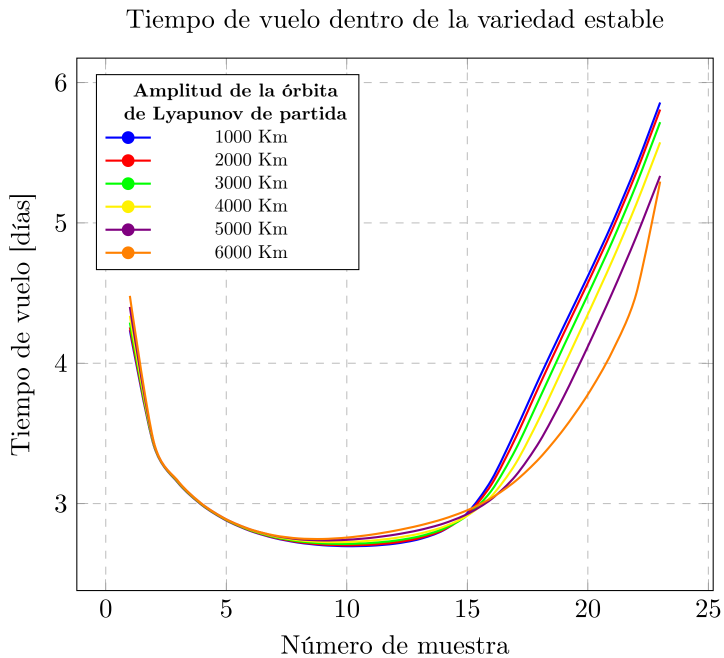
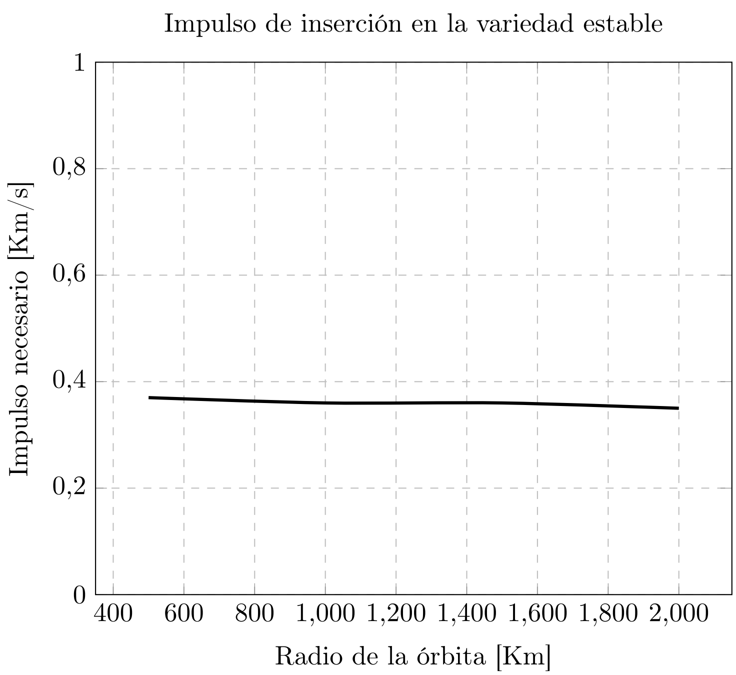


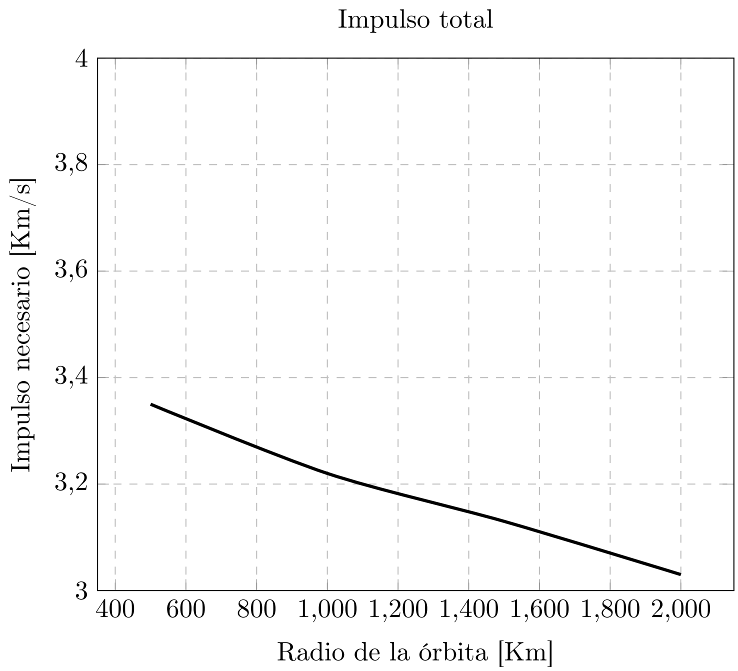
Figura 18: Representación de la inserción de 5 satélites en órbitas terrestres bajas en un sistema rotatorio no-inercial[11]

A la hora de calcular el impulso (δv), se debe tener en cuenta no solo el coste de inserción en la variedad inestable, sino también el coste de inserción en la órbita terrestre final (de nuevo, el tiempo fluye en sentido contrario). Este último es un factor a destacar, ya que la diferencia de velocidad necesaria para escapar de la órbita terrestre sube de forma importante cuanto más cercana esté esta a la Tierra, a diferencia del impulso de inserción en la variedad estable.

[11]Generado con Matlab. Datos sacados del trabajo de Dawn Perry Gordon: "TRANSFERS TO EARTH-MOON L2 HALO ORBITS USING LUNAR PROXIMITY AND INVARIANT MANIFOLDS". Código por Gereshes: https://github.com/gereshes/Matlab-Astrodynamics-Library


Tiempo de vuelo dentro de la variedad estable
Amplitud de la órbita de Lyapunov de partida
1000 Km
2000 Km
3000 Km
4000 Km
5000 Km
6000 Km
Tiempo de vuelo [días]
Número de muestra
0
5
10
15
20
25
3
4
5
6

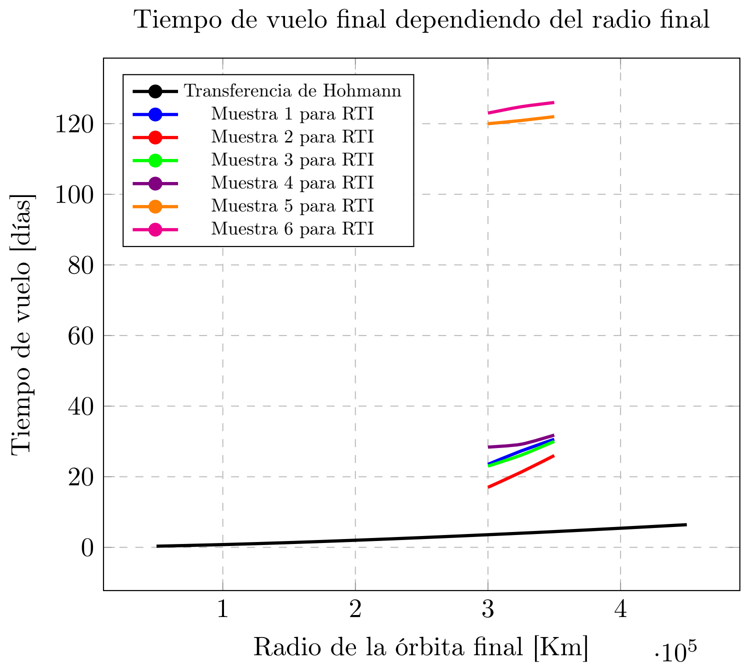

Impulso de inserción en la variedad estable
Impulso necesario [Km/s]
Radio de la órbita [Km]
0
0,2
0,4
0,6
0,8
1
400
600
800
1,000
1,200
1,400
1,600
1,800
2,000

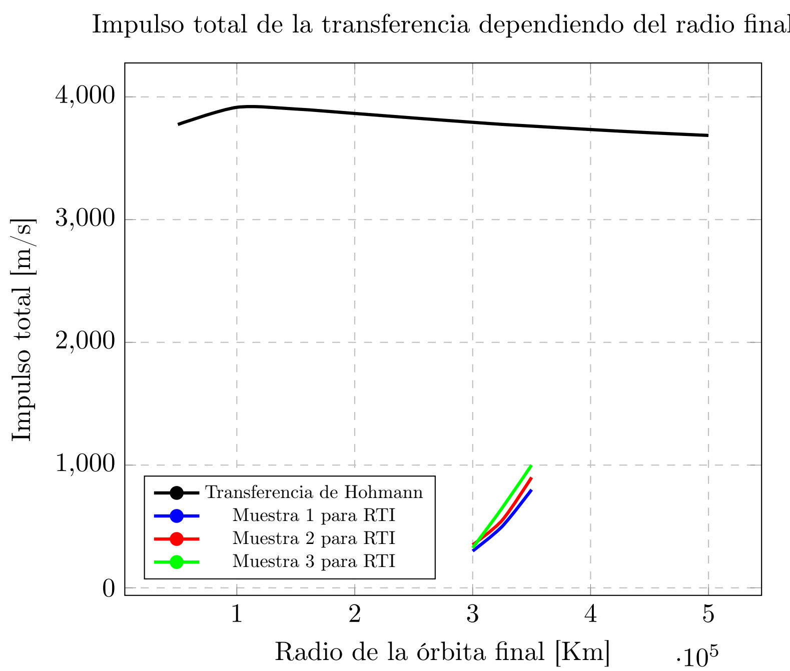


El cálculo del impulso total involucra también el cálculo de las órbitas de Lyapunov de partida, tarea larga y tediosa que, debido a las complicaciones con mi algoritmo, fui incapaz de llevar a cabo a pesar de mis esfuerzos. Es por esto que usaré los datos de D. P. Gordon y otros autores. Las barras para los resultados de la RTI (agrupados en muestras dependiendo del tipo de transferencia y/o procedencia de los datos[12][13]) se muestran únicamente en los rangos en los que es posible realizar una inserción en una órbita suficientemente estable.

[12] https://descanso.jpl.nasa.gov/monograph/series12/LunarTraj–04Chapter3TransferstoLunarLibrationOrbits.pdf

[13] Trabajo de Franco Bernelli Zazzera, Francesco Topputo, Mauro Massari: "Assessment of Mission Design Including Utilization of Libration Points and Weak Stability Boundaries"


Tiempo de vuelo final dependiendo del radio final
Transferencia de Hohmann
Muestra 1 para RTI
Muestra 2 para RTI
Muestra 3 para RTI
Muestra 4 para RTI
Muestra 5 para RTI
Muestra 6 para RTI
Tiempo de vuelo [días]
120
100
80
60
40
20
0
1
2
3
4
Radio de la órbita final [Km]
$\cdot 10^5$


Impulso total de la transferencia dependiendo del radio final
Impulso total [m/s]
4,000
3,000
2,000
1,000
0
Transferencia de Hohmann
Muestra 1 para RTI
Muestra 2 para RTI
Muestra 3 para RTI
1
2
3
4
5
Radio de la órbita final [Km]
$\cdot 10^5$

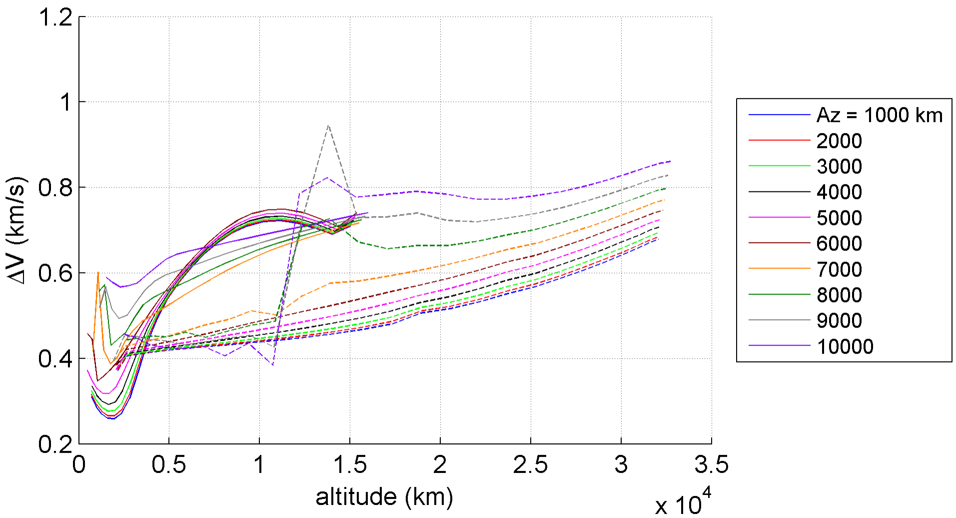


Figura 19: Impulso total de la maniobra usando la RTI, dependiendo de la altitud de la órbita final respecto a la Luna, por D. P. Gordon[14]

5. Conclusiones y comparación entre las transferencias de Hohmann y la RTI

Después de analizar nuestros resultados, podemos resumir la respuesta a nuestra pregunta en la siguiente tabla:

Para el mismo desplazamiento	Tiempo de Vuelo	Gasto Energético
Transferencia de Hohmann	Menor	Mayor
RTI	Mayor	Menor

En comparación con el resto de métodos de desplazamiento por el sistema solar, la transferencia de Hohmann ha demostrado ser una de las mejores en términos de eficiencia de combustible, tiempo de vuelo y dificultad. Sin embargo, el hecho de que se necesite hacer más de una maniobra (la transferencia desde la órbita inicial a la elíptica, y desde la elíptica a la final), no solo dificulta el proceso, sino que además añade gastos a la misión, puesto que la nave se ve en la obligación de transportar el combustible necesario para la segunda transferencia hasta poder usarlo. Esto último impide la reutilización de las partes del cohete correspondientes, a parte de ocupar el espacio que, de otra manera, habría ocupado una carga más valiosa.

[14] Imagen obtenida del trabajo de Dawn Perry Gordon: "TRANSFERS TO EARTH-MOON L2 HALO ORBITS USING LUNAR PROXIMITY AND INVARIANT MANIFOLDS"

A pesar de ser increíblemente eficientes en comparación con el resto de maniobras, y de acortar los tiempos de vuelo al mínimo, las transferencias de Hohmann suponen un gasto energético que perjudica gravemente al desarrollo de nuevas misiones espaciales. La RTI, por el contrario, ofrece la posibilidad de realizar transferencias energéticamente baratas, sacrificando a cambio rapidez. Aunque los tiempos de vuelo sean mayores, si se utilizan de forma bien planeada, superarán con creces la eficacia del resto de transferencias, por lo que posee una capacidad para enviar cargas de masa elevada al espacio mucho mayor que la de las transferencias de Hohmann.

Las aplicaciones en la vida real en el futuro de la RTI serán definitivamente mayores a las de las transferencias de Hohmann. En escenarios como el transporte de naves nodriza humanas, posiblemente con el fin de colonizar otros planetas, donde el tiempo no es un factor tan crítico como lo es la eficiencia energética, la RTI ofrecería un medio para desplazar las ingentes cantidades de masa en movimiento con un gasto energético mínimo, comparado con otro tipo de maniobras. Cabe destacar, además, la facilidad con la que un cuerpo se puede desplazar de variedad en variedad, de sistema en sistema, a lo largo del sistema solar, con la ayuda de la RTI.

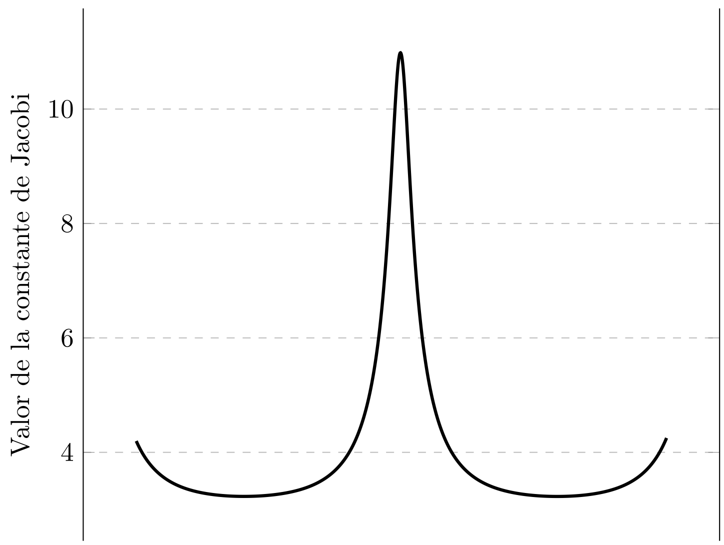
En segundo lugar, las órbitas de halo parecen ser posiciones idóneas para estaciones espaciales, satélites u observatorios que desean estar en reposo respecto al sistema en el que se encuentran. Por ejemplo, una base espacial posicionada en una órbita alrededor del punto L2estaría en el lugar perfecto para dar salida a misiones interplanetarias, o para recibirlas. También sería el lugar perfecto para construir naves de un tamaño demasiado grande para hacerlo en la Tierra, brindándonos la facilidad de construir sin preocuparse de la gravedad, además de tener a la nave puesta ya en órbita, eliminando la necesidad de incluir los aparatosos y costosos propulsores necesarios para abandonar la Tierra.

En conclusión, puede que las transferencias de Hohmann sean una mejor opción para las misiones espaciales a día de hoy, dada su eficiencia y corta duración. Sin embargo, es innegable que el potencial que ofrece la RTI para futuras misiones, mucho más ambiciosas que las de ahora, hará que a la larga sea el método de transporte por excelencia.

6. Posibles mejoras

Mis hipótesis respecto al por qué de las anomalías en los resultados de mi algoritmo se centran principalmente en el software usado para el desarrollo del algoritmo. Mis dificultades no son un caso aislado, y tras hablarlo con un experto[15] en el tema, puedo concluir que los errores de predicción de mi programa se deben en parte a su imprecisión a la hora de calcular parámetros tan sensibles como la constante de Jacobi, que a lo largo de la línea temporal no se mantenía constante, siguiendo un patrón que se correspondía con su proximidad al cuerpo más cercano.

Valor de la constante de Jacobi a lo largo del tiempo



Sin embargo, existe la posibilidad de que el problema sea otro. De cualquier modo, el proyecto no quedará abandonado, y pretendo arreglarlo y continuar mis investigaciones con él. Sin embargo, hasta que eso ocurra, me he visto obligado a usar software de terceros para seguir con el estudio de las órbitas de halo. Este último funciona de la misma manera que el mío, pero con una mayor precisión, que le permite llegar a resultados satisfactorios.

[15] Rubinsztejn, Ari

7. Bibliografía

7.1. Trabajos escritos

- L. Anderson, R. y S. Parker, J. Low-Energy Lunar Trajectory Design.
- Bernelli Zazzera, F., Massari, M. y Topputo, F. Assessment of Mission Design Including Utilization of Libration Points and Weak Stability Boundaries.
- Castelli, R., Dellnitz, M., Mingotti, G. and Zanzottera, A. Low-Energy Earth-to-Halo Transfers in the Earth–Moon Scenario with Sun-Perturbation.
- J. Dichmann, D., J. Doedel, E. y C. Paffenroth, R. (2002). The Computation of Periodic Solutions of 3-Body Problem Using the Numerical Continuation Software Auto.
- Eagle, David. Orbital Mechanics with MATLAB, The Hohmann Orbit Transfer.
- P. Gordon, Dawn. Transfers to Earth-Moon L2 Halo Orbits Using Lunar Proximity and Invariant Manifolds.
- J. Grebow, Daniel (2006). Generating Periodic Orbits in the Circular Restricted Threebody Problem with Applications to Lunar South Pole Coverage.
- D. Hall, C. y Kim, Mischa. Lyapunov and Halo Orbits about L2.
- S. Koon, Wang, W. Lo, Martin, E. Marsden, Jerrold, D. Ross, Shane. (2011). Dynamical Systems, the Three-Body Problem and Space Mission Design.
- Marcote Gerard Gómez, M., Masdemont, J. Trajectory Correction manoeuvres in the Transfer to Libration Point Orbits.
- McCaine, Gina (2003-04). Halo orbit design and optimization.
- R. Rausch, Raoul (2005). Earth to Halo Orbit Transfer Trajectories.
- E. Roberts, C. (2002). The SOHO Mission L1 Halo Orbit Recovery From the Attitude Control Anomalies of 1998.
- Sang Koon, W. CDS140B: Computation of Halo Orbit.

7.2. Recursos en línea

- Effective potential. [en línea]. Disponible en: https://en.wikipedia.org/wiki/Effective_potential
- Hohmann transfer orbit. [en línea]. Disponible en: https://en.wikipedia.org/wiki/Hohmann_transfer_orbit
- Lagrangian point. [en línea]. Disponible en: https://en.wikipedia.org/wiki/Lagrangian_point
- Newton's law of universal gravitation. [en línea]. Disponible en: https://www.math24.net/newtons-law-universal-gravitation/
- Rotating Frames. [en línea]. Disponible en: https://www.dpmms.cam.ac.uk/ stcs/courses/dynamics/-lecturenotes/section4.pdf
- Rotating reference frame. [en línea]. Disponible en: https://en.wikipedia.org/wiki/Rotating_reference_frame
- Rubinsztejn, Ari. Earth-Moon Colinear Lagrange Points Periodic Orbits. [base de datos]. Disponible en: https://arirubin.space/Earth-Moon_System/periodicCoLinear.html
- Rubinsztejn, Ari. Category: CR3BP. [en línea]. Disponible en: https://gereshes.com/category/math/astrodynamics/cr3b
- The Numerical Algorithm to Find the Halo Orbits. [en línea]. Disponible en: http://jjensen.org/Scans/ch4001.pdf
- Theory of Periodic Orbits. [en línea]. Disponible en: http://jjensen.org/Scans/ch2001.pdf
- Variedad (matemáticas). [en línea]. Disponible en: https://es.wikipedia.org/wiki/Variedad_(matem %C3 %A1ticas)

8. Anexos

8.1. Tablas de datos de la transferencia de Hohmann

8.1.1. Impulsos necesarios para llegar a la órbita deseada

Radio final [KM]	**Primera maniobra [m/s]**	**Segunda maniobra [m/s]**	**Impulso total [m/s]**
50000	2402	1372	3775
100000	2680	1234	3915
150000	2787	1114	3901
200000	2843	1022	3865
250000	2878	950	3828
300000	2901	891.7	3793
350000	2918	843	3762
400000	2931	802	3734
450000	2942	766	3708
500000	2950	735	3686

8.1.2. Diferencia de energía entre la órbita de partida y la de llegada

Radio final [Km]	**Diferencia de energía [MJ]**
50000	23.46
100000	25.12
150000	25.72
200000	26.03
250000	26.22
300000	26.34
350000	26.43
400000	26.5
450000	26.56
500000	26.6

8.1.3. Resultados analíticos y simulados para la transferencia de Hohmann

Radio final [Km]	Impulso (simulado) [m/s]	Error del resultado simulado	Impulso (analítico) [m/s]
50000	7.87	2.26	9
100000	19.5	1	19
150000	32.3	1.4	33
200000	48.3	3.4	50
250000	68.3	2.6	67
300000	85.9	0.2	86
350000	107	4	109
400000	135	4	133
450000	158.2	2.4	157
500000	183	2	182

8.2. Tablas de datos de la RTI

8.2.1. Tiempo de vuelo dentro de la variedad estable dependiendo de la amplitud de la órbita de Lyapunov de partida

1000 Km de radio [días]	2000 Km de radio [días]	3000 Km de radio [días]	4000 Km de radio [días]	5000 Km de radio [días]	6000 Km de radio [días]
4.23568768	4.2559365	4.28999993	4.3383027	4.40143162	4.48019536
3.41009686	3.41301085	3.41756836	3.4232719	3.42936792	3.43484796
3.14759193	3.14947637	3.15231031	3.15559837	3.15860091	3.16032138
2.98780328	2.98940416	2.99172373	2.99420257	2.99600248	2.99597619
2.87964609	2.88135084	2.88379637	2.88634545	2.88803315	2.88751334
2.80486896	2.80699781	2.81010959	2.81349507	2.81606302	2.81625537
2.75415898	2.75700693	2.76129367	2.76626025	2.77071347	2.77290078
2.7216697	2.72549842	2.73142471	2.73868023	2.74602137	2.75155739
2.70332658	2.70833034	2.71626108	2.72640056	2.73753569	2.74773903
2.69638849	2.70263572	2.71274162	2.72611794	2.74168922	2.75763184
2.69967416	2.70701233	2.71912384	2.73565963	2.75581251	2.77801923
2.71441301	2.72228648	2.73563389	2.75452608	2.7785996	2.80670071
2.74612283	2.75314439	2.76577763	2.78490371	2.81092523	2.84326256
2.80872579	2.81157883	2.8189006	2.83337587	2.85695617	2.88995741
2.93270443	2.92379808	2.91519969	2.9138965	2.92555834	2.95236151
3.1676423	3.13546633	3.09200534	3.05222604	3.03183185	3.03927181
3.51922692	3.46736818	3.38445143	3.28395116	3.19744857	3.16076154
3.90004853	3.84592072	3.75094853	3.61175281	3.44319086	3.32391835
4.26355326	4.21348677	4.12281222	3.97843319	3.76209225	3.53054148
4.62003277	4.57285535	4.48732252	4.34841969	4.1195093	3.77723935
4.99368587	4.94677686	4.86237559	4.72622366	4.4966414	4.07928473
5.40294682	5.35467703	5.26820269	5.12983943	4.89776843	4.49005395
5.85791088	5.80770151	5.71777819	5.57420605	5.33500197	5.29438467

8.2.2. Impulso de inserción en la variedad estable

Radio final [Km]	Impulso [Km/s]
500	0.37
1000	0.36
1500	0.36
2000	0.35

8.2.3. Impulso total

Radio final [Km]	Impulso [Km/s]
500	3.35
1000	3.22
1500	3.13
2000	3.03

8.2.4. Tiempo de vuelo final dependiendo del radio final

Radio final [Km]	Transferencia de Hohmann [días]	Muestra de RTI 1 [días]	Muestra de RTI 2 [días]	Muestra de RTI 3 [días]	Muestra de RTI 4 [días]	Muestra de RTI 5 [días]	Muestra de RTI 6 [días]
50000	0.3279166667	null	null	null	null	null	null
100000	0.7791666667	null	null	null	null	null	null
150000	1.345833333	null	null	null	null	null	null
200000	2.0125	null	null	null	null	null	null
250000	2.758333333	null	null	null	null	null	null
300000	3.579166667	23.5	17	23	28.4	120	123
325000	4.01	27.3	21.3	26.1	29.2	120.9	124.8
350000	4.458333333	30.6	26	30	31.8	122	126
400000	5.416666667	null	null	null	null	null	null
450000	6.416666667	null	null	null	null	null	null

8.2.5. Impulso total de la transferencia dependiendo del radio final

Radio final [Km]	Transferencia de Hohmann [m/s]	Muestra de RTI 1 [m/s]	Muestra de RTI 2 [m/s]	Muestra de RTI 3 [m/s]
50000	3775	null	null	null
100000	3915	null	null	null
150000	3901	null	null	null
200000	3865	null	null	null
250000	3828	null	null	null
300000	3793	300	350	325
325000	3776	500	550	650
350000	3762	800	900	1000
400000	3734	null	null	null
450000	3708	null	null	null
500000	3686	null	null	null

8.3. Código de desarrollo propio

8.3.1. Simulador de ‘n’ cuerpos

```
//Simulaci n de un sistema de 'n' cuerpos.
//La imagen 'noise' s necesaria para calcular una distribuci n normal de las part culas. Cualquier imagen de ruido gaussiano sirve.
//La variable 'size' representa la densidad de part culas generadas. Si es demasiado baja, convendr  subir la opacidad de las part culas

import java.util.*;

List<PVector> attractors = new ArrayList<PVector>();
List<Particle> particles = new ArrayList<Particle>();

float[] velo;
float[] mass;
float punt, dm, dr, col1, col2, col3;
float dmax=0;
float dmin=999999999;
PImage img;
int size = 10;
color c;
int a;

void setup() {
  for(int i=0; i<width;i+=size){
    for(int u=0; u<height;u+=size){
      particles.add(new Particle(i,u));
    }
  }
  velo = new float[particles.size()];
  mass = new float[particles.size()];

  img = loadImage("noise.png");
  image(img,0,0);
  for(int i=0; i<width;i+=size){
    for(int u=0; u<height;u+=size){
      c=(get(i, u));
      a = c & 0xFF;

      mass[((height/size))*int(i/size)+int(u/size)] = map(a,0,255,200,25);
    }
   }

  fullScreen();
}

void draw() {
  background(0);
  stroke(255);
  strokeWeight(.1);

  attractors.clear();
  for (int i = 0; i < particles.size(); i++) {
    Particle particle = particles.get(i);
    particle.system(i);
  }

  for (int i = 0; i < particles.size(); i++) {
    Particle particle = particles.get(i);

    for (int j = 0; j < attractors.size(); j++) {
      particle.attracted(attractors.get(j), mass[j]);
    }
    if(dr/dm>dmax){
      dmax=dr/dm;
```

```
    }
    if(dr/dm<dmin){
      dmin=dr/dm;
    }
    println(dmax,dmin);
    col1=map(dr/dm,50000,5,120,230);
    col2=map(dr/dm,50000,5,40,150);
    col3=map(dr/dm,50000,5,140,30);
    dm=0;
    dr=0;
    particle.update();
    particle.show(mass[i]/65,col1,col2,col3);
  }
  saveFrame();
}

class Particle {
  PVector pos;
  PVector posp;
  PVector prev;
  PVector vel;
  PVector acc;
  float mv = 0;

  Particle(float x, float y) {
    pos = new PVector(x, y);
    posp = new PVector(x, y);
    prev = new PVector(x, y);
    vel = new PVector(random(-mv,mv),random(-mv,mv));
    acc = new PVector();
  }

  void system(int i) {
    attractors.add(new PVector(this.pos.x, this.pos.y));
    velo[i]=mag(vel.x,vel.y);
  }

  void update() {
    vel.add(acc);
    vel.limit(50);
    pos.add(vel);
    acc.mult(0);
  }

  void show(float mass,float col1,float col2,float col3) {
    stroke(col1,col2,col3,150);
    strokeWeight(mass);
    point(this.pos.x, this.pos.y);
  }

  void attracted(PVector target, float mass) {
    PVector force = PVector.sub(target, pos);
    float d = force.mag();
    float G = .01;
    if(d<300){ //para compensar la falta de fuerzas fuera de la pantalla
      d = constrain(d, 5, 10000);
      dr+=d*d;
      dm += 1;
    }else{
      d=1000000;
    }
    float strength = mass * G / (d * d );
    force.setMag(strength);
    acc.add(force);
    //vel.mult(.999);
  }
```

```
}
```

Listing 1: Simulador de sistemas gravitatorios de 'n' cuerpos. Programa desarrollado con Processing.

8.3.2. Simulador de potencial efectivo de un sistema

```
//Simulaci n de un sistema de 2 cuerpos y su correpondiente potencial efectivo.
//La variable 'freq' modifica la resoluci n del resultado.

import java.util.*;

List<PVector> attractors = new ArrayList<PVector>();
List<Particle> particles = new ArrayList<Particle>();

float[] velo;
float[] mass;
float punt, dm, dr, col1, col2, col3, disp, disn, disp2, disn2,m0,m1,m2,max,x,velm;
float dmax=0;
float dmin=999999999;
int size = 50;
int a,freq;
PVector sol1,sol2,masscen;
boolean destruct;
PVector centre= new PVector(750,500);

void setup() {
  background(0);
  particles.add(new Particle(1250,500));
  particles.add(new Particle(centre.x,centre.y));

  freq=4;

  mass = new float[particles.size()];
  mass[0]=150;
  mass[1]=1000;

  velo = new float[particles.size()];
  velo[0]=5;

  fullScreen();
}

void draw() {
  background(0);
  stroke(255);
  strokeWeight(.1);

  attractors.clear();
  for (int i = 0; i < particles.size(); i++) {
    Particle particle = particles.get(i);
    if(i==0 ||i==1){
      particle.system();
    }
  }

  for (int i = particles.size()-1; i > -1 ; i--) {
    Particle particle = particles.get(i);

    for (int j = 0; j < attractors.size(); j++) {
      particle.attracted(attractors.get(j), mass[j]);
    }
    particle.update(i);
    if(destruct==false){
      particle.show();
    }else{
      particles.remove(i);
      destruct=false;
```

```
    }
  }

    for (int j = 0; j < width; j+=freq) {
      for (int k = 0; k < height; k+=freq) {
          strokeWeight(1);
          masscen = new PVector((mass[0]*posP.x+mass[1]*centre.x)/(mass[0]+mass[1]),(mass[0]*posP.y+mass[1]*centre.y)/(mass[0]+mass[1]));
          float one=+sq(((sqrt(mass[1]/abs(dist(masscen.x,masscen.y,posP.x,posP.y))))/dist(masscen.x,masscen.y,posP.x,posP.y)*dist(masscen.x,masscen.y,j,k)))/2+(mass[1]/dist(centre.x,centre.y,j,k)+mass[0]/dist(posP.x,posP.y,j,k));
          if(one>max && one<50){
            max=one;
          }
          for (float y = -5; y < 500; y+=.1) {
            if(one>y-.005 && one<y+.005){
              stroke(255);
              point(j,k);
            }
          }
      }
    }
    float one=+sq(((sqrt(mass[1]/abs(dist(masscen.x,masscen.y,posP.x,posP.y))))/dist(masscen.x,masscen.y,posP.x,posP.y)*dist(masscen.x,masscen.y,mouseX,mouseY)))/2+(mass[1]/dist(centre.x,centre.y,mouseX,mouseY)+mass[0]/dist(posP.x,posP.y,mouseX,mouseY));
    text(one,100,100);
}

PVector pos1,pos2,posP,velP;
float G = .1;
float dif=100;

class Particle {
  PVector pos;
  PVector posp;
  PVector prev;
  PVector vel;
  PVector acc;
  float mv = 0;

  Particle(float x, float y) {
    pos = new PVector(x, y);
    posp = new PVector(x, y);
    prev = new PVector(x, y);
    vel = new PVector(0,0);
    if(pos.x==1250 && pos.y==500){
      vel.add(-(pos.y-centre.y),pos.x-centre.x);
      vel.setMag(sqrt(G*1000/abs(dist(centre.x,centre.y,pos.x,pos.y))));
    }if(pos.y==centre.x && pos.x==centre.y){
      vel.add(0,0);
    }

    acc = new PVector();
  }

  void system() {
    attractors.add(new PVector(this.pos.x, this.pos.y));
  }

  void update(int particle) {
    vel.add(acc);
    vel.limit(50);
    pos.add(vel);
    acc.mult(0);

    if(particle==1){
      centre=pos;
    }
```

```
    if(particle==0){
      posP=pos;
      velP=vel;
    }
  }

  void show() {
    stroke(255,200);
    strokeWeight(5);
    point(this.pos.x, this.pos.y);
  }

  void attracted(PVector target, float mass) {
    PVector force = PVector.sub(target, pos);

    float d = force.mag();
    d = constrain(d, 50, 10000);
    float strength = mass * G / (d * d);
    force.setMag(strength);
    acc.add(force);

    if(mag(this.vel.x, this.vel.y)>velm){
      velm=mag(this.vel.x, this.vel.y);
    }
  }
}
```

Listing 2: Simulador del potencial efectivo de un sistema de 2 cuerpos. Programa desarrollado con Processing.

8.3.3. Simulación 3d de un sistema de 3 cuerpos y sus puntos de Lagrange

```
//Simulaci n de un sistema de 3 cuerpos, su potencial efectivo, y el comportamiento de
    objetos en sus puntos de Lagrange
//Pulsar '0' para obtener un resultado de buena calidad visual.

import java.util.*;
import peasy.PeasyCam;
PeasyCam cam;

List<PVector> attractors = new ArrayList<PVector>();
List<Particle> particles = new ArrayList<Particle>();
List<PVector> trajectory = new ArrayList<PVector>();

float[] velo;
float[] mass;
float punt, dm, dr, col1, col2, col3, max,f1,t;
float dmax=0;
float dmin=999999999;
PImage img;
int size = 50;
int a;
float x,velm;
float G = 1;

float freq=5;
PVector sol1,sol2,masscen;
PVector centre= new PVector(0,0);

void setup() {
  fullScreen(P3D);
  translate(width/2,height/2);
  cam = new PeasyCam(this, 1000);

  particles.add(new Particle(0,0,0,50,255,255,0));
  particles.add(new Particle(0,500,0,10,70,180,245));
  particles.add(new Particle(0,465.3319363,0,1,30,230,40));
  particles.add(new Particle(0,-500.2083333,0,1,30,230,40));
```

```
  particles.add(new Particle(0,534.66806372,0,1,30,230,40));
  particles.add(new Particle(465,183,0,1,30,230,40));
  particles.add(new Particle(-465,183,0,1,30,230,40));

  mass = new float[particles.size()];
  mass[0]=1000;
  mass[1]=1;
  mass[2]=.0000001;
  mass[3]=.0000001;
  mass[4]=.0000001;
  mass[5]=.0000001;
  mass[6]=.0000001;

  velo = new float[particles.size()];
  velo[0]=5;
}

void draw() {
  if(keyPressed){
    freq=.6;
  }
  background(255);
  stroke(255);
  strokeWeight(.1);

  attractors.clear();
  for(int i = 0; i < particles.size(); i++){
    Particle particle = particles.get(i);
    particle.system(i);
  }

  for(int i = 0; i < particles.size(); i++){
    Particle particle = particles.get(i);

    for(int j = 0; j < attractors.size(); j++){
      particle.attracted(attractors.get(j), mass[j]);
    }
    particle.update(i);
    particle.show();
  }

  for(int l = 0; l < trajectory.size(); l++){
    strokeWeight(3);
    stroke(255,0,0);
    point(trajectory.get(l).x,trajectory.get(l).y,trajectory.get(l).z);
  }

  for (float j = -700; j < 700; j+=freq) {
    for (float k = -sqrt(abs(490000-sq(j))); k < sqrt(abs(490000-sq(j))); k+=freq) {
      strokeWeight(1);
      masscen = new PVector((mass[1]*posP.x+mass[0]*centre.x)/(mass[1]+mass[0]),(mass[1]*
    posP.y+mass[0]*centre.y)/(mass[1]+mass[0]),0);
      float one=+sq(((sqrt(mass[0]/abs(dist(masscen.x,masscen.y,posP.x,posP.y))))/dist(
    masscen.x,masscen.y,posP.x,posP.y)*dist(masscen.x,masscen.y,j,k)))/2+(mass[0]/dist(
    centre.x,centre.y,j,k)+mass[1]/dist(posP.x,posP.y,j,k));
      for (float y = 0; y < 3.5; y+=.05) {
        if(one>y-.0075 && one<y+.0075){
          stroke(0,100);
          point(j,k,-one*100+296);
        }
      }
    }
    }
}

PVector pos1,pos2,posP,velP;
float dif=100;

```

```
class Particle {
  PVector pos;
  PVector posp;
  PVector prev;
  PVector vel;
  PVector acc;
  PVector mass;
  PVector col;
  float mv = 0;

  Particle(float x, float y, float z, float s, float c1, float c2, float c3) {
    pos = new PVector(x, y, z);
    posp = new PVector(x, y, z);
    prev = new PVector(x, y, z);
    vel = new PVector();
    mass = new PVector(s,s);
    col = new PVector(c1,c2,c3);
    if(pos.y==0){
      vel.add(0,0,0);
    }if(pos.y==500){
      vel.add(sqrt(2),0,0);
    }if(pos.y==465.3319363){
      vel.add(sqrt(2)/500*465.3319363,0,0);
    }if(pos.y==-500.2083333){
      vel.add(-sqrt(2)/500*500.2083333,0,0);
    }if(pos.y==534.66806372){
      vel.add(sqrt(2)/500*534.66806372,0,0);
    }if(pos.x==-465){
      vel.add(0.517602,1.31522,0);
    }if(pos.x==465){
      vel.add(0.517602,-1.31522,0);
    }

    acc = new PVector();
  }

  void system(int i) {
    attractors.add(new PVector(this.pos.x, this.pos.y, this.pos.z));
    velo[i]=mag(vel.x,vel.y,vel.z);
  }

  void update(int particle) {
    vel.add(acc);
    vel.limit(50);
    pos.add(vel);
    acc.mult(0);

    if(particle==0){
      centre=pos;
    }
    if(particle==1){
      posP=pos;
      velP=vel;
    }
  }

  void show() {
    stroke(this.col.x,this.col.y,this.col.z);
    strokeWeight(10);
    pushMatrix();
    translate(this.pos.x,this.pos.y,this.pos.z);
    trajectory.add(new PVector(this.pos.x, this.pos.y, this.pos.z));
    if(trajectory.size()>2000){
      trajectory.remove(0);
    }
    sphere(this.mass.x);
    popMatrix();
  }
```

```

  void attracted(PVector target, float mass) {
    PVector force = PVector.sub(target, pos);
    float d = force.mag();
    d = constrain(d, 5, 10000);
    float strength = mass * G / (d * d);
    force.setMag(strength);
    acc.add(force);

    if(mag(this.vel.x, this.vel.y, this.vel.z)>velm){
      velm=mag(this.vel.x, this.vel.y, this.vel.z);
    }
  }
}
```

Listing 3: Programa utilizado para las representaciones del problema de los 3 cuerpos y sus puntos de Lagrange. Desarrollado con Processing.

8.3.4. Simulador del CR3BP

```
//Simulaci n de un sistema de 3 cuerpos en un sistema rotatorio no-inercial.
//presionar 'o' para ver la constante de Jacobi en forma de l neas equipotenciales.
//presionar 'O' para ver la imagen con m s calidad visual.

import java.util.*;
import peasy.PeasyCam;
PeasyCam cam;

float def=.01;
float ratio=.01215;
boolean orb;
PVector pos = new PVector(1.03,-.05,-.03);
PVector vel = new PVector(.0003,.0000,.0004);
PVector acc = new PVector();
PVector pos1=new PVector(-ratio,0,0);
PVector pos2=new PVector(1-ratio,0,0);
List<PVector> trajectory = new ArrayList<PVector>();

void setup() {
  fullScreen(P3D);
  cam = new PeasyCam(this, 500,0,0,500);
}

void draw() {
  background(255);

  if(orb==true){
    for (float j = -5; j < 5; j+=def) {
        for (float k = -5; k < 5; k+=def) {
          float jacobi=(sq(j)+sq(k))+2*(1-ratio)/dist(pos1.x,pos1.y,pos1.z,j,k,0)+2*ratio/dist(pos2.x,pos2.y,pos2.z,j,k,0);

          float l1=3;
          float l2=3.5;

          for(float lin=l1;lin<l2;lin+=.1){
            if(jacobi>lin-.01 && jacobi<lin+.01){
              strokeWeight(3);
              stroke(map(jacobi,l1,l2,0,255),map(jacobi,l1,l2,150,0),map(jacobi,l1,l2,255,0),20);
              point(map(j,-5,5,-5000,5000),map(k,-5,5,-5000,5000),-jacobi*200+635);
            }
          }
        }
      }
  }

```

```
  stroke(255);
  trajectory.add(new PVector(map(pos.x,-5,5,-5000,5000),map(pos.y,-5,5,-5000,5000),map(pos.z
    ,-5,5,-5000,5000)));
  for (int l = 0; l < trajectory.size()-1; l++) {
    strokeWeight(2.5);
    //point(trajectory.get(l).x, trajectory.get(l).y, trajectory.get(l).z);
    stroke(0,255,0);
    line(trajectory.get(l).x, trajectory.get(l).y, trajectory.get(l).z, trajectory.get(l+1).
    x, trajectory.get(l+1).y, trajectory.get(l+1).z);
    stroke(0);
  }

  if(dist(pos1.x,pos1.y,pos1.z,pos.x,pos.y,pos.z)>.02){
    acc=new PVector(2*vel.y + pos.x -(1-ratio)*(pos.x+ratio)/pow(sqrt(sq(pos.x+ratio)+sq(pos
    .y)+sq(pos.z)),3) -ratio*(pos.x-1+ratio)/pow(sqrt(sq(pos.x-1+ratio)+sq(pos.y)+sq(pos.z))
    ,3),
    -2*vel.x + pos.y -(1-ratio)*pos.y/pow(sqrt(sq(pos.x+ratio)+sq(pos.y)+sq(pos.z)),3) -
    ratio*pos.y/pow(sqrt(sq(pos.x-1+ratio)+sq(pos.y)+sq(pos.z)),3),
    -(1-ratio)*pos.z*pow(sqrt(sq(pos.x+ratio)+sq(pos.y)+sq(pos.z)),-3) - ratio*pos.z*pow(
    sqrt(sq(pos.x-1+ratio)+sq(pos.y)+sq(pos.z)),-3));
  }

  acc.mult(.000001);
  vel.add(acc);
  pos.add(vel);

  strokeWeight(75);
  point(map(-ratio,-5,5,-5000,5000),0,0);
  strokeWeight(10);
  point(map(1-ratio,-5,5,-5000,5000),0,0);
  strokeWeight(3);
  point(map(pos.x,-5,5,-5000,5000),map(pos.y,-5,5,-5000,5000),map(pos.z,-5,5,-5000,5000));
}

void keyPressed(){
  if(key==CODED){
    if(keyCode==UP){
      def-=.002;
    }if(keyCode==DOWN){
      def+=.002;
    }
  }if(key=='0'){
      def=.001;
  }if(key=='1'){
      def=.1;
  }
  if(key=='o'){
    if(orb==false){
      orb=true;
    }else{
      orb=false;
    }
  }
}
```

Listing 4: Simulador del Problema de los 3 cuerpos Circular Reducido. Programa desarrollado con Processing.

8.3.5. Algoritmo selector de órbitas periódicas en el CR3BP

```
//Simulaci n de un sistema de 3 cuerpos en un sistema rotatorio no-inercial.
//Cuenta con un algoritmo de selecci n de  rbitas  peri dicas (beta).
//La sensibilidad del selector est  representada como la variable 'di2', y se deber
    ajustar para cada caso.

float t,jac;
float G=1;
float m1=1000;
```

```
float m2=100;
float M=m1+m2;
float mu1=G*m1;
float mu2=G*m2;
float R=mu1+mu2;
float w=sqrt(G*M/pow(R, 3));
float ratio=m2/(m1);

PVector pos1;
PVector pos2;

import java.util.*;
import peasy.PeasyCam;
PeasyCam cam;

List<PVector> attractors = new ArrayList<PVector>();
List<Particle> particles = new ArrayList<Particle>();
List<PVector> trajectory = new ArrayList<PVector>();

float[] velo;
float[] mass;
float punt, dm, dr, col1, col2, col3, max, f1;
float dmax=0;
float dmin=999999999;
PImage img;
int size = 50;
float x, velm;

PVector sol1, sol2, masscen;
PVector centre= new PVector(0, 0);

boolean jaco1,jaco2,jaco3,orb,ciclo,first,passed,repet;
float freq=100;
float jac1;
float ma=0;
float mi=500;
float di1=1;
float di2=.001;
int rep;

void setup() {
  fullScreen(P3D);
  translate(width/2, height/2);
  cam = new PeasyCam(this, 1000);

  particles.add(new Particle(-ratio, 0, 0, 0, 0, 0, 50, m1,0,0,0));
  particles.add(new Particle(1-ratio, 0, 0, 0, 0, 0, 20, m2,0,0,0));

  for (float i = ma; i < mi; i+=di1) {
    particles.add(new Particle(1.08,0,0,i,.08,0,5,0,i,0,0));
  }

  first=true;
}

void draw() {
  t+=.001;
  background(0);
  stroke(255);
  strokeWeight(.1);

  for (int i = 0; i < particles.size(); i++) {
    Particle particle = particles.get(i);
    particle.update(i);
    particle.show();
  }
  if(repet==true){
    repet=false;
```

```
    for (int i = 2; i < particles.size(); i++) {
      Particle particle = particles.get(i);
      println(mi,ma,"lol");
      particle.repe(map(i,0,particles.size(),mi,ma));
    }
    mi=500;
    ma=0;
  }

  for (int l = 0; l < trajectory.size()-1; l++) {
    if (orb==true) {
      strokeWeight(2.5);
      stroke(255, 0, 0);
      point(trajectory.get(l).x, trajectory.get(l).y, trajectory.get(l).z);
    }
  }
  if(jaco1==true){
    for (int j = -750; j < 750; j+=freq) {
      for (float k = -sqrt(abs(562500-sq(j))); k < sqrt(abs(562500-sq(j))); k+=freq) {
        stroke(255);
        strokeWeight(2);
        float jacobi=(sq(map(j,-2500,2500,-5,5))+sq(map(k,-2500,2500,-5,5)))+2*(1-ratio)/dist(map(pos1.x,-5,5,-2500,2500),map(pos1.y,-5,5,-2500,2500),map(pos1.z,-5,5,-2500,2500),j,k,0)+2*ratio/dist(map(pos2.x,-5,5,-2500,2500),map(pos2.y,-5,5,-2500,2500),map(pos2.z,-5,5,-2500,2500),j,k,0);

        for(float lin=0;lin<2;lin+=.1){
          if(jacobi>lin-.01 && jacobi<lin+.01){
            point(j,k,-jacobi*100);
          }
        }
      }
    }
  }
  if(jaco2==true){
    for (int j = -1250; j < 1251; j+=1) {
      for (int k = -1250; k < 1251; k+=freq) {
        stroke(255,50);
        strokeWeight(2);
        float jacobi=(sq(map(j,-750,750,-.1,.1))+sq(map(k,-750,750,-.1,.1)))+2*(1-ratio)/dist(map(pos1.x,-5,5,-2500,2500),map(pos1.y,-5,5,-2500,2500),map(pos1.z,-5,5,-2500,2500),j,k,0)+2*ratio/dist(map(pos2.x,-5,5,-2500,2500),map(pos2.y,-5,5,-2500,2500),map(pos2.z,-5,5,-2500,2500),j,k,0);
        point(j,k,-jacobi*10000);
      }
    }
    for (int j = -1250; j < 1251; j+=freq) {
      for (int k = -1250; k < 1251; k+=1) {
        stroke(255,50);
        strokeWeight(2);
        float jacobi=(sq(map(j,-750,750,-.1,.1))+sq(map(k,-750,750,-.1,.1)))+2*(1-ratio)/dist(map(pos1.x,-5,5,-2500,2500),map(pos1.y,-5,5,-2500,2500),map(pos1.z,-5,5,-2500,2500),j,k,0)+2*ratio/dist(map(pos2.x,-5,5,-2500,2500),map(pos2.y,-5,5,-2500,2500),map(pos2.z,-5,5,-2500,2500),j,k,0);
        point(j,k,-jacobi*10000);
      }
    }
  }
  if(jaco3==true){
    for (int j = -750; j < 750; j+=freq) {
      for (float k = -sqrt(abs(562500-sq(j))); k < sqrt(abs(562500-sq(j))); k+=freq) {
        stroke(255);
        strokeWeight(2);
        float jacobi=(sq(map(j,-2500,2500,-1,1))+sq(map(k,-2500,2500,-1,1)))+2*(1-ratio)/dist(map(pos1.x,-5,5,-2500,2500),map(pos1.y,-5,5,-2500,2500),map(pos1.z,-5,5,-2500,2500),j,k,0)+2*ratio/dist(map(pos2.x,-5,5,-2500,2500),map(pos2.y,-5,5,-2500,2500),map(pos2.z,-5,5,-2500,2500),j,k,0);

```

```
        if(jacobi>map(mouseX,0,width,0,.1)-.002 && jacobi<map(mouseX,0,width,0,.1)+.002){
          stroke(0,0,255);
        }else{
          stroke(255);
        }
        strokeWeight(2);
        point(j,k,-jacobi*1000);
      }
    }
  }
}

void keyReleased(){
  ciclo=false;
  if(key=='o'){
    if(orb==false){
      orb=true;
    }else{
      orb=false;
    }
  }
  if(key=='j'){
    if(jaco1==false && jaco2==false && jaco3==false && ciclo==false){
      jaco1=true;
      jaco2=false;
      jaco3=false;
      ciclo=true;
    }if(jaco1==true && jaco2==false && jaco3==false && ciclo==false){
      jaco1=false;
      jaco2=true;
      jaco3=false;
      ciclo=true;
    }if(jaco1==false && jaco2==true && jaco3==false && ciclo==false){
      jaco1=false;
      jaco2=false;
      jaco3=true;
      ciclo=true;
    }if(jaco1==false && jaco2==false && jaco3==true && ciclo==false){
      jaco1=false;
      jaco2=false;
      jaco3=false;
      ciclo=true;
    }
    ciclo=true;
  }
  if(jaco2==true){
    if(key=='1'){
      freq=200;
    }if(key=='2'){
      freq=150;
    }if(key=='3'){
      freq=100;
    }if(key=='4'){
      freq=50;
    }if(key=='5'){
      freq=25;
    }if(key=='6'){
      freq=15;
    }if(key=='7'){
      freq=10;
    }if(key=='8'){
      freq=5;
    }if(key=='9'){
      freq=3;
    }
  }
  if(jaco1==true || jaco3==true){
    if(key=='1'){
```

```
      freq=100;
    }if(key=='2'){
      freq=75;
    }if(key=='3'){
      freq=50;
    }if(key=='4'){
      freq=25;
    }if(key=='5'){
      freq=15;
    }if(key=='6'){
      freq=10;
    }if(key=='7'){
      freq=5;
    }if(key=='8'){
      freq=3;
    }if(key=='9'){
      freq=1;
    }
  }
}

PVector posP,velP;
float dif=100;

float redu=.00001;

class Particle {
  PVector pos;
  PVector posp;
  PVector prev;
  PVector vel;
  PVector acc;
  PVector mass;
  PVector col,data;
  float mv = 0;

  Particle(float x, float v1, float y, float v2, float z, float v3, float s, float m, float
    d1, float d2, float d3) {
    pos = new PVector(x, y, z);
    posp = new PVector(x, y, z);
    prev = new PVector(x, y, z);
    vel = new PVector(v1*redu,v2*redu,v3*redu);
    mass = new PVector(s,m);
    acc = new PVector();
    data = new PVector(d1,d2,d3);
  }

  void update(int particle) {
    pos1=new PVector(-ratio,0,0);
    pos2=new PVector(1-ratio,0,0);
    if(particle>=2){
      acc.mult(0);
      float r1=sqrt(sq(this.pos.x+ratio)+sq(this.pos.y));
      float r2=sqrt(sq(this.pos.x-1+ratio)+sq(this.pos.y));
      float a=1;
      float b=1;
      acc=new PVector(a*(2*this.vel.y+this.pos.x)-((1-ratio)*(this.pos.x+ratio)/pow(dist(
    pos1.x,pos1.y,pos1.z,this.pos.x,this.pos.y,this.pos.z),3)+ratio*(this.pos.x-1+ratio)/pow
    (dist(pos2.x,pos2.y,pos2.z,this.pos.x,this.pos.y,this.pos.z),3))*b,
      a*(-2*this.vel.x+this.pos.y)-((1-ratio)*this.pos.y/pow(dist(pos1.x,pos1.y,pos1.z,this.
    pos.x,this.pos.y,this.pos.z),3)+ratio*this.pos.y/pow(dist(pos2.x,pos2.y,pos2.z,this.pos.
    x,this.pos.y,this.pos.z),3))*b,
      (-(1-ratio)*this.pos.z/pow(dist(pos1.x,pos1.y,pos1.z,this.pos.x,this.pos.y,this.pos.z)
    ,3)-ratio*this.pos.z/pow(dist(pos2.x,pos2.y,pos2.z,this.pos.x,this.pos.y,this.pos.z),3))
    *b);
      strokeWeight(2);
      acc.mult(redu);
      acc.limit(.01);
```

```
      vel.add(acc);
      pos.add(vel);


      float sens=.000253;
      if(millis()>1000 && mass.y<1 && this.pos.x>.5 && this.pos.z<0 && this.pos.y>=-di2 && this.pos.y<=di2 && this.vel.z>=-di2 && this.vel.z<=di2 && this.vel.x>=-di2 && this.vel.x<=di2){
        mass.x=6;
        passed=true;
        if(this.data.x<mi){
          mi=this.data.x;
        }if(this.data.x>ma){
          ma=this.data.x;
        }
        println(mi,ma);
      }if(((millis()>20000 + 20000*rep && mass.y<1))){
        if(passed==true){
          passed=false;
          di1*=.9;
          di2*=.9;
        }else{
          di1*=.9;
          di2*=1.05;
        }
        repet=true;
        rep++;
      }
    }
  }

  void show() {
    float x,y,z;
    pushMatrix();
    translate(map(this.pos.x,-5,5,-2500,2500),map(this.pos.y,-5,5,-2500,2500),map(this.pos.z,-5,5,-2500,2500));
    strokeWeight(mass.x);
    if(mass.x==6){
      stroke(0,255,0);
    }else{
      stroke(255);
    }
    point(0,0,0);
    popMatrix();

    if(mass.y<1){
      text(data.x+"  "+data.y+"  "+data.z,map(this.pos.x,-5,5,-2500,2500),map(this.pos.y,-5,5,-2500,2500),map(this.pos.z,-5,5,-2500,2500));
      if(first==true){
        jac1=(sq(map(this.pos.x,-2500,2500,-1,1))+sq(map(this.pos.y,-2500,2500,-1,1)))+2*(1-ratio)/dist(map(pos1.x,-5,5,-2500,2500),map(pos1.y,-5,5,-2500,2500),map(pos1.z,-5,5,-2500,2500),this.pos.x,this.pos.y,0)+2*ratio/dist(map(pos2.x,-5,5,-2500,2500),map(pos2.y,-5,5,-2500,2500),map(pos2.z,-5,5,-2500,2500),this.pos.x,this.pos.y,0)-sq(mag(this.vel.x,this.vel.y,this.vel.z));
        //println(jac1);
        first=false;
      }

      trajectory.add(new PVector(map(this.pos.x,-5,5,-2500,2500),map(this.pos.y,-5,5,-2500,2500),map(this.pos.z,-5,5,-2500,2500)));
      if(trajectory.size()>200000){
        trajectory.remove(0);
      }
    }
  }

  void repe(float i){
    pos=new PVector(1.08,0,.08);
```

```
    vel=new PVector(0,i*redu,0);
    mass.x=5;
  }
}
```

Listing 5: Algoritmo selector de órbitas periódicas en el Problema de los 3 cuerpos Circular Reducido. Programa desarrollado con Processing.

8.4. Código ajeno

Los simuladores utilizados en este trabajo (el simulador de transferencias de Hohmann, y el simulador del CR3BP) se pueden encontrar respectivamente en las siguientes páginas web:

- https://es.mathworks.com/matlabcentral/fileexchange/38942-the-hohmann-orbit-transfer
- https://github.com/gereshes/Matlab-Astrodynamics-Library