



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA AEROESPACIAL EN AERONAVEGACIÓN

TRABAJO FIN DE GRADO

**DISEÑO PRELIMINAR DE MISIONES INTERPLANETARIAS
EMPLEANDO MANIOBRAS ASISTIDAS POR GRAVEDAD**

Autor: JOSE MARÍA HERRERA SAMPABLO

Tutor: HODEI URRUTXUA CEREIRO

Diseño preliminar de misiones interplanetarias empleando maniobras asistidas por gravedad

Autor

Jose María Herrera Sampablo

Tutor/es

Hodei Urrutxua Cereijo

Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación



Escuela Técnica Superior
de Ingeniería de Telecomunicación

Ingeniería Aeroespacial en Aeronavegación



Escuela Técnica Superior
de Ingeniería de Telecomunicación



Universidad
Rey Juan Carlos

MADRID, Septiembre 2019

Agradecimientos

En primer lugar, a mis padres, por todo su esfuerzo y dedicación en hacer de mí una mejor persona, y por toda el apoyo, la paciencia y la ayuda que me han brindado, sin la cuál no habría sido posible estar donde hoy estoy.

A mis hermanas Eva, Isabel y Cristina, las cuales me han orientado y ayudado en todas y cada una de mis decisiones, y que han sido ejemplo en el cuál fijarme durante todos estos años, y en los venideros.

A todos los compañeros de clase, por compartir estos maravillosos años. En especial, a Isa y Deyna, y también a todos los que formaron parte de mi Erasmus por compartir una experiencia tan única.

No me gustaría olvidarme, por último, de mi tutor, Hodei, por plantearme este tema, transmitirme su pasión y, sobre todo, por su infinita paciencia y por hacerme un hueco siempre que lo he necesitado.

A todos vosotros, y a todos aquellos que me dejó por nombrar, muchas gracias por formar parte en este camino.

Resumen

El estudio de las misiones de exploración interplanetaria parte siempre de un análisis preliminar de las posibles trayectorias que permiten cumplir con el propósito final. Este tipo de estudios incurren en modelos simplificados tanto del Sistema Solar como de los métodos empleados en la definición de las transferencias implicadas.

En las últimas décadas, además, las trayectorias que han empleado asistencias gravitacionales en diferentes planetas del Sistema Solar han sido cada vez más y más empleadas debido a los buenos resultados obtenidos en términos enérgéticos.

Buscando emprender un estudio preliminar de misión, se ha desarrollado e implementado un modelo que emplea la maniobra de asistencia gravitacional y recurre al método de ajuste de cónicas y el problema de Lambert para encontrar aquella trayectoria que minimiza el valor del impulso requerido en misiones hacia planetas exteriores.

Índice general

Agradecimientos	iv
Resumen	v
1. Introducción	1
1.1. Objetivos	2
1.2. Estructura del trabajo	2
2. Movimiento Orbital Kepleriano	3
2.1. Órbita Kepleriana	3
2.2. Elementos Orbitales	3
2.2.1. Elementos orbitales a partir de vector de estado conocido	6
2.2.2. Vector de estado a partir de los elementos orbitales	8
2.3. Posición orbital en función del tiempo. Ecuación de Kepler	9
2.3.1. Trayectorias Elípticas	11
2.3.2. Trayectorias hiperbólicas	13
3. Maniobra de Asistencia Gravitacional	15
3.1. Modelo del fly-by	15
3.2. Órbitas hiperbólicas en el seno de la gravisfera	16
3.3. Fly-by. Análisis de la maniobra	19
3.4. Efecto sobre los elementos orbitales	22
3.4.1. La incongruencia energética	22
3.5. Efecto sobre los elementos orbitales. Continuación	23
4. Modelo de Ajuste de Cónicas	27
4.1. Esfera de influencia gravitacional	27
4.2. Modelo de ajuste de cónicas	29
4.3. El problema de Lambert	30
4.4. Efemérides planetarias	34
5. Diseño de Trayectorias Interplanetarias	38
5.1. Problema matemático	38
5.1.1. Método Práctico	39
5.2. Búsqueda de la transferencia de menor ΔV_{total}	43
6. Implementación Práctica del Problema	45
6.1. Implementación y código	45

7. Caso Práctico	53
7.1. Tierra - Venus - Júpiter	53
7.1.1. Trayectoria Tierra - Venus. Porkchop plots	54
7.1.2. Fly-by en Venus y llegada a Júpiter	55
7.1.3. Búsqueda del mínimo ΔV_{total} para la misión Tierra - Venus - Júpiter	57
7.2. Otros Itinerarios	58
8. Conclusiones y Trabajos Futuros	60
8.1. Conclusiones	60
8.2. Líneas de trabajo futuro	61
Bibliografía	62
A. Anexo I - Gráficas Capítulo 7	64
A.1. Fechas y tiempos de vuelo fijados para las misiones	64
A.2. Gráficas Misiones	65
A.2.1. Gráficas Optimizadas Tierra-Venus-Júpiter	65
A.2.2. Gráficas Tierra-Tierra-Júpiter	66
A.2.3. Gráficas Tierra-Marte-Júpiter	67
A.2.4. Gráficas Tierra-Venus-Neptuno	68
A.2.5. Gráficas Tierra-Tierra-Neptuno	69
A.2.6. Gráficas Tierra-Júpiter-Neptuno	70
A.3. Optimización ΔV_{total}	71
B. Anexo II - Código	72
B.1. planet_consts.py	72
B.2. core.py	78
B.3. main.py	87
B.4. validation.py	93
C. Anexo III - Validación del Código	99
C.1. Validación de las funciones <i>vstate_to_coe</i> y <i>coe_to_vstate</i>	99
C.2. Validación de la función <i>kepler_eliptic</i>	100
C.3. Validación de la función <i>planet_state</i>	101
C.4. Validación de la función <i>lambert_problem</i>	103
C.5. Validación de los pork-chop plot	104
C.6. Validación manual de la maniobra de asistencia por gravedad	105
C.6.1. Maniobra de asistencia para el caso $r_p > r_{planeta}$	105
C.6.2. Maniobra de asistencia para el caso $r_p < r_{planeta}$	107

Índice de figuras

2.1. Elementos Orbitales. Figura tomada de (Curtis (2005)).	4
2.2. Definición de las anomalías. Figura tomada de (Anónimo (2010)).	5
2.3. Parámetros de la hipérbola. Figura adaptada de (Curtis (2005)).	14
3.1. Rotación del vector velocidad durante la maniobra de fly-by. Figura tomada de (Labunsky y cols. (1998)).	17
3.2. Posibles maniobras de asistencia gravitacional. Figuras (a) y (b) tomadas de (Labunsky y cols. (1998)).	19
3.3. Maniobra de asistencia. Vectores coplanarios o casi-coplanarios. Figura tomada de (Labunsky y cols. (1998)).	20
3.4. Análisis vectorial del fly-by. Figura tomada de (Labunsky y cols. (1998)).	23
4.1. Fases de una trayectoria interplanetaria. Figura tomada de (Fernández (2014)).	30
4.2. Problema de Lambert para una órbita entre la Tierra y Marte. Figura tomada de (Fernández (2014)).	31
5.1. Definición de una trayectoria interplanetaria. El caso a) refiere la transferencia entre el planeta 1 y 2, y el caso b) refiere la transferencia entre el planeta 2 y 3.	43
6.1. Diagrama de flujo que relaciona las funciones implementadas.	52
7.1. Pork-chop plot de v_∞ para la transferencia Tierra-Venus	54
7.2. Tipos de trayectorias. Figurada tomada de (Fernández (2014)).	55
7.3. Mallado que representa la magnitud ΔV_{flyby} para cada posible trayectoria	56
7.4. Ratio $r_p/r_{planeta}$ generado durante la maniobra de fly-by	56
7.5. Variación del ángulo φ durante la maniobra de asistencia	57
7.6. Magnitud ΔV_{total} para la misión Tierra-Venus-Júpiter	57
A.1. Gráficas de la optimización de la misión Tierra-Venus-Júpiter	65
A.2. Gráficas Misión Tierra-Tierra-Júpiter	66
A.3. Gráficas Misión Tierra-Marte-Júpiter	67
A.4. Gráficas Misión Tierra-Venus-Neptuno	68
A.5. Gráficas Misión Tierra-Tierra-Neptuno	69
A.6. Gráficas Misión Tierra-Júpiter-Neptuno	70
A.7. Optimización Δv_{total}	71
C.1. Pork-chop plot generados	104
C.2. Pork-chop plot de referencia	104

Índice de tablas

4.1. Elementos keplerianos y sus ratios de cambio respecto a la época J2000, válidos para el intervalo 3000 BC - 3000 AD	35
4.2. Términos adicionales para calcular M en Júpiter, Saturno, Urano, Neptuno y Plutón, válidos para el intervalo 3000 BC - 3000 AC	36
7.1. Resultados para una misión Tierra-Venus-Júpiter	58
7.2. Primera aproximación para ΔV_{total}	58
7.3. Optimización de ΔV_{total}	59
A.1. Fechas y tiempos de vuelo para la misión Tierra-Tierra-Júpiter	64
C.1. Validación de las funciones que transforman coordenadas cartesianas en elementos orbitales y viceversa	100
C.2. Validación de la función <i>kepler_eliptic</i>	101
C.3. Validación de la función <i>julian_date</i>	101
C.4. Validación de la función <i>planet_elements</i>	102
C.5. Validación de la función <i>planet_state</i>	102
C.6. Validación de la función <i>lambert_problem</i>	103
C.7. Validación de la función <i>r_periapse</i>	107
C.8. Validación de la función <i>fly_by</i>	109

Índice de Códigos

6.1.	Función que retorna el vector de estado de cada planeta	45
6.2.	Funciones referidas en planet_state	46
6.3.	Función que devuelve la transferencia heliocéntrica entre dos planetas.	47
6.4.	Bucle for que retornará los valores de los impulsos necesarios.	48
6.5.	Función que evalúa el valor del parámetro r_p	49
6.6.	Función que evalúa la maniobra de asistencia por gravedad.	49
6.7.	Ejemplo de creación de los arrays de fechas y tiempos.	50
6.8.	Cálculo del mínimo ΔV_{total} y las fechas en las que se obtiene.	51
6.9.	Re-ajuste de las constantes de fechas.	51
6.10.	Búsqueda de una nueva solución mínima.	52

1. Introducción

La exploración del Universo siempre ha sido uno de los hitos más perseguidos a lo largo de la historia, llegando a su punto de mayor esplendor durante la época de la carrera espacial.

Esta etapa condujo al desarrollo de las primeras misiones enviadas fuera de la atmósfera terrestre, teniendo como punto culminante la llegada del hombre a la Luna por parte de Estados Unidos en Julio de 1969. Sin embargo, más allá del afán por conquistar la Luna, se desarrollaron otras líneas de trabajo sobre la exploración planetaria en el Sistema Solar.

Un ejemplo fue el programa Venera de la Unión Soviética, que convirtió la sonda Venera 3 en el primer objeto de fabricación humana en impactar en la superficie de otro planeta.

El desarrollo tecnológico que se ha sucedido desde aquella época ha multiplicado este tipo de misiones, existiendo en la actualidad misiones dirigidas hacia diferentes cuerpos celestes como Mercurio, Saturno, Plutón, incluso el Sol y algunos asteroides. Es más, las sondas Voyager 1 y Voyager 2, actualmente son los primeros objetos en navegar el espacio interestelar.

Sin embargo, no sólo se ha avanzado tecnológicamente, sino que también se han producido importantes avances en el conocimiento y los métodos empleados. El mejor ejemplo se encuentra en el planteamiento de la maniobra de asistencia gravitacional como método impulsivo en este tipo de misiones, lo cuál generó un amplio debate, para convertirse posteriormente en una de las bases de las trayectorias interplanetarias.

En épocas anteriores se solucionaba el teorema de Lambert para definir trayectorias heliocéntricas, y los resultados eran expuestos en las famosas pork-chop plots. Los ingenieros del *Jet Propulsion Laboratory JPL* de la NASA realizaron una gran cantidad de pork-chop plots buscando la opción óptima de trayectoria para las misiones Voyager.

En la actualidad, estos métodos han sido perfeccionados recurriendo a algoritmos de optimización más potentes y complejos, que son capaces de retornar una solución óptima a partir de unas condiciones y requisitos inciales impuestos.

Estas soluciones óptimas no son otras que aquellas trayectorias que permiten cumplir el propósito fijado potenciando las posibilidades científicas y reduciendo todo lo posible el consumo energético requerido.

Sin embargo, llegar a este punto es siempre un proceso complicado, que tiene como etapa de partida un estudio preliminar, cuyo objetivo será estimar aquellas trayectorias que resultan de interés para después ser re-calculadas, seleccionadas y finalmente, emprendidas.

1.1. Objetivos

El objetivo fundamental de este trabajo será emprender un análisis preliminar de misión, encontrando aquellas trayectorias interplanetarias que minimicen el impulso energético requerido en misiones hacia planetas exteriores.

La búsqueda de este objetivo final se fundamentará en la consecución de los siguientes objetivos específicos:

- Definir la dinámica del movimiento orbital y de los elementos que definen el estado de la nave en el espacio.
- Exponer los resultados de efectuar una maniobra de asistencia gravitacional.
- Presentar el modelo empleado para el análisis preliminar de la trayectoria cuya base es construida a partir del modelo de ajuste de cónicas, el concepto de esfera de influencia gravitacional y el problema de Lambert.
- Desarrollar un programa en Python para ejecutar el análisis preliminar.
- Comparar una serie de itinerarios hacia planetas exteriores realizando una asistencia gravitatoria en diferentes planetas.

1.2. Estructura del trabajo

De acuerdo a lo anterior, la estructura de este trabajo se divide en dos partes diferenciadas.

En primer lugar, los capítulos 2, 3 y 4 establecen las bases y herramientas necesarias para un análisis preliminar: el capítulo 2 define una órbita Kepleriana no perturbada y las herramientas necesarias para describirla y encontrar el estado de un cuerpo orbitando en ella en cualquier instante t ; el capítulo 3 modeliza la maniobra de asistencia gravitacional que impulsará la nave en su camino al planeta destino, y por último, el capítulo 4 describe la manera en que las transferencias heliocéntricas son calculadas mediante la resolución del problema de Lambert y aproxima estas transferencias a la trayectoria planetocéntrica de la nave empleando el modelo de ajuste de cónicas.

La segunda parte incluye los capítulos 5, 6 y 7: el capítulo 5 expone un algoritmo para resolver matemáticamente la trayectoria entre los planetas fijados empleando las herramientas descritas en la primera parte; el capítulo 6 implementa, por medio de Python, un programa que permite obtener y optimizar una solución a la trayectoria deseada, y por último, el capítulo 7 itera una secuencia de misiones cuyas trayectorias serán analizadas de forma preliminar, encontrando para ellas una solución óptima que minimice el impulso requerido.

Para concluir, en el capítulo 8 se expondrán las conclusiones extraídas, y se presentarán una serie de posibles líneas de trabajo futuro, que permitan mejorar la herramienta creada y orientarla a análisis más complejos.

2. Movimiento Orbital Kepleriano

2.1. Órbita Kepleriana

Fue **Johannes Kepler**, astrónomo y matemático alemán, quién a principios del siglo XVII describió mediante sus tres conocidas leyes la dinámica del movimiento planetario: los planetas describen órbitas elípticas alrededor del Sol, estando este en uno de sus focos, y de manera tal que cuanto más cerca se encuentren del mismo, mayor es la velocidad de su movimiento.

Pero estas leyes no se limitan al movimiento planetario, pueden extenderse a cualquier cuerpo en órbita alrededor de otro. Es decir, la trayectoria seguida por una nave en viaje interplanetario puede modelarse como una secuencia de diferentes órbitas elípticas en el seno del campo gravitatorio solar. Además, como consecuencia de la segunda ley, el movimiento de este cuerpo quedará limitado a un plano, conocido como **plano orbital**.

Por tanto, cuando en mecánica orbital se hace referencia a una **órbita kepleriana**, se define la trayectoria elíptica (u otra trayectoria cónica) que un cuerpo sigue con respecto a otro y que queda inscrita en un plano orbital.

A la hora de determinar dicha trayectoria únicamente se tiene en cuenta la atracción gravitacional que ejercen ambos cuerpos, despreciando los efectos gravitatorios de terceros cuerpos y fenómenos como la presión de radiación, la no uniformidad del cuerpo orbitado o los efectos de la relatividad general. Además, se considera un cuerpo central mucho mayor cuyo centro de masas se establece como centro de masas del sistema.

De forma general, las órbitas keplerianas pueden ser definidas a través de los conocidos como **Elementos Orbitales**. Este conjunto de seis elementos describen la órbita en el plano, su orientación en las tres dimensiones del espacio y definen la posición de un cuerpo en dicha órbita.

2.2. Elementos Orbitales

Los seis elementos que definen por completo una órbita son los siguientes: **momento angular específico h , inclinación i , longitud del nodo ascendente Ω , excentricidad e , argumento de perihelio ω y anomalía verdadera θ** .

De forma habitual, el momento angular h y la anomalía verdadera θ son sustituidos por otros dos elementos descriptivos como son el semieje mayor a y la anomalía media M .

Este conjunto de elementos puede ser obtenido a partir del conocido **vector de estado** del cuerpo en cuestión, el cuál queda definido, en el sistema de referencia geocéntrico ecuatorial, a partir de las componentes del **vector posición** y **velocidad** del cuerpo en la forma expresada:

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \quad (2.1a)$$

$$\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k} \quad (2.1b)$$

Si en un análisis preliminar de misión, el vector de estado inicial es asignado, esto es, los vectores posición y velocidad inicial ($\mathbf{r}_0, \mathbf{v}_0$) del cuerpo son conocidos, el conjunto de elementos que definen esta primera órbita pueden ser obtenidos.

A continuación se expondrá un algoritmo con el cual obtener los elementos orbitales a partir de un vector de estado dado. Sin embargo, previamente es necesario presentar dichos elementos, así como otros parámetros auxiliares que facilitarán la resolución del algoritmo.

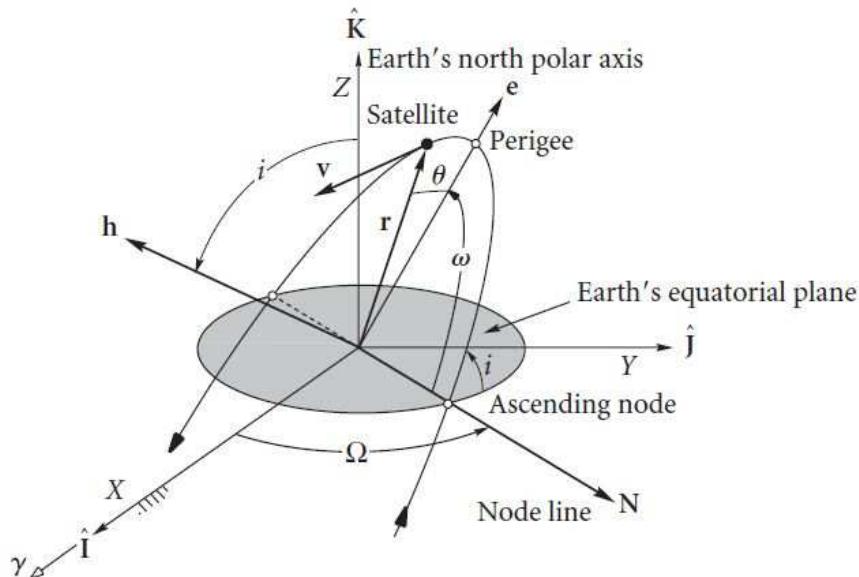


Figura 2.1: Elementos Orbitales. Figura tomada de (Curtis (2005)).

La **excentricidad** e y el **momento angular específico** h permiten definir una órbita en el plano. La excentricidad proporciona una idea de la forma de la órbita, de cuán distante se encuentra ésta de una circunferencia ($e = 0$), y el momento angular específico h , al ser definido como un vector en dirección siempre perpendicular a los vectores posición y velocidad, confina la órbita en un plano.

Una vez que la órbita es situada en su plano, y teniendo en cuenta el plano de referencia de la **eclíptica**, se obtiene la **Línea de nodos N**. Esta es la línea de intersección entre ambos planos y que contiene dos puntos de interés: el **nodo ascendente** Ω , punto en el que

el cuerpo cruza el plano de referencia moviéndose desde el hemisferio sur hacia el hemisferio norte, y el **nodo descendente** \bar{U} , donde ocurre lo opuesto.

Con la línea de nodos N fijada, es posible conocer la orientación espacial de la órbita a partir de los llamados **ángulos de Euler** (i, Ω, ω).

Atendiendo a la Figura 2.1, la **longitud del nodo ascendente** Ω es el ángulo formado entre la línea de nodos y el eje X positivo del plano de referencia. La **inclinación** i es el ángulo diedro entre el plano de referencia y el plano de la órbita medido de acuerdo a la regla de la mano derecha, o, de igual forma, el ángulo entre la normal al plano orbital (definido por el vector h) y el eje Z positivo del plano de referencia. El último de los ángulos es el **argumento de periastro** ω , medido en el plano de la órbita entre la línea de nodos y el vector excentricidad.¹

Por último, para localizar un punto concreto en la órbita es necesario conocer la **anomalía verdadera** θ . Sin embargo, ya se expuso que en numerosas ocasiones esta anomalía verdadera es sustituida por la ya mencionada **anomalía media** M .

Relacionada con estas dos anomalías, existe una tercera, conocida como **anomalía excéntrica** E .

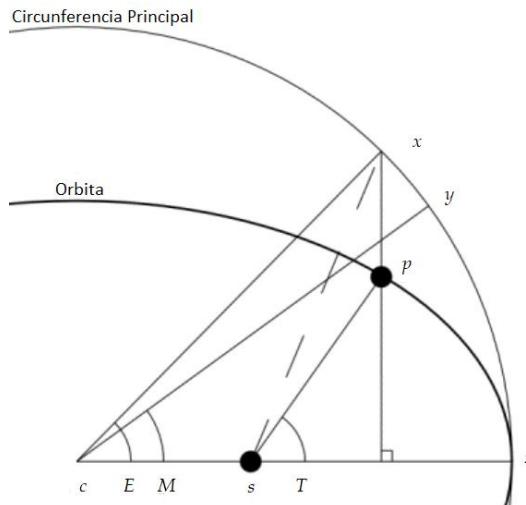


Figura 2.2: Definición de las anomalías. Figura tomada de (Anónimo (2010)).

Se define la **anomalía verdadera** θ de una órbita elíptica como el ángulo formado entre los vectores foco-satélite sP y foco-periapsis sz . Mediante la anomalía verdadera se puede conocer la posición de un cuerpo a lo largo de su órbita.

Por otro lado, para definir la anomalía media y la anomalía excéntrica se debe recurrir a la circunferencia principal de la elipse², pues la **anomalía excéntrica** E es definida como

¹El vector excentricidad pasa por el periapsis de la órbita, punto de la trayectoria en el que ambos cuerpos en interacción se encuentran más próximos.

²La circunferencia principal de una elipse es aquella cuyo centro es el centro de la elipse y su diámetro es

el ángulo medido desde el centro de la elipse entre la proyección del cuerpo orbitador sobre la circunferencia principal y la línea que une el origen de la elipse con su periapsis. De igual forma, la **anomalía media** M es el ángulo medido desde el centro de la elipse entre la proyección de un cuerpo ficticio con movimiento uniforme sobre la circunferencia principal y la línea que une el centro con el periapsis de la elipse.

La anomalía verdadera y la anomalía excéntrica se encuentran relacionadas a través de la expresión 2.2:

$$\tan \frac{\theta}{2} = \sqrt{\frac{1+e}{1-e}} \tan \frac{E}{2} \quad (2.2)$$

De igual manera, anomalía excéntrica y anomalía media se encuentran relacionadas a través de la conocida **Ecuación de Kepler**, que será desarrollada con posterioridad y que tiene la forma que sigue:

$$M = E - e \sin E \quad (2.3)$$

Definido todo el conjunto de parámetros anteriores, ya es posible desarrollar el algoritmo que conducirá a los elementos orbitales partiendo de un vector de estado conocido.

2.2.1. Elementos orbitales a partir de vector de estado conocido

Conocidas las componentes de los vectores posición y velocidad de un cuerpo en el espacio en un momento cualquiera del tiempo, es posible obtener los elementos orbitales descriptivos de su trayectoria a partir del siguiente algoritmo descrito en (Curtis (2005)).

1. Calcular la distancia a partir del vector posición dado.

$$r = \sqrt{\mathbf{r} \cdot \mathbf{r}} = \sqrt{x^2 + y^2 + z^2} \quad (2.4)$$

2. Calcular la velocidad a partir del vector velocidad dado.

$$v = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{v_x^2 + v_y^2 + v_z^2} \quad (2.5)$$

3. Calcular la velocidad radial.

$$v_r = \frac{\mathbf{r} \cdot \mathbf{v}}{r} = \frac{x \cdot v_x + y \cdot v_y + z \cdot v_z}{r} \quad (2.6)$$

La velocidad radial permite conocer si el cuerpo se encuentra en su vuelo hacia el periastro ($v_r < 0$) o desde el mismo ($v_r > 0$).

igual a dos veces el semieje mayor de la elipse en cuestión

4. Calcular el vector momento angular específico.

$$\mathbf{h} = \mathbf{r} \times \mathbf{v} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x & y & z \\ v_x & v_y & v_z \end{bmatrix} \quad (2.7)$$

5. Calcular el módulo del momento angular específico.

$$h = \sqrt{\mathbf{h} \cdot \mathbf{h}} \quad (2.8)$$

6. Calcular la inclinación.

$$i = \arccos \frac{v_z}{h} \quad (2.9)$$

i debe recaer entre $0^\circ < i < 180^\circ$. En el caso particular en que $90^\circ < i < 180^\circ$, la órbita es retrogada, es decir, el cuerpo girará en sentido horario.

7. Calcular el vector que define la línea de nodos.

$$\mathbf{N} = \mathbf{k} \cdot \mathbf{h} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & 1 \\ h_x & h_y & h_z \end{bmatrix} \quad (2.10)$$

8. Calcular la magnitud de la línea de nodos.

$$N = \sqrt{\mathbf{N} \cdot \mathbf{N}} \quad (2.11)$$

9. Calcular la longitud del nodo ascendente, RA (Ω).

$$\Omega = \arccos \frac{N_x}{N} \quad (2.12)$$

A la hora de situar Ω en el cuadrante correcto, surge ambigüedad que es resuelta teniendo en cuenta la siguiente relación:

$$\Omega = \begin{cases} \arccos \frac{N_x}{N} & (N_y \geq 0) \\ 360^\circ - \arccos \frac{N_x}{N} & (N_y \leq 0) \end{cases} \quad (2.13)$$

10. Calcular el vector excentricidad.

$$\mathbf{e} = \frac{1}{\mu} \left[(v^2 - \frac{\mu}{r}) \mathbf{r} - r \mathbf{v}_r \cdot \mathbf{v} \right] \quad (2.14)$$

11. Calcular la excentricidad.

$$e = \sqrt{\mathbf{e} \cdot \mathbf{e}} \quad (2.15)$$

12. Calcular el argumento de periastro ω .

$$\omega = \arccos \frac{\mathbf{N} \cdot \mathbf{e}}{Ne} \quad (2.16)$$

Igual que ocurría con Ω , a la hora de calcular ω surge ambigüedad de cuadrantes, que queda resuelta atendiendo a la siguiente condición:

$$\omega = \begin{cases} \arccos \frac{\mathbf{N} \cdot \mathbf{e}}{Ne} & (e_z \geq 0) \\ 360^\circ - \arccos \frac{\mathbf{N} \cdot \mathbf{e}}{Ne} & (e_z \leq 0) \end{cases} \quad (2.17)$$

13. Calcular la anomalía verdaera θ .

$$\theta = \arccos \frac{\mathbf{e} \cdot \mathbf{r}}{er} \quad (2.18)$$

Al calcular θ se vuelve a encontrar ambigüedad de cuadrantes solucionada según:

$$\theta = \begin{cases} \arccos \frac{\mathbf{e} \cdot \mathbf{r}}{er} & (v_r \geq 0) \\ 360^\circ - \arccos \frac{\mathbf{e} \cdot \mathbf{r}}{er} & (v_r \leq 0) \end{cases} \quad (2.19)$$

Siguiendo este algoritmo, si en un análisis preliminar se conoce el vector de estado descriptivo del cuerpo ($\mathbf{r}_0, \mathbf{v}_0$) en el instante inicial t_0 , el conjunto de elementos orbitales de este primer segmento de trayectoria puede ser obtenido directamente.

De forma similar, conocidos los elementos orbitales característicos de una trayectoria, el vector de estado descriptivo del cuerpo en un instante de tiempo t puede ser calculado.

La anomalía verdadera θ refleja el cambio de posición del cuerpo en su trayectoria, por lo que a través de esta, es posible obtener el vector de estado del cuerpo para cada instante t .

2.2.2. Vector de estado a partir de los elementos orbitales

El conjunto de seis elementos orbitales permite en cada instante t localizar el cuerpo a lo largo de la trayectoria que este sigue en el espacio. Sin embargo, es interesante para el diseño de una misión conocer también las componentes vectoriales que definen la posición y velocidad del cuerpo en cada momento. Es decir, conocer el vector de estado ($\mathbf{r}_i, \mathbf{v}_i$) que en un instante t_i describe la situación del cuerpo.

Si los elementos orbitales para un instante de tiempo t de interés son conocidos, es posible obtener el vector de estado descriptivo de ese instante mediante la secuencia de **rotación clásica de Euler** ($R_3(\gamma)R_1(\beta)R_2(\alpha)$).

En este caso, para llevar a cabo la transformación desde el sistema geocéntrico ecuatorial al sistema perifocal³ los ángulos a utilizar son los que definen la orientación de la órbita en

³Se expone la conversión entre el sistema geocéntrico ecuatorial y el sistema perifocal. Esta conversión puede extenderse a cualquier otro conjunto de sistemas de referencia.

el espacio (i, Ω, ω) , de forma que se obtiene la siguiente matriz de rotación:

$$[Q_{X\bar{x}}] = [\mathbf{R}_3(\omega)] [\mathbf{R}_1(i)] [\mathbf{R}_3(\Omega)] \quad (2.20a)$$

$$[Q_{X\bar{x}}] = \begin{bmatrix} -\sin \Omega \cos i \sin \omega + \cos \Omega \cos \omega & \cos \Omega \cos i \sin \omega + \sin \Omega \cos \omega & \sin i \sin \omega \\ -\sin \Omega \cos i \cos \omega - \cos \Omega \sin \omega & \cos \Omega \cos i \cos \omega - \sin \Omega \sin \omega & \sin i \cos \omega \\ \sin \Omega \sin i & -\cos \Omega \sin i & \cos i \end{bmatrix} \quad (2.20b)$$

Al tratarse de una **matriz ortogonal** la transformación inversa entre los sistemas vendrá dada por su matriz traspuesta ($[Q]_{\bar{x}X} = [Q]_{X\bar{x}}^T$).

Una vez la matriz de transformación ha sido obtenida, el siguiente algoritmo (Curtis (2005)) permite llegar al vector de estado característico del instante t .

1. Calcular el vector de posición en el sistema perifocal.

$$\mathbf{r}_{\bar{x}} = \frac{h^2}{\mu} \frac{1}{1 + e \cos \theta} \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} \quad (2.21)$$

2. Calcular el vector velocidad en el sistema perifocal.

$$\mathbf{v}_{\bar{x}} = \frac{\mu}{h} \begin{bmatrix} -\sin \theta \\ e + \cos \theta \\ 0 \end{bmatrix} \quad (2.22)$$

3. Calcular la matriz de transformación $[Q]_{X\bar{x}}$ y su traspuesta $[Q]_{\bar{x}X}^T$.

4. Transformar los vectores posición y velocidad de acuerdo a las expresiones siguientes:

$$\mathbf{r}_X = [Q]_{\bar{x}X} \mathbf{r}_{\bar{x}} \quad (2.23a)$$

$$\mathbf{v}_X = [Q]_{\bar{x}X} \mathbf{v}_{\bar{x}} \quad (2.23b)$$

Este algoritmo emplea la anomalía verdadera θ como parámetro indicador del cambio de posición. Es decir, θ es función del tiempo.

Sin embargo, no existe una expresión que los relacione directamente. Para poder obtener el valor de θ en cada momento t es necesario recurrir a la **ecuación de Kepler** (Ecuación 2.3), como se explicará en la siguiente sección.

2.3. Posición orbital en función del tiempo. Ecuación de Kepler

Cuando se recurre a la **ecuación de la órbita** (Ecuación 2.24), esta muestra la posición del cuerpo m en su movimiento alrededor del cuerpo M , pero no proporciona ninguna información

sobre la posición de m a lo largo del tiempo:

$$r = \frac{h^2}{\mu} \left(\frac{1}{1 + e \cos \theta} \right) \quad (2.24)$$

La **Ecuación 2.25**, sin embargo, relaciona la anomalía verdadera con el tiempo, de forma que si esta es sustituida en la **ecuación de la órbita** (Ecuación 2.24), se obtiene una relación temporal de θ (Ecuación 2.26).

$$\frac{d\theta}{dt} = \frac{h}{r^2} \quad (2.25)$$

$$\frac{\mu^2}{h^3} dt = \frac{d\theta}{(1 + e)^2} \quad (2.26)$$

Definiendo ahora t_p como el momento del tiempo en el que el cuerpo pasa por el periapsis de la órbita, y teniendo en cuenta que el origen del tiempo es arbitrario, en este punto, por tanto, es posible establecer su origen⁴. Así, en el periapsis de la órbita se tienen las siguientes dos condiciones:

- $t_p = 0$
- $\theta = 0$

Teniendo en cuenta ambas condiciones, es posible integrar ambos lados de la Ecuación 2.26:

$$\frac{\mu^2}{h^3} t = \int_0^\theta \frac{d\theta}{(1 + e \cos \theta)^2} \quad (2.27)$$

La integral de la parte derecha conduce a tres posibles casos:

$$\int \frac{dx}{(a + b \cos x)^2} = \frac{1}{(a^2 - b^2)^{\frac{3}{2}}} \left(2 \operatorname{atan} \sqrt{\frac{a - b}{a + b}} \tan \frac{x}{2} - \frac{b \sqrt{a^2 - b^2} \sin x}{a + b \cos x} \right) \quad (b < a) \quad (2.28)$$

$$\int \frac{dx}{(a + b \cos x)^2} = \frac{1}{a^2} \left(\frac{1}{2} \tan \frac{x}{2} + \frac{1}{6} \tan^3 \frac{x}{2} \right) \quad (b = a) \quad (2.29)$$

$$\int \frac{dx}{(a + b \cos x)^2} = \frac{1}{(b^2 - a^2)^{\frac{3}{2}}} \left[\frac{b \sqrt{b^2 - a^2} \sin x}{a + b \cos x} - a \ln \left(\frac{\sqrt{b + a} + \sqrt{b - a} \tan \frac{x}{2}}{\sqrt{b + a} - \sqrt{b - a} \tan \frac{x}{2}} \right) \right] \quad (b > a) \quad (2.30)$$

⁴Siempre es recomendable medir el tiempo a partir del paso por el periapsis de la órbita.

2.3.1. Trayectorias Elípticas

En primer lugar, dado que el modelo empleado describe las trayectorias orbitales en el seno de la heliosfera como secciones elípticas, el caso a desarrollar será el primero (Ecuación 2.28), donde las constantes a y b , adquieren los siguientes valores:

- $a = 1$
- $b = e$, siendo $0 < e < 1$

$$\frac{\mu^2}{h^3}t = \frac{1}{(1-e^2)^{\frac{3}{2}}} \left(2\text{atan} \sqrt{\frac{1-e}{1+e}} \tan \frac{\theta}{2} - \frac{e\sqrt{1-e^2}\sin\theta}{1+e\cos\theta} \right) \quad (2.31)$$

Despejando la expresión en paréntesis, se obtiene a la izquierda la expresión de la **anomalía media** de la elipse, M_e :

$$M_e = \frac{\mu^2}{h^3}(1-e^2)^{\frac{3}{2}}t \quad (2.32)$$

Si además se tiene en cuenta la ecuación del periodo T para órbitas elípticas:

$$\frac{\mu^2}{h^3}(1-e^2)^{\frac{3}{2}} = \frac{2\pi}{T} \quad (2.33)$$

La expresión para la anomalía media M_e puede ser aún más simplificada:

$$M_e = \frac{2\pi}{T}t \quad (2.34)$$

Como se explicó previamente, la anomalía media M_e es definida como el ángulo medido desde el centro de la elipse entre la línea de ápsides y la proyección de un cuerpo ficticio con **movimiento uniforme** sobre la circunferencia principal. Este movimiento uniforme viene definido por el parámetro n , conocido como **movimiento medio**.

Este parámetro representa el hecho de que, en toda órbita, en cada periodo T , el cuerpo recorre 2π radianes. Es decir, se puede definir una **velocidad angular media** según la siguiente expresión:

$$n = \frac{2\pi}{T} \quad (2.35)$$

que permite expresar M_e como:

$$M_e = nt \quad (2.36)$$

Si en este momento se retorna a la Ecuación 2.31, esta sigue siendo una expresión compleja que puede simplificarse introduciendo el concepto de **anomalía excéntrica** E . Teniendo en cuenta la relación entre anomalía verdadera y excéntrica definida según la Ecuación 2.2, despejando E y sustituyendo de nuevo en la Ecuación 2.31 se obtiene la conocida y previamente

mencionada **Ecuación de Kepler**:

$$M_e = E - e \sin(E) \quad (2.37)$$

Si la anomalía verdadera θ es conocida, es posible calcular la anomalía excéntrica E usando la Ecuación 2.2, y una vez E es obtenida, M_e puede ser calculada directamente a partir de la ecuación de Kepler. Si además, el periodo T de la órbita es conocido, a partir de la Ecuación 2.34 es fácil encontrar el tiempo de vuelo desde el periapsis.

Por el contrario, si el tiempo t es la variable conocida, M_e puede ser directamente obtenida y con ella calcular E operando con la ecuación de Kepler.

Sin embargo, este caso conduce a una **ecuación trascendental** que únicamente puede ser resuelta a través de métodos iterativos, como el **método de Newton** aquí empleado.

Para resolver la ecuación de Kepler mediante el método iterativo de Newton y obtener así la anomalía excéntrica E , se presenta el siguiente algoritmo (Curtis (2005)).

1. En primer lugar, se estima un valor incial de la raíz de E de acuerdo a:

$$E = \begin{cases} M_e + \frac{e}{2} & (M_e < \pi) \\ M_e - \frac{e}{2} & (M_e > \pi) \end{cases} \quad (2.38)$$

2. Una vez obtenido E_i en cualquier paso previo, obtener:

$$f(E_i) = E_i - e \sin E_i - M_e \quad (2.39a)$$

$$f'(E_i) = 1 - e \cos E_i \quad (2.39b)$$

3. Calcular:

$$ratio_i = \frac{f(E_i)}{f'(E_i)} \quad (2.40)$$

4. Si $|ratio_i|$ es mayor que la tolerancia elegida ($tolerance = 10^{-6}$), es necesario calcular un nuevo valor para E :

$$E_{i+1} = E_i - ratio_i \quad (2.41)$$

Con este nuevo valor de E , se vuelve a iterar desde el paso 2.

5. Cuando el $ratio_i$ sea menor que la tolerancia establecida, este valor de E será tomado como solución del algoritmo.

Una vez que la anomalía excéntrica E es calculada, mediante la Ecuación 2.2 se obtiene el valor de la anomalía verdadera θ , y a partir de ella y junto con los otros elementos orbitales es posible obtener el vector de estado del cuerpo para el momento de tiempo elegido de acuerdo al algoritmo desarrollado en la sección 2.2.2.

2.3.2. Trayectorias hiperbólicas

De forma general, la trayectoria de la nave en el interior de la esfera de influencia gravitacional no será tenida en cuenta, siendo modelada la maniobra como una rotación instantánea del vector velocidad. Sin embargo, si esto no es tenido en cuenta, la trayectoria de la nave en el interior de la esfera de influencia debe ser considerada. Esta trayectoria sería modelizada como una **curva hiperbólica**, de modo que para conocer la posición del cuerpo a lo largo de la misma habría que recurrir a la ecuación de Kepler aplicada a curvas hiperbólicas.

En este caso, la expresión de partida sería la Ecuación 2.30, que refleja el caso $b > a$, donde:

- $a = 1$
- $b = e$, siendo $e \geq 1$.

De modo que:

$$\frac{\mu^2}{h^3}t = \frac{1}{e^2 - 1} \frac{e \sin \theta}{(1 + e \cos \theta)} - \frac{1}{(e^2 - 1)^{\frac{3}{2}}} \ln \left(\frac{\sqrt{e+1} + \sqrt{e-1} \tan \frac{\theta}{2}}{\sqrt{e+1} - \sqrt{e-1} \tan \frac{\theta}{2}} \right) \quad (2.42)$$

Esta expresión puede ser simplificada tras un breve desarrollo matemático que permite definir en última instancia la **anomalía media** para la hipérbola M_h como:

$$M_h = \frac{\mu^2}{h^3} (e^2 - 1)^{\frac{3}{2}} t \quad (2.43)$$

De igual forma que para la elipse, la expresión puede ser simplificada si introducimos un ángulo análogo a la anomalía excéntrica E : la **variable adimensional F** .

$$\sinh F = \frac{y}{b} \quad (2.44)$$

Donde las variables y, b son definidas de acuerdo a la Figura 2.3.

Teniendo en cuenta este nuevo parámetro, y tras una serie de desarrollos matemáticos, finalmente se llega a la ecuación de Kepler para la hipérbola:

$$M_h = e \sinh F - F \quad (2.45)$$

El procedimiento de resolución iterativo mediante el método de Newton sigue la misma estructura que para el caso elíptico, siendo la única diferencia la estimación inicial de F . Para análisis computacionales, es una buena estimación inicial la siguiente asunción:

$$F_0 = M_h \quad (2.46)$$

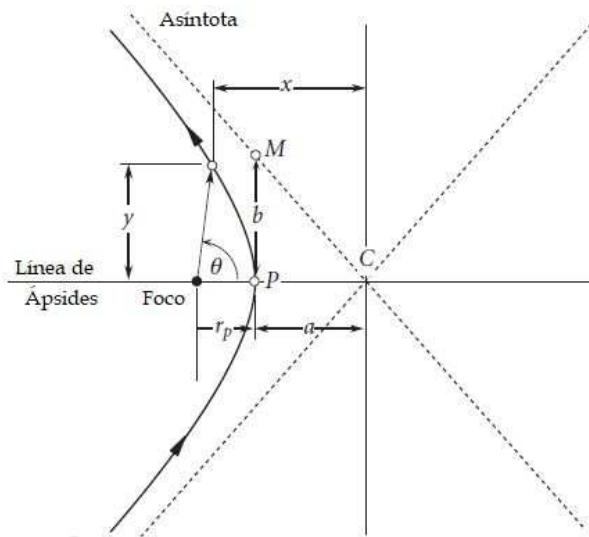


Figura 2.3: Parámetros de la hipérbola. Figura adaptada de (Curtis (2005)).

La ecuación de Kepler, aplicada tanto al caso elíptico como hiperbólico, junto con los elementos orbitales definidos y los algoritmos expuestos para calcular el vector de estado correspondiente (y viceversa) conforman un marco de herramientas y nociones teóricas que constituyen la base del movimiento orbital kepleriano, y que han permitido definir un contexto básico previo al análisis en profundidad de la **maniobra de asistencia gravitacional** (o **fly-by**) que será descrita en el próximo capítulo.

3. Maniobra de Asistencia Gravitacional

Las **maniobras de asistencia gravitacional** son de gran utilidad al permitir modificar los parámetros característicos de la trayectoria orbital de una nave, tanto aquellos que definen su forma como aquellos que definen su orientación en el espacio, obteniendo un nuevo segmento de órbita con características diferentes al anterior. De esta forma se logran alcanzar diferentes objetivos reduciendo costes en términos de necesidades energéticas.

En cualquier caso, los parámetros descriptivos de la órbita post-asistencia pueden ser identificados atendiendo al modelo planteado para el análisis.

3.1. Modelo del fly-by

El modelo empleado para el diseño de un misión interplanetaria en este trabajo asienta su base en las aproximaciones siguientes:

- El cuerpo en órbita es modelado como un cuerpo **sin masa**. Esto es, el cuerpo se encuentra sujeto a la acción gravitatoria de otros cuerpos, pero este no tiene campo gravitatorio propio, no genera atracción sobre ellos.
- El campo gravitatorio del Sistema Solar queda modelado como el campo gravitatorio del Sol y la influencia gravitacional de cada planeta queda limitada a su **esfera de influencia gravitatoria**.
- La trayectoria es representada por una serie de segmentos de **movimiento Kepleriano no perturbado** tanto en gravisfera solar como en la gravisfera del planeta.

Modelizar el campo gravitatorio del Sistema Solar de acuerdo a lo anterior, implica que la **maniobra de asistencia gravitacional** o **fly-by** generará una serie de segmentos de trayectoria los cuales tendrán que ser ajustados en el punto en que la nave cruza la gravisfera del cuerpo. Para ajustar estos segmentos de trayectoria se recurrirá al modelo de **ajuste de cónicas**, que será detallado en el Capítulo 4.

Sin embargo, el modelo previo puede ser incluso más simplificado si la esfera de influencia del cuerpo atractivo es considerada en relación con las características de su órbita y de su tamaño:

- La esfera de influencia de un cuerpo atractivo en relación con el radio de la órbita del mismo puede considerarse **infinitesimal**.
- La esfera de influencia de un cuerpo atractivo en relación con el tamaño del propio cuerpo puede considerarse **infinita**.

Bajo las anteriores hipótesis, la esfera de influencia gravitacional de un cuerpo atractivo en el marco de un viaje interplanetario puede asumirse de **radio nulo**, de modo que el resultado del fly-by queda reducido a una **rotación instantánea del vector velocidad** de la nave.

Según esto, las condiciones posteriores al fly-by pueden ser determinadas siguiendo una estructura básica de rotación mediante una matriz de transformación:

$$(\mathbf{x})_{\text{after}} = R(\varphi) (\mathbf{x})_{\text{before}} \quad (3.1)$$

Antes de entrar en profundidad con este modelo de asistencia, en primer lugar se estudiará una aproximación más compleja eliminando la última hipótesis y asumiendo que el cuerpo en el interior de la gravisfera se mueve siguiendo una trayectoria hiperbólica.

3.2. Órbitas hiperbólicas en el seno de la gravisfera

Al considerar el radio de la esfera de influencia, la trayectoria seguida por el cuerpo en su interior deberá ser modelada. Según el modelo de **ajuste de cónicas**, la trayectoria fuera de la gravisfera del planeta es modelada como una sucesión de segmentos elípticos, mientras que en el interior será aproximada por una curva hipérbolica.

Según esta aproximación, la maniobra de asistencia gravitacional se traduce en una **rotación tanto del vector velocidad** del cuerpo como de su **vector posición**, definidos, en coordenadas planetocéntricas, como:

$$\mathbf{r}_1 = \mathbf{R}_1 - \mathbf{R}_{pl} \quad (3.2a)$$

$$\mathbf{v}_1 = \mathbf{V}_1 - \mathbf{V}_{pl} \quad (3.2b)$$

Donde $(\mathbf{R}_1, \mathbf{V}_1)$ representan la posición y velocidad de la nave en el sistema de referencia heliocéntrico, y $(\mathbf{R}_{pl}, \mathbf{V}_{pl})$ la posición y velocidad del cuerpo atractivo en el momento t en que la maniobra comienza.

Conociendo estos vectores en el momento inicial, es posible calcular las condiciones post-maniobra según la siguiente transformación:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \Omega(\varphi^*) \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad (3.3a)$$

$$\begin{pmatrix} v_{x2} \\ v_{y2} \\ v_{z2} \end{pmatrix} = \Omega(\varphi) \begin{pmatrix} v_{x1} \\ v_{y1} \\ v_{z1} \end{pmatrix} \quad (3.3b)$$

Siendo (x_1, y_1, z_1) y (v_{x1}, v_{y1}, v_{z1}) las componentes de los vectores posición y velocidad del cuerpo a la entrada de la esfera; (x_2, y_2, z_2) y (v_{x2}, v_{y2}, v_{z2}) las componentes de los vectores posición y velocidad del mismo después del fly-by, y siendo Ω la matriz de transformación.

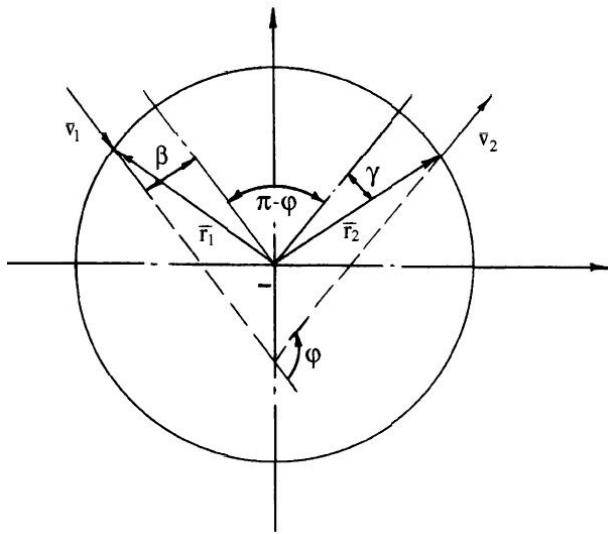


Figura 3.1: Rotación del vector velocidad durante la maniobra de fly-by. Figura tomada de (Lambunsky y cols. (1998)).

Para calcular el estado final de la maniobra de acuerdo a las ecuaciones 3.3a y 3.3b hay que recurrir a la matriz de transformación Ω definida en la ecuación 3.4.

$$\Omega = \begin{bmatrix} \cos \varphi & \frac{c_3}{c} \sin \varphi & -\frac{c_2}{c} \sin \varphi \\ -\frac{c_3}{c} \sin \varphi & \cos \varphi & \frac{c_1}{c} \sin \varphi \\ \frac{c_2}{c} \sin \varphi & -\frac{c_1}{c} \sin \varphi & \cos \varphi \end{bmatrix} \quad (3.4)$$

Esta matriz es función de las componentes del **vector de áreas c** y el **ángulo de rotación**:

- φ : definido como el ángulo de rotación del vector velocidad del cuerpo.

$$\varphi = 2 \arctan \frac{K_{pl}}{\beta v_\infty^2} \quad (3.5)$$

- φ^* : definido como el ángulo de rotación del vector posición del cuerpo.

$$\varphi^* = \pi + \varphi - 2\gamma = 2 \arcsin \frac{\beta}{R_{sphpl}} \quad (3.6)$$

Donde:

- K_{pl} es el **parámetro gravitacional** del cuerpo sobre el que se lleva a cabo la maniobra.
- β es la distancia desde el centro del cuerpo atractivo a la que quiere realizarse la

maniobra. Queda definida por la siguiente ecuación:

$$\beta = \frac{\mathbf{r}_1 \mathbf{v}_1}{v_1} \quad (3.7)$$

Este parámetro debe ser analizado con detenimiento, ya que es necesario establecer un límite para evitar fenómenos de frenado atmosférico, e incluso re-entradas. Este límite será, por tanto, función de la distancia mínima admisible al planeta, r_{0min} :

$$r_{0min} = r_{pl} + h_{atm} \quad (3.8)$$

Donde el primer término representa el **radio planetario** y el segundo la **altura de la atmósfera** de dicho cuerpo. Con este valor, la condición límite queda fijada:

$$\beta \leq \beta_{min} = r_{0min} \sqrt{1 + \frac{2K_{pl}}{r_{0min} v_\infty^2}} \quad (3.9)$$

- v_∞ representa el **exceso de velocidad hiperbólica** de la nave.

$$v_\infty = \left(v_1^2 - \frac{2K_{pl}}{r_1} \right)^{1/2} \quad (3.10)$$

v_∞ es la magnitud de $\mathbf{v}_{\infty 1}$ y $\mathbf{v}_{\infty 2}$, que se corresponden respectivamente con el exceso de velocidad hiperbólica inicial y final de la nave, estando ambos vectores dirigidos según las asíntotas de la hipérbola formada durante el fly-by.

$$|\mathbf{v}_{\infty 1}| = |\mathbf{v}_{\infty 2}| \quad (3.11)$$

Dado que la trayectoria en el interior de la esfera gravitacional durante el fly-by es tenida en cuenta, el tiempo transcurrido desde la entrada a la gravisfera en el instante t_1 hasta que se abandona la misma en el instante t_2 debe ser por tanto considerado. De este modo, se podrá localizar la nave a lo largo de su trayectoria hiperbólica.

Conociendo ambos instantes de tiempo, es posible calcular los elementos que definirán el nuevo segmento de trayectoria en el campo gravitatorio del Sol según:

$$\mathbf{R}_2 = \mathbf{R}(T_1 + \Delta t) + \mathbf{r}_1 \Omega(\varphi^*) \quad (3.12a)$$

$$\mathbf{V}_2 = \mathbf{V}(T_1 + \Delta t) + \mathbf{v}_1 \Omega(\varphi) \quad (3.12b)$$

Donde los vectores **R** y **V** son calculados de acuerdo a las efemérides planetarias.

El modelo descrito hasta este punto incurre en un algoritmo complejo de cálculo numérico que puede simplificarse asumiendo una esfera de influencia de radio nulo.

3.3. Fly-by. Análisis de la maniobra

El modelo aquí empleado asume que las dimensiones de la gravisfera pueden despreciarse al considerar las hipótesis ya expuestas que relacionaban el tamaño de esta con las propiedades de la órbita del cuerpo y su tamaño.

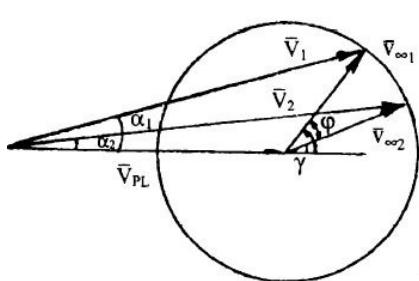
Esta asunción tiene un primer impacto en la duración de la propia maniobra, que pasa a ser considerada instantánea:

$$\Delta t = T_2 - T_1 = 0 \quad (3.13)$$

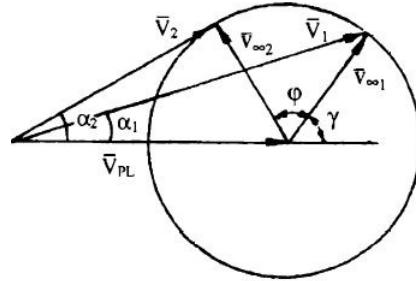
Así, el punto de entrada y el punto de salida de la gravisfera pueden ser considerados el mismo. Esta posición, como se verá más adelante, puede ser aproximada por el vector posición del cuerpo atractivo para ese instante.

En definitiva, el resultado de la maniobra de asistencia gravitacional se traduce en una **rotación instantánea del vector velocidad de la nave**. Este nuevo vector velocidad permitirá definir las características postfly-by de la nueva órbita en el seno heliocéntrico.

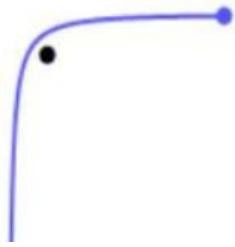
Este cambio $\Delta\mathbf{V}$ en el vector velocidad de la nave puede ser tanto positivo como negativo, es decir, el resultado puede inducir una **aceleración** o una **deceleración** de su movimiento (**Figura 3.2**).



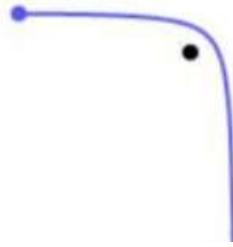
(a) Deceleración gravitacional



(b) Aceleración gravitacional



(c) Esquema maniobra de deceleración



(d) Esquema maniobra de aceleración

Figura 3.2: Posibles maniobras de asistencia gravitacional. Figuras (a) y (b) tomadas de (Labunsky y cols. (1998)).

- El caso 3.2a muestra el diagrama de una maniobra de **deceleración gravitacional**, en la que la **nave** alcanza el punto de intersección de las órbitas **antes** que el **cuerpo atractivo**.

$$|\mathbf{V}_2| < |\mathbf{V}_1| \quad (3.14)$$

- El caso 3.2b muestra el diagrama de una maniobra de **aceleración gravitacional**, en la que es el **cuerpo atractivo** el que alcanza el punto de intersección de las órbitas **antes** que la **nave**.

$$|\mathbf{V}_2| > |\mathbf{V}_1| \quad (3.15)$$

Los vectores que aparecen en la Figura 3.2 tienen el siguiente significado:

- \mathbf{V}_1 es el vector velocidad de la nave en su camino hacia el cuerpo.
- \mathbf{V}_2 es el vector velocidad de la nave a la salida de la maniobra de asistencia gravitacional.
- $\mathbf{v}_{\infty 1}$ representa el exceso de velocidad hiperbólica inicial de la nave.
- $\mathbf{v}_{\infty 2}$ representa el exceso de velocidad hiperbólica final de la nave.

Atendiendo a los ángulos definidos en la Figura 3.2, puede apreciarse que en la maniobra de **deceleración** los ángulos γ y φ se suman, haciendo que $\mathbf{v}_{\infty 1}$ rote en **sentido antihorario**, mientras que, en el caso de una **aceleración**, estos ángulos se restan de forma que $\mathbf{v}_{\infty 1}$ rota en **sentido horario**.

La relación de estos ángulos determinará el valor máximo o mínimo del vector velocidad a la salida de la maniobra de asistencia:

- \mathbf{V}_2 alcanzará su mínimo cuando $\varphi + \gamma = \pi$.
- \mathbf{V}_2 alcanzará su máximo cuando $\varphi - \gamma = 0$

Cabe destacar que existen una serie de casos especiales cuando el vector velocidad de la nave es **colineal** (o **casi colineal**) con el vector velocidad del cuerpo. Este caso es el mostrado en la Figura 3.3, según el cual el vector \mathbf{V}_2 tendrá el mismo valor sea cual sea el sentido de rotación del vector $\mathbf{v}_{\infty 1}$.

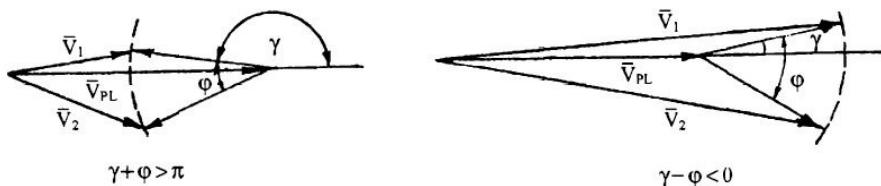


Figura 3.3: Maniobra de asistencia. Vectores coplanarios o casi-coplanarios. Figura tomada de (Lambunsky y cols. (1998)).

En este caso, la maniobra de aceleración sólo tendrá lugar cuando el vector \mathbf{V}_1 de la nave sea menor que el vector \mathbf{V}_{pl} ; de igual manera, la maniobra de deceleración sólo tendrá lugar cuando el vector \mathbf{V}_1 de la nave sea mayor que el vector \mathbf{V}_{pl} .

De acuerdo a todo lo expuesto, se puede concluir que el efecto de este tipo de maniobra se ve reflejado no sólo en la rotación del vector velocidad de la nave, sino también en su módulo. Este cambio viene definido por el parámetro ΔV , que si es modificado, por ejemplo, introduciendo un impulso artificial adicional permitirá obtener una órbita de salida u otra.

$$\Delta V = |\mathbf{V}_2| - |\mathbf{V}_1| \quad (3.16)$$

El valor ΔV es función de una serie de parámetros sobre los que puede tenerse decisión, entre ellos la **altura** a la que quiere realizarse el fly-by. Dado que el **ángulo de rotación** φ es función de la altura, modificando este parámetro es posible modificar el ΔV resultante.

El ángulo de rotación φ puede definirse atendiendo a las Ecuaciones 3.5 y 3.9, tal que:

$$\varphi = 2 \arcsin \frac{K_{pl}}{K_{pl} + r_{0min} v_{\infty 1}^2} \quad (3.17)$$

Y según la expresión previa, se puede definir:

$$\Delta V = 2v_{\infty 1} \sin \frac{\varphi}{2} = \frac{2v_{\infty} K}{K + r_{\pi} v_{\infty 1}^2} \quad (3.18)$$

Siendo K el parámetro gravitacional del cuerpo, y r_{π} el radio pericentral de la hipérbola. Este último muestra la dependencia de la altura a la que el fly-by tiene lugar.

Según la Ecuación 3.18, el máximo valor será alcanzado cuando r_{π} tome el valor del radio planetario (r_{pl}), caso que lógicamente no puede ser alcanzado. Si se impone la condición:

$$\frac{\partial V}{\partial v_{\infty 1}} = 0|_{r_{\pi} r_{pl}} \quad (3.19)$$

Se encuentra que ΔV_{max} se obtiene cuando:

$$v_{\infty} = \left(\frac{K}{r_{pl}} \right)^{\frac{1}{2}} \quad (3.20)$$

Este valor para v_{∞} coincide con la **velocidad de rotación local en la superficie** del cuerpo sobrevolado.

Una vez expuestos los efectos del fly-by sobre las componentes del vector de estado de la nave en órbita, a continuación se analizará qué efectos tiene esta maniobra sobre los elementos descriptivos de la órbita.

3.4. Efecto sobre los elementos orbitales

Ya se ha visto que el conjunto de seis elementos orbitales permiten parametrizar por completo la órbita descrita por el cuerpo. Si las condiciones de la nave en el momento en el que entra en la esfera de influencia (vector de estado, ángulos de entrada...) y otros parámetros característicos del fly-by son conocidos o previamente establecidos, como pueden ser la altura a la se realizará o el ΔV que se quiere obtener, el conjunto de elementos orbitales que definirán la post-órbita pueden ser predichos.

Para resolver este problema, es necesario recurrir al siguiente planteamiento:

Se consideran dos cuerpos de masas m_1 y m_2 ($m_1 \gg m_2$) moviéndose en órbitas coplanarias en el seno del campo gravitatorio de un cuerpo mayor cuya constante gravitacional es K . El cuerpo de masa m_1 tiene además una constante gravitacional K_1 .

Antes de continuar con el problema, en primer lugar será necesario hacer un pequeño inciso en el análisis energético de la maniobra.

3.4.1. La incongruencia energética

De acuerdo al **principio de conservación de la energía**, la nave al aproximarse al cuerpo debería acelerar su movimiento (aumentando su energía cinética), pero tras su paso, y a medida que se aleja de él, debería perder exactamente esa misma cantidad de energía. Sin embargo, tras la maniobra la nave experimenta un cambio en el módulo de su velocidad, violando así el principio de conservación.

Esta incongruencia es resuelta analizando la situación, primero en un **sistema de referencia centrado en m_1** , y después en otro sistema **centrado en el cuerpo de constante K** (en este caso, el Sol).

En el primer caso, el cuerpo m_1 es considerado estacionario, es decir, su vector velocidad no es tenido en cuenta. En esta situación, la nave se acelerará a media que se acerque a m_1 : ganará **energía cinética** en su aproximación, perdiendo **energía potencial**. De igual manera, al alejarse de m_1 ocurrirá exactamente lo contrario. En consecuencia, la energía involucrada en la maniobra permanece constante.

Sin embargo, si el cuerpo central es el Sol, la **componente vectorial de la velocidad del cuerpo m_1** ha de ser considerada. En este caso, al finalizar el fly-by la nave verá sumada (o restada) la componente del vector velocidad del cuerpo m_1 sobrevolado, **aumentando (o disminuyendo)** su velocidad en el sistema de referencia heliocéntrico.

Energéticamente, al interactuar la nave y el cuerpo m_1 en el sistema de referencia heliocéntrico, se produce una **transferencia de energía y momento cinético** entre ambos: de acuerdo a la **tercera ley de Newton**, la nave se verá acelerada, y el cuerpo m_1 se verá en consecuencia decelerado. Sin embargo, dado que la masa del cuerpo es potencialmente mayor a la de la nave, el efecto de esta maniobra sobre el cuerpo es apenas perceptible.

La nave **Voyager**, en su paso por **Júpiter** en 1979 experimentó un importante impulso al aumentar su velocidad alrededor de 10km/s. Como consecuencia, Júpiter sufrió una deceleración de unos 10^{-24} km/s, un valor indiferente para un cuerpo de tal tamaño.

3.5. Efecto sobre los elementos orbitales. Continuación

Retomando el problema expuesto anteriormente, y atendiendo a la Figura 3.4, se definen:

- \mathbf{R}, \mathbf{V} para el cuerpo de masa m_1
- \mathbf{R}, \mathbf{V}_1 para el cuerpo de masa m_2 , que representará la nave.

Después de la maniobra, la nave verá su velocidad modificada, siendo su vector velocidad \mathbf{V}_2 .

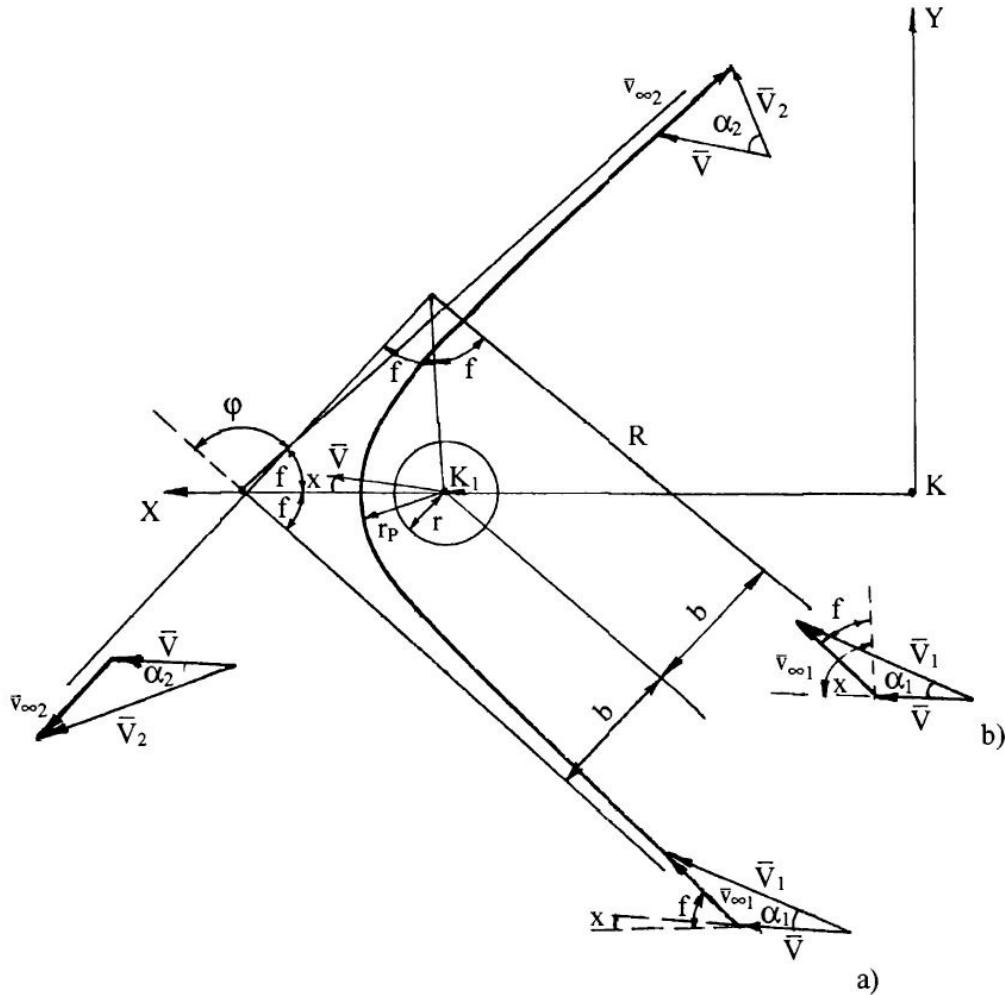


Figura 3.4: Análisis vectorial del fly-by. Figura tomada de (Labunsky y cols. (1998)).

Según la Figura 3.4, los **vectores velocidad** de la nave pueden ser definidos en función de χ y f :

- χ : ángulo formado entre el eje de la hipérbola y la dirección del vector \mathbf{V} .
- f : ángulo formado entre la dirección de la asíntota hiperbólica y el eje de la propia hipérbola.

$$V_{1x} = v_\infty \cos f + V \cos \chi \quad (3.21a)$$

$$V_{1y} = v_\infty \sin f + V \sin \chi \quad (3.21b)$$

$$V_{2x} = v_\infty \cos(\pi - f) + V \cos \chi \quad (3.21c)$$

$$V_{2y} = v_\infty \sin(\pi - f) + V \sin \chi \quad (3.21d)$$

El ángulo γ es definido como aquel formado por los vectores \mathbf{V} y \mathbf{v}_∞ . La relación entre este ángulo y los dos previamente mencionados describe los dos tipos de maniobra expuestos:

- $\chi = f - \gamma$. La nave pasa por delante del cuerpo, con la consiguiente **deceleración** de la nave.
- $\chi = f + \gamma$. La nave pasa después que el cuerpo, con la consiguiente **aceleración** de la nave.

Mediante los vectores velocidad \mathbf{V}_1 y \mathbf{V}_2 es posible definir el **cambio de energía** que experimenta la nave:

$$\Delta h = V_2^2 - V_1^2 = -4v_\infty V \cos f \cos \chi \quad (3.22)$$

ó

$$\Delta h = -4v_\infty V \cos f \cos(f \pm \gamma) \quad (3.23)$$

Resulta que γ puede ser encontrado recurriendo al **triángulo de velocidades** que queda definido en el diagrama. En este se encuentra también el ángulo α , que describe el **ángulo de entrada** de la nave a la gravisfera y que queda definido entre los vectores \mathbf{V} y \mathbf{V}_1 .

Si además se tiene en cuenta que el ángulo f se relaciona con el ángulo de rotación φ , se puede llevar a cabo el siguiente desarollo:

$$f = \frac{\pi - \varphi}{2} \quad (3.24)$$

$$\sin \gamma = \frac{V_1 \sin \alpha_1}{v_\infty} \quad (3.25a)$$

$$\cos \gamma = \frac{V_1 \cos \alpha_1 - V}{v_\infty} \quad (3.25b)$$

$$\cos f = \sin \frac{\varphi}{2} = \frac{1}{1 + r_p \frac{v_\infty^2}{K_1}} = (1 + mn^2)^{-1} \quad (3.26a)$$

$$\sin f = n(m^2n^2 + 2m)^{\frac{1}{2}} \cos f \quad (3.26b)$$

$$v_\infty = (V_1^2 + V^2 - 2V_1 V \cos \alpha_1)^{\frac{1}{2}} \quad (3.27)$$

Donde m es la distancia pericéntrica relativa,

$$m = \frac{r_p}{r} \quad (3.28)$$

y n define el exceso de velocidad hipérbolica relativa,

$$n = \frac{v_\infty}{v_{cr}} \quad (3.29)$$

tomando v_{cr} el valor de la velocidad circular en la superficie del cuerpo sobrevolado.

$$v_{cr} = \left(\frac{K_1}{r} \right)^{\frac{1}{2}} \quad (3.30)$$

Considerando lo anterior, la Ecuación 3.23 puede ser expresada en función de h_1 y α_1 .

$$h_2 = h_1 - 4V(1 + mn^2)^{-2} \left[V_1 \cos \alpha_1 - V_1 \sin \alpha_1 (2mn^2 + m^2n^4)^{\frac{1}{2}} \right] \quad (3.31)$$

La definición de la nueva órbita tras el fly-by conlleva un cambio en la **integral de área**:

$$\Delta C = R(V_1 \cos \alpha_1 - V_2 \cos \alpha_2) \quad (3.32)$$

Introduciendo en la Ecuación 3.32 el valor del coseno para los ángulos de entrada y salida, se tiene:

$$\Delta C = R \frac{V_1^2 - V_2^2}{2V} = -\frac{R \Delta h}{2V} = 2Rv_\infty \cos f \cos \chi \quad (3.33)$$

Ahora con las Ecuaciones 3.31 y 3.33, se puede obtener la **constante de áreas** para cualquier valor de V_1 y α_1 .

$$C_2 = C_1 + 2R(1 + mn^2)^{-2} \left[V_1 \cos \alpha_1 - V_1 \sin \alpha_1 (2mn + m^2n^4)^{\frac{1}{2}} \right] \quad (3.34)$$

Por último, es posible calcular el cambio que experimenta la **línea de ápsides** si se recurre a la integral vectorial de Laplace, simplificándola además teniendo en cuenta que la maniobra

se asumen instantánea, por lo que $\mathbf{R}_1 = \mathbf{R}_2$:

$$\Delta\lambda = \mathbf{C}_1 \times \mathbf{V}_1 - \mathbf{C}_2 \times \mathbf{V}_2 \quad (3.35)$$

El ángulo de rotación de la línea de ápsides δ coincide con el ángulo de rotación del vector velocidad de la nave, y puede ser calculado atendiendo a la siguiente relación:

$$\cos \delta = \frac{(V_1^2 + V_2^2)(1 + mn^2) - 2v_\infty}{2V_1V_2} \quad (3.36)$$

Con los valores h_2 , C_2 , λ_2 es posible obtener con posterioridad los elementos Keplerianos correspondientes a la nueva órbita que definirá el segmento de trayectoria a seguir en el seno de la heliosfera.

4. Modelo de Ajuste de Cónicas

Un cuerpo en trayectoria interplanetaria en el Sistema Solar experimenta las fuerzas gravitatoria no sólo del Sol, sino también del resto de cuerpos. Esta situación puede ser modelada a través de complejos métodos numéricos de alta precisión, pero, el **método de ajuste de cónicas**, más simplificado, permite una modelización más sencilla y acorde a la precisión requerida en un análisis preliminar.

Este método emplea las **esferas de influencia gravitacional** de los planetas implicados en la transferencia para reducir un problema de n cuerpos en un problema de $n-1$ cuerpos. Es decir, divide las trayectorias de la nave en trayectorias planetocéntricas en torno al planeta llegada o partida cuando la nave se encuentra en la gravisfera planetaria, y una trayectoria heliocéntrica en torno al Sol cuando la nave se encuentra fuera de la misma. En esta situación, la transferencia heliocéntrica entre los dos planetas puede ser definida recurriendo al **problema de Lambert**, válido únicamente para el problema de dos cuerpos.

A continuación, en la sección 4.1 se define el concepto de esfera de influencia, que será empleado para desarrollar en más detalle el método de ajuste de cónicas en la sección 4.2. Con las trayectorias ya expuestas, en la sección 4.3 se desarrolla el algoritmo que definirá la trayectoria heliocéntrica de la nave. Por último, la sección 4.4 expone un propagador planetario que permitirá definir la posición de los planetas que representarán a los puntos P_1 y P_2 del problema de Lambert.

4.1. Esfera de influencia gravitacional

En el Sistema Solar, la fuerza gravitatoria ejercida por el Sol es la más intensa de todas, sintiendo el resto de cuerpos esta influencia, y pudiendo asumirse que es la única que experimentan en su camino en el espacio. Sin embargo, una nave en misión interplanetaria puede experimentar de forma más acusada la gravedad de un cuerpo celeste si esta entra en su zona de influencia. Esto puede considerarse así puesto que la **Ley de la Gravedad** establece que la fuerza gravitatoria **disminuye drásticamente con la distancia** al centro de atracción.

Esta zona es conocida como **esfera de influencia gravitacional**, y en su interior puede suponerse que sobre el vehículo únicamente actúa la **fuerza gravitatoria del cuerpo en cuestión**, despreciando el efecto causado por la fuerza gravitatoria del Sol.

La esfera de influencia de cualquier cuerpo celeste es simplemente una **estimación de la distancia** a partir de la cuál la atracción gravitatoria del planeta es una simple perturbación, **dominando la atracción gravitatoria del Sol**. Es decir, estima la frontera a partir de la cuál el problema queda reducido a un problema de dos cuerpos entre el Sol y la nave.

Dicho de otro modo, la esfera de influencia es útil para reducir un problema de n cuerpos, como el que se encuentra en una trayectoria interplanetaria, en una situación de $n - 1$ problemas de dos cuerpos.

Para estimar el valor de la zona de influencia de un planeta o un cuerpo celeste se puede recurrir a diferentes modelos de aproximación. En el caso que aquí se expone, el modelo empleado es el de las **esferas de influencia gravitacional** o gravisferas de Laplace. Existen otros métodos como las esferas de Hill o las esferas de mínima desviación.

Según este modelo, si un vehículo está sometido a la influencia de dos cuerpos masivos m_1 y m_2 , se pueden encontrar dos situaciones:

- La nave se encuentra próxima al cuerpo m_1 de forma que la influencia del cuerpo m_2 se considera como una perturbación:

$$\mathbf{g} = \mathbf{g}_1 + \mathbf{g}'_2 = \frac{\mu_1}{r_1^2} + \mu_2 \left| \frac{\mathbf{r}_2}{r_2^3} - \frac{\mathbf{r}_{12}}{r_{12}^3} \right| \quad (4.1)$$

- La nave se encuentra próxima al cuerpo m_2 de forma que la influencia del cuerpo m_1 se considera una perturbación:

$$\mathbf{g} = \mathbf{g}_2 + \mathbf{g}'_1 = \frac{\mu_2}{r_2^2} + \mu_1 \left| \frac{\mathbf{r}_1}{r_1^3} - \frac{\mathbf{r}_{12}}{r_{12}^3} \right| \quad (4.2)$$

Donde \mathbf{r}_{12} representa la distancia que separa los cuerpos m_1 y m_2 .

La frontera que separa ambas zonas de influencia gravitacional se encuentra cuando se cumple la condición siguiente:

$$\frac{g'_2}{g_1} = \frac{g'_1}{g_2} \quad (4.3)$$

Además, si uno de los dos cuerpos es mucho más masivo que el otro, por ejemplo, $m_1 \gg m_2$, esta zona de separación es aproximadamente una esfera que envuelve al cuerpo menos masivo (m_2 en este caso).

Una vez establecida la frontera y su forma, el radio de la esfera gravitacional de un cuerpo, según este modelo, puede ser aproximado de acuerdo a la siguiente expresión:

$$r_{SOI} = a_p \left(\frac{m_p}{M} \right)^{\frac{2}{5}} \quad (4.4)$$

Donde a_p es el semi-eje mayor del planeta atractivo, m_p es la masa del planeta atractivo y M es la masa del Sol.

4.2. Modelo de ajuste de cónicas

De forma general, el **método de ajuste de cónicas** emplea la esfera de influencia y las hipótesis que se realizaron al principio sobre esta para **reducir un problema de n cuerpos** transformándolo en **$n - 1$ problemas de dos cuerpos**.

Este método asume que la **esfera de influencia** de un planeta tiene un **radio infinito** si esta es observada desde dicho planeta, pero tiene **radio cero** si se observa desde el Sol. De esta forma se pueden considerar diferentes problemas de dos cuerpos:

- La trayectoria del vehículo **fuerza de la esfera de influencia** de un cuerpo celeste se considera un problema de dos cuerpos con el **Sol como foco atractivo principal**, y asume los planetas y su gravisfera como puntos que coinciden con el centro planetario.
- La trayectoria del vehículo en el **interior de la esfera de influencia** de un cuerpo celeste se considera un problema de dos cuerpos con el **cuerpo celeste como foco atractivo principal**.

Además, en ambas situaciones, el cuerpo describe una **órbita Kepleriana no perturbada**, es decir, una órbita como las descritas en la sección 2.1.

Atendiendo a todo lo anterior, la trayectoria de un vehículo entre dos planetas puede dividirse en las siguientes etapas:

- **Fase planetocéntrica en el planeta de partida:** el vehículo se encuentra dentro de la esfera de influencia del cuerpo, por lo que la atracción gravitatoria del Sol es despreciada. El vehículo seguirá una trayectoria cónica, alcanzando una velocidad \mathbf{V}_{SoI} en la frontera de la gravisfera.
- **Fase heliocéntrica:** el vehículo se considera sometido únicamente a la fuerza gravitatoria del Sol. Esta fase parte del punto \mathbf{P}_1 , en la frontera de la esfera de influencia del planeta, con una velocidad $\mathbf{V}_1 = \mathbf{V}_{planet1} + \mathbf{V}_{SoI}$, y acaba en el punto \mathbf{P}_2 .
- **Llegada al planeta destino:** el vehículo entra en la esfera gravitacional del planeta destino, donde seguirá una curva cónica. La velocidad a la llegada es $\mathbf{V}_{SoI} = \mathbf{V}_2 - \mathbf{V}_{planet2}$.

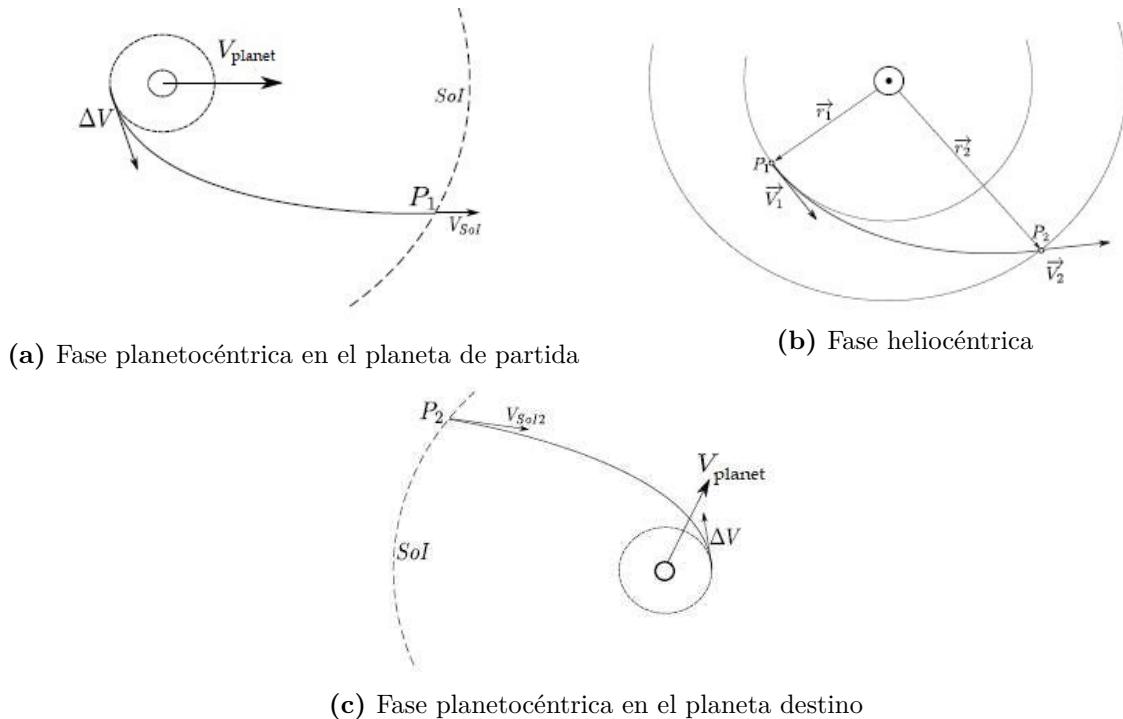


Figura 4.1: Fases de una trayectoria interplanetaria. Figura tomada de (Fernández (2014)).

Cuando se estudian las **fases planetocéntricas**, la esfera de influencia es considerada infinita, por tanto la velocidad de salida \mathbf{V}_{SoI} será la **velocidad hiperbólica** \mathbf{v}_∞ , lo cuál permite ajustar en la frontera de la gravisfera los segmentos de trayectoria heliocéntrica con los segmentos de trayectoria planetocéntrica.

Por otra parte, en la **fase heliocéntrica**, ya se ha visto que tanto la esfera de influencia como el propio planeta son representados como un punto centrado en el centro planetario, por lo que los puntos \mathbf{P}_1 y \mathbf{P}_2 se corresponden con las **posiciones de los planetas** para el instante de tiempo t en el que quiera producirse la salida y llegada, respectivamente.

De acuerdo a este modelo, por tanto, es posible emprender el diseño preliminar de una misión interplanetaria en el Sistema Solar, de modo que las trayectorias en el seno de las esferas de influencia de los cuerpos celestes sean aproximadas por curvas cónicas de movimiento Kepleriano, y el segmento de trayectoria heliocéntrica puede ser calculado recurriendo al problema de Lambert entre los puntos \mathbf{P}_1 y \mathbf{P}_2 .

4.3. El problema de Lambert

Como se ha visto, en una misión interplanetaria pueden diferenciarse varias etapas, lo que permite analizar detenidamente cada una de las maniobras realizadas.

El **problema de Lambert**, válido para un problema de dos cuerpos, puede por tanto, ser empleado para analizar la trayectoria heliocéntrica de la nave según el ajuste de cónicas.

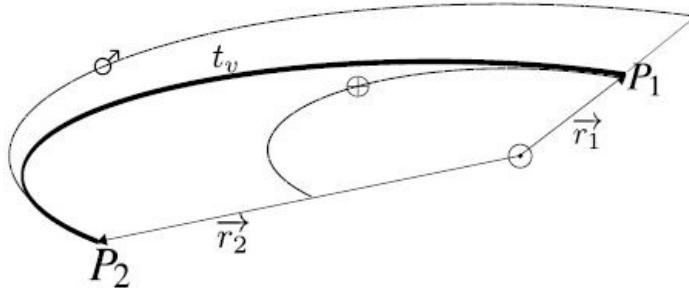


Figura 4.2: Problema de Lambert para una órbita entre la Tierra y Marte. Figura tomada de (Fernández (2014)).

Según Lambert, si se conocen los **puntos inicial P_1 y final P_2** de la trayectoria y el **tiempo de vuelo t_v** entre ambos, es posible calcular la órbita de transferencia que unirá ambos puntos.

Conociendo los vectores posición \mathbf{r}_1 y \mathbf{r}_2 de estos puntos, el cambio en la anomalía verdadera θ puede ser calculado:

$$\cos \Delta\theta = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \quad (4.5)$$

Sim embargo, mediante este cálculo surge ambigüedad a la hora de fijar el cuadrante en el que se encontraría θ , siendo necesario recurrir a la componente Z del vector resultante de $\mathbf{r}_1 \times \mathbf{r}_2$. Al mismo tiempo, es necesario considerar dos posibilidades:

- **Trayectoria directa (o prógrada)**, dónde $0^\circ < i < 90^\circ$.
- **Trayectoria retrógrada**, dónde $90^\circ < i < 180^\circ$.

Teniendo en cuenta todo lo anterior, se puede resolver la ambigüedad según:

$$\Delta\theta = \begin{cases} \cos \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} & (\mathbf{r}_1 \times \mathbf{r}_2)_Z \geq 0 \\ 360^\circ - \cos \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} & (\mathbf{r}_1 \times \mathbf{r}_2)_Z < 0 \end{cases} \quad (4.6a)$$

$$\Delta\theta = \begin{cases} \cos \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} & (\mathbf{r}_1 \times \mathbf{r}_2)_Z < 0 \\ 360^\circ - \cos \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} & (\mathbf{r}_1 \times \mathbf{r}_2)_Z \geq 0 \end{cases} \quad (4.6b)$$

Siendo el caso 4.6a para trayectorias *prógradas* y el caso 4.6b para trayectorias *retrógradas*.

Lambert establece que el **tiempo de transferencia Δt** desde el punto P_1 al punto P_2 es **independiente de la excentricidad e** de la órbita, **dependiendo** únicamente de la **suma de las magnitudes de los vectores posición** de los puntos ($r_1 + r_2$), el **semi-eje mayor a** de la elipse y la **cuerda c** que une P_1 y P_2 .

Si Δt es conocido entre P_1 y P_2 , o si de igual manera, se establece una fecha de comienzo y un tiempo de vuelo para la misión o una fecha de llegada al destino, el problema de Lambert

se reduce a **obtener la órbita que une ambos puntos**.

En el momento en que \mathbf{v}_1 es obtenido, la órbita de transferencia es calculada, pues mediante este vector de estado es posible calcular los elementos orbitales del cuerpo, o de otra forma, mediante los **coeficientes de Lagrange**, el resto de puntos pueden ser determinados.

$$\mathbf{r}_2 = f\mathbf{r}_1 + g\mathbf{v}_1 \quad (4.7a)$$

$$\mathbf{v}_2 = \dot{f}\mathbf{r}_1 + \dot{g}\mathbf{v}_1 \quad (4.7b)$$

Resolviendo para \mathbf{v}_1 , se tiene:

$$\mathbf{v}_1 = \frac{1}{g}(\mathbf{r}_2 - f\mathbf{r}_1) \quad (4.8)$$

Si ahora se sustituye en la ecuación 4.7b, y se tiene en cuenta la ecuación 4.9, finalmente se obtiene el vector \mathbf{v}_2 :

$$f\dot{g} - \dot{f}g = 1 \quad (4.9)$$

$$\mathbf{v}_2 = \frac{1}{g}(\dot{g}\mathbf{r}_2 - \mathbf{r}_1) \quad (4.10)$$

En definitiva, si los coeficientes de Lagrange pueden ser determinados, el problema de Lambert es resuelto.

Estos coeficientes pueden ser expresados en función del cambio en la anomalía verdadera $\Delta\theta$:

$$f = 1 - \frac{\mu r_2}{h^2}(1 - \cos \Delta\theta) \quad g = \frac{r_1 r_2}{h} \sin \Delta\theta \quad (4.11a)$$

$$\dot{f} = \frac{\mu}{h} \frac{1 - \cos \Delta\theta}{\sin \Delta\theta} \left[\frac{\mu}{h^2}(1 - \cos \Delta\theta) - \frac{1}{r_1} - \frac{1}{r_2} \right] \quad \dot{g} = 1 - \frac{\mu r_1}{h^2}(1 - \cos \Delta\theta) \quad (4.11b)$$

Sin embargo, estas expresiones puede re-escribirse en función de la **variable universal** χ :

$$f = 1 - \frac{\chi^2}{r_1} C(z) \quad g = \Delta t - \frac{1}{\sqrt{\mu}} \chi^3 S(z) \quad (4.12a)$$

$$\dot{f} = \frac{\sqrt{\mu}}{r_1 r_2} \chi [z S(z) - 1] \quad \dot{g} = 1 - \frac{\chi^2}{r_2} C(z) \quad (4.12b)$$

done $z = \alpha\chi^2$.

A partir del conjunto de ecuaciones previas puede establecerse una relación entre Δt y $\Delta\theta$:

$$\frac{r_1 r_2}{h} \sin \Delta\theta = \Delta t - \frac{1}{\sqrt{\mu}} \chi^3 S(z) \quad (4.13)$$

El **momento angular** h puede eliminarse de las expresiones anteriores si se igualan ambos

conjuntos para f , de manera que:

$$h = \sqrt{\frac{\mu r_1 r_2 (1 - \cos \Delta\theta)}{\chi^2 C(z)}} \quad (4.14)$$

Si ahora esta ecuación 4.14 es sustituida en la ecuación 4.13, en última instancia se obtiene:

$$\sqrt{\mu} \Delta t = \chi^3 S(z) + A \chi \sqrt{C(z)} \quad (4.15)$$

donde:

$$A = \sin \Delta\theta \sqrt{\frac{r_1 r_2}{1 - \cos \Delta\theta}} \quad (4.16)$$

Igualando las ecuaciones 4.11b y 4.12b para \dot{f} , se puede encontrar una relación entre z y χ que no involucre ningún parámetro orbital:

$$\frac{\mu}{h} \frac{1 - \cos \Delta\theta}{\sin \Delta\theta} \left[\frac{\mu}{h^2} (1 - \cos \Delta\theta) - \frac{1}{r_1} - \frac{1}{r_2} \right] = \frac{\sqrt{\mu}}{r_1 r_2} \chi [z S(z) - 1] \quad (4.17)$$

Sustituyendo h por la ecuación 4.14 y tras una serie de operaciones matemáticas, finalmente se obtiene:

$$\chi^2 C(z) = r_1 + r_2 + A \frac{z S(z) - 1}{\sqrt{C(z)}} \quad (4.18)$$

Si definimos $y(z)$ como:

$$y(z) = r_1 + r_2 + A \frac{z S(z) - 1}{\sqrt{C(z)}} \quad (4.19)$$

Queda una expresión más simplificada de la ecuación 4.18:

$$\chi = \sqrt{\frac{y(z)}{C(z)}} \quad (4.20)$$

La ecuación 4.20 es la relación entre χ y z que se buscaba. De forma que si ahora es sustituida en la ecuación 4.15:

$$\sqrt{\mu} \Delta t = \left[\frac{y(z)}{C(z)} \right]^{\frac{3}{2}} S(z) + A \sqrt{y(z)} \quad (4.21)$$

Dado Δt , resolver la ecuación 4.21 para z requiere recurrir al ya citado método iterativo de Newton.

Teniendo en cuenta todo lo anterior, y si \mathbf{r}_1 , \mathbf{r}_2 y Δt son conocidos, se puede recurrir al siguiente algoritmo (Curtis (2005)) para resolver el problema de Lambert.

1. Calcular la magnitud de los vectores \mathbf{r}_1 y \mathbf{r}_2 .
2. Elegir una órbita bien *prógrada* o bien *retrógrada*, y calcular $\Delta\theta$ de acuerdo a las ecuaciones 4.6a y 4.6b.
3. Calcular A según la ecuación 4.16.
4. Emplear el método de Newton para resolver la ecuación 4.21 para z . Las funciones a emplear para resolver la iteración son las siguientes:

$$F(z) = \left[\frac{y(z)}{C(z)} \right]^{\frac{3}{2}} S(z) + A\sqrt{y(z)} - \sqrt{\mu}\Delta t \quad (4.22)$$

$$F'(z) = \begin{cases} \left[\frac{y(z)}{C(z)} \right]^{\frac{3}{2}} \left(\frac{1}{2z} \left[(z) - \frac{3}{2} \frac{S(z)}{C(z)} \right] + \frac{3}{4} \frac{S(z)^2}{C(z)} \right) + \frac{A}{8} \left[3 \frac{S(z)}{C(z)} \sqrt{y(z)} + A \sqrt{\frac{C(z)}{y(z)}} \right] & (z \neq 0) \\ \frac{\sqrt{2}}{40} y(z)^{\frac{3}{2}} + \frac{A}{8} \left[\sqrt{y(z)} + A \sqrt{\frac{1}{2y(z)}} \right] & (z = 0) \end{cases} \quad (4.23)$$

donde $C(z)$ y $S(z)$ corresponden a las conocidas Stumpff functions.

5. Calcular $y(z)$ según la ecuación 4.19.
6. Calcular los coeficientes de Lagrange según las expresiones originadas al sustituir las ecuaciones 4.20 y 4.21 en las ecuaciones 4.12a y 4.12b:

$$f = 1 - \frac{y(z)}{r_1} \quad (4.24a)$$

$$g = A \sqrt{\frac{y(z)}{\mu}} \quad (4.24b)$$

$$\dot{f} = \frac{\sqrt{\mu}}{r_1 r_2} \sqrt{\frac{y(z)}{C(z)}} [zS(z) - 1] \quad (4.24c)$$

$$\dot{g} = 1 - \frac{y(z)}{r_2} \quad (4.24d)$$

7. Calcular los vectores \mathbf{v}_1 y \mathbf{v}_2 mediante las ecuaciones 4.8 y 4.10.

Una vez los vectores velocidad \mathbf{v}_1 y \mathbf{v}_2 han sido obtenidos, es posible calcular los **elementos orbitales que definen la trayectoria** entre \mathbf{P}_1 y \mathbf{P}_2 empleando el algoritmo expuesto en la sección 2.2.1 y tomando como vector de estado $\mathbf{r}_1, \mathbf{v}_1$ (o $\mathbf{r}_2, \mathbf{v}_2$).

4.4. Efemérides planetarias

Los puntos \mathbf{P}_1 y \mathbf{P}_2 vendrán definidos por la posición de los planetas en el instante considerado. Para poder encontrar estos puntos de la forma más fidedigna posible será necesario

calcular el vector de estado planetario correspondiente a ese instante t .

A partir del siguiente **propagador planetario**, basado en efemérides, este vector de estado podrá ser calculado para cada momento t del tiempo.

Ya que la precisión aquí requerida no es muy elevada, será suficiente con un cálculo aproximado y menos preciso de las efemérides mediante la **formulación Kepleriana** de las mismas. Los **elementos keplerianos** y el **ratio de su cambio** se muestran en las Tablas 4.1 y 4.2, estando referidos a la **época J2000**¹.

Esta formulación no es más que un **modelo lineal**, según el cuál los **elementos orbitales** que describen la órbita de un planeta se calculan de la siguiente forma:

$$(a, e, i, \Omega, \varpi, L) = (a_0, e_0, i_0, \Omega_0, \varpi_0, L_0) + (\dot{a}, \dot{e}, \dot{i}, \dot{\Omega}, \dot{\varpi}, \dot{L})T \quad (4.25)$$

Estos elementos keplerianos definen el **semi-eje mayor** a , la **excentricidad** e , la **inclinación** I , la **longitud media** L , la **longitud del perihelio** ϖ y la **longitud del nodo ascendente** Ω .

Tabla 4.1: Elementos keplerianos y sus ratios de cambio respecto a la época J2000, válidos para el intervalo 3000 BC - 3000 AD

	a [au, au/cty]	e [rad, rad/cty]	I [deg, deg/cty]	L [deg, deg/cty]	ϖ [deg, deg/cty]	Ω [deg, deg/cty]
Mercurio	0.38709843 0.00000000	0.20563661 0.00002123	7.00559432 -0.00590158	252.25166724 149472.67486623	77.45771895 0.15940013	48.33961819 -0.12214182
Venus	0.72332102 -0.00000026	0.00676399 -0.00005107	3.39777545 0.00043494	181.97970850 58517.81560260	131.76755713 0.05679648	76.67261496 -0.27274174
Tierra	1.00000018 -0.00000003	0.01673163 -0.00003661	-0.00054346 -0.01337178	100.46691572 35999.37306329	102.93005885 0.31795260	-5.11260389 -0.24123856
Marte	1.52371243 0.00000097	0.09336511 0.00009149	1.85181869 -0.00724757	-4.56813164 19140.29934243	-23.91744784 0.45223625	49.71320984 -0.26852431
Jupiter	5.20248019 -0.00002864	0.04853590 0.00018026	1.29861416 -0.00322699	34.33479152 3034.90371757	14.27495244 0.18199196	100.29282654 0.13024619
Saturno	9.54149883 -0.00003065	0.05550825 -0.00032044	2.49424102 0.00451969	50.07571329 1222.11494724	92.86136063 0.54179478	113.63998702 -0.25015002
Urano	19.18797948 -0.00020455	0.04685740 -0.00001550	0.77298127 -0.00180155	314.20276625 428.49512595	172.43404441 0.09266985	73.96250215 0.05739699
Neptuno	30.06952752 0.00006447	0.00895439 0.00000818	1.77005520 0.00022400	304.22289287 218.46515314	46.68158724 0.01009938	131.78635853 -0.00606302
Plutón	39.48686035 0.00449751	0.24885238 0.00006016	17.14104260 0.00000501	238.96535011 145.18042903	224.09702598 -0.00968827	110.30167986 -0.00809981

¹J2000 se refiere a la época de las coordenadas actualmente en uso. Se trata de una fecha precisa a la cual hacen referencia las coordenadas celestes, y que en este caso dicha fecha queda fijada el 1 de Enero del 2000

Y donde T es el número de siglos desde *J2000*, según:

$$T = \frac{T_{eph} - 2451545.0}{36525} \quad (4.26)$$

siendo T_{eph} el día Juliano al que estamos propagando, 2451545 el día Juliano correspondiente al 1 de Enero de 2000 y 36525 el número de días en un siglo.

Tabla 4.2: Términos adicionales para calcular M en Júpiter, Saturno, Urano, Neptuno y Plutón, válidos para el intervalo 3000 BC - 3000 AC

	b [°/cty ²]	c [°]	s [°]	f [°/cty]
Júpiter	-0.00012452	0.06064060	-0.35635438	38.35125000
Saturno	0.00025899	-0.13434469	0.87320147	38.35125000
Urano	0.00058331	-0.97731848	0.17689245	7.67025000
Neptuno	-0.00041348	0.68346318	-0.10162547	7.67025000
Plutón	-0.01262724			

Mediante estos elementos y atendiendo al algoritmo que se presenta a continuación (Standish y Williams (s.f.)), será posible calcular el vector de estado descriptivo del planeta para cualquier instante t .

1. Calcular el valor de cada elemento orbital de acuerdo a las ecuaciones 4.25 y 4.26.
2. Calcular el **argumento de perihelio** ω y la **anomalía media** M según las siguientes expresiones:

$$\omega = \varpi - \Omega \quad (4.27a)$$

$$M = L - \varpi + bT^2 + c \cos(fT) + s \sin(fT) \quad (4.27b)$$

3. Buscar la anomalía excéntrica E como solución de la ecuación de Kepler.
4. Calcular las coordenadas del vector posición \mathbf{r}' en el sistema de referencia heliocéntrico.

$$x' = a(\cos E - e) \quad (4.28a)$$

$$y' = a\sqrt{1 - e^2} \sin E \quad (4.28b)$$

$$z' = 0 \quad (4.28c)$$

5. Llevar a cabo un rotación desde el plano de referencia heliocéntrico con su eje X alineado en la dirección foco-perihelio al sistema de referencia de la eclíptica, con su eje X alineado hacia el equinoccio vernal, empleando la secuencia de rotación de Euler:

$$\mathbf{r}_{ecl} = R_3(-\Omega)R_1(-I)R_3(-\omega)\mathbf{r}' \quad (4.29)$$

$$x_{ecl} = (\cos \omega \cos \Omega - \sin \omega \sin \Omega \cos I)x' + (-\sin \omega \cos \Omega - \cos \omega \sin \Omega \cos I)y' \quad (4.30a)$$

$$y_{ecl} = (\cos \omega \sin \Omega + \sin \omega \cos \Omega \cos I)x' + (-\sin \omega \sin \Omega + \cos \omega \cos \Omega \cos I)y' \quad (4.30b)$$

$$z_{ecl} = (\sin \omega \sin I)x' + (\cos \omega \sin I)y' \quad (4.30c)$$

6. Calcular las coordenadas del vector posición \mathbf{r}_{eq} en el sistema de referencia ecuatorial si es necesario:

$$x_{eq} = x_{ecl} \quad (4.31a)$$

$$y_{eq} = \cos \varepsilon y_{ecl} - \sin \varepsilon z_{ecl} \quad (4.31b)$$

$$z_{eq} = \sin \varepsilon y_{ecl} + \cos \varepsilon z_{ecl} \quad (4.31c)$$

siendo $\varepsilon = 23^\circ 43' 928$ la oblicuidad del eje terrestre.

5. Diseño de Trayectorias Interplanetarias

En la actualidad, el diseño de la trayectoria interplanetaria de una nave en una misión espacial es una parte vital de la misma, pues una trayectoria eficiente en la cuál la masa de combustible embarcada sea la mínima necesaria, conlleva la posibilidad de extraer mayor cantidad de datos científicos de interés.

Es por esto que las maniobras asistidas por gravedad son recurrentemente utilizadas en estos ámbitos, pues permiten reducir el consumo energético a la vez que explotan las posibilidades científicas y de exploración.

En base a las herramientas y asunciones desarrolladas en los capítulos anteriores, a continuación se plantea la forma en que se ha llevado a cabo el cálculo preliminar y la búsqueda de una trayectoria interplanetaria con los mínimos requerimientos de impulso adicional entre un planeta de partida y un planeta destino, llevando a cabo una maniobra asistida por gravedad en un planeta intermedio.

5.1. Problema matemático

De acuerdo al modelo de **ajuste de cónicas** es posible dividir una trayectoria entre dos planetas en **tres subsegmentos** diferenciados: un primer tramo hiperbólico dentro de la esfera de influencia gravitacional del cuerpo de partida, un segundo subsegmento de trayectoria elíptica en el seno de la Heliosfera y un tramo hiperbólico a la llegada de la nave a la gravisfera del planeta destino.

Según esto, el problema que aquí se plantea puede ser dividido en dos subproblemas que incluyen los subsegmentos anteriores: una primera trayectoria entre el planeta de partida y el planeta en el cuál se llevará a cabo el fly-by, y otra trayectoria entre este planeta y el planeta destino.

El ajuste entre ambos subproblemas vendrá dado por el fly-by a realizar en el planeta intermedio, de forma que en esta maniobra se tendrá un impulso ΔV que servirá para acoplar ambos segmentos de la trayectoria interplanetaria.

Sin embargo, este no es el único impulso que aparece a lo largo de la trayectoria completa, puesto que para abandonar la gravisfera del planeta partida, y de igual modo, quedar sometido a la influencia del planeta destino, dos impulsos definidos ΔV adicionales serán necesarios.

Por tanto, la forma de evaluar qué trayectoria es más adecuada será función del valor de un ΔV total, definido como la suma de los impulsos descritos anteriormente.

Para llevar esto a la práctica, en primer lugar, se seleccionarán los **planetas origen** y **destino**, así como un **planeta escala** en el que tendrá lugar la maniobra de asistencia. De igual manera se fijará un rango de **fechas de partida** y un rango de tiempos de vuelo, donde:

- T_{tof1} es el tiempo de vuelo empleado entre el planeta origen y el planeta escala.
- T_{tof2} es el tiempo de vuelo empleado entre el planeta escala y el planeta destino.

Para cada una de las posibles combinaciones de fecha y tiempos de vuelo, se evaluará el ΔV_{total} de la trayectoria, siendo el menor de ellos la solución para el problema establecido. Posteriormente, y partiendo de este resultado, se habrá de buscar un nuevo mínimo re-iterando el problema expuesto, como se verá en la sección 5.2.

5.1.1. Método Práctico

Cuando los parámetros que se mencionaron en la sección anterior ya han sido fijados, para cada posible combinación de fecha y tiempos de vuelo, el procedimiento práctico para obtener el ΔV_{total} se describe a continuación.

En primer lugar, se calcula el **subsegmento elíptico** que describe la trayectoria entre el **planeta de partida** y el **planeta escala**, en el intervalo entre la fecha de inicio T_0 y el tiempo de vuelo establecido T_{tof1} .

De acuerdo al modelo de ajuste de cónicas, la **posición heliocéntrica de la nave** en esos instantes de tiempo se corresponderá con la posición de los planetas de origen y escala (\mathbf{R}_1 , \mathbf{R}_2).

Dado que T_0 y T_{tof1} son conocidos, es posible obtener, mediante el cálculo de las efemérides planetarias, tanto los vectores posición (\mathbf{R}_1 , \mathbf{R}_2) como los vectores velocidad (\mathbf{V}_1 , \mathbf{V}_2) de ambos planetas. Es decir, se pueden definir los puntos \mathbf{P}_1 y \mathbf{P}_2 con los que recurrir al **problema de Lambert**.

$$\mathbf{R}_1 = \mathbf{R}(T_0) , \quad \mathbf{V}_1 = \mathbf{V}(T_0) \quad (5.1a)$$

$$\mathbf{R}_2 = \mathbf{R}(T_0 + T_{tof1}) , \quad \mathbf{V}_2 = \mathbf{V}(T_0 + T_{tof1}) \quad (5.1b)$$

La resolución del problema de Lambert proporcionará las **velocidades heliocéntricas** de la nave en el momento que abandona el planeta origen ($\mathbf{V}_{departure}^{(1)}$)¹ y llega al planeta de escala ($\mathbf{V}_{arrival}^{(2)}$).

Empleando una de las parejas de vectores resultantes (\mathbf{R}_1 , $\mathbf{V}_{dep}^{(1)}$) ó (\mathbf{R}_2 , $\mathbf{V}_{arr}^{(2)}$), se pueden calcular los **elementos orbitales** descriptivos de la trayectoria atendiendo al algoritmo expuesto en la sección 2.2.1.

Sin embargo, para poder entrar en este subsegmento de la trayectoria, primero es necesario abandonar el tirón gravitacional generado por el planeta de origen.

¹El superíndice (X) refiere al planeta en torno al cuál la nave se encuentra.

Esto se logra aportando un ΔV adicional a la nave, que será definido como $\Delta V_{departure}$.

Si se considera que el subsegmento heliocéntrico sigue una **órbita de transferencia de Hohmann**, dónde el punto de partida del planeta origen es el periapsis de la órbita y el punto de llegada es el apoasis de la misma, se encuentra que el **vector velocidad heliocéntrica de la nave a la salida** $\mathbf{V}_{dep}^{(1)}$ es paralelo tanto a la asíntota de la hipérbola de salida como al **vector velocidad del planeta** \mathbf{V}_1 .

Con la última deducción y teniendo en cuenta que el módulo del vector $\mathbf{V}_{dep}^{(1)}$ de la nave es mayor que el módulo del vector \mathbf{V}_1 del planeta en la frontera de la gravisfera, se llega a la conclusión que el ΔV_{dep} será igual al **exceso de velocidad hiperbólica** v_∞ :

$$\Delta V_{dep} = v_\infty = V_{dep}^{(1)} - V_1 \quad (5.2)$$

Una vez que el segmento entre el planeta de origen y el planeta sobre el que se realizará la maniobra de asistencia ha sido definido, a continuación se procede a calcular el segmento definido entre este **planeta escala** y el **planeta destino** de la trayectoria.

El método de cálculo es exactamente el mismo que para el primer segmento: en primer lugar se obtienen los vectores de estado descriptivos de ambos planetas para los instantes definidos:

$$\mathbf{R}_2 = \mathbf{R}(T_0 + T_{tof1}) , \quad \mathbf{V}_2 = \mathbf{V}(T_0 + T_{tof1}) \quad (5.3a)$$

$$\mathbf{R}_3 = \mathbf{R}(T_0 + T_{tof1} + T_{tof2}) , \quad \mathbf{V}_3 = \mathbf{V}(T_0 + T_{tof1} + T_{tof2}) \quad (5.3b)$$

Una vez conocidos estos vectores de estado, se emplea el **algoritmo de Lambert** para obtener tanto el vector **velocidad heliocéntrica de salida** del planeta escala $\mathbf{V}_{departure}^{(2)}$, como el vector **velocidad heliocéntrica de llegada** al planeta destino $\mathbf{V}_{arrival}^{(3)}$.

Igual que en el segmento anterior era necesario proporcionar un ΔV para abandonar la influencia gravitacional del planeta origen, al llegar al planeta destino será necesario aportar un ΔV para quedar en órbita en torno a este. Este ΔV será definido por la **velocidad hiperbólica excedente**:

$$\Delta V_{arr} = v_\infty = V_{arr}^{(3)} - V_3 \quad (5.4)$$

Cuando ambas trayectorias heliocéntricas entre los tres planetas han quedado completamente definidas, el siguiente y último paso es analizar el fly-by en el seno de la gravisfera del planeta escala. Esta maniobra permitirá acoplar ambos segmentos en el entorno global de la trayectoria.

Si se hace recapitulación de los pasos previos, se observa que las **velocidades heliocéntricas** tanto a la llegada como a la salida del fly-by quedan fijadas por la **solución del problema de Lambert**.

Para alcanzar mediante el fly-by las condiciones de salida definidas para iniciar el segundo subsegmento heliocéntrico, se requiere de un impulso ΔV , que será definido como ΔV_{flyby} .

A la hora de calcular este impulso, es necesario diferenciar dos casos:

- El **radio de pericentro** de la hipérbola es **mayor** que el **radio del planeta**.
- El **radio de pericentro** de la hipérbola es **menor** que el **radio del planeta**.

El **radio de pericentro** r_p define la **altura** a la que tiene lugar la maniobra de asistencia.

Atendiendo ahora a lo expuesto en el capítulo 3, resulta que el **ángulo φ rotado** por el vector velocidad durante la maniobra de asistencia es función de r_p .

Por tanto, φ puede calcularse fácilmente conocidas las condiciones inciales y finales del fly-by: φ queda definido por los vectores velocidad a la llegada y a la salida de la maniobra (en coordenadas planetocéntricas):

$$\mathbf{v}_{arr}^{(2)} = \mathbf{v}_{\infty 1} = \mathbf{V}_{arr}^{(2)} - \mathbf{V}_2 \quad (5.5a)$$

$$\mathbf{v}_{dep}^{(2)} = \mathbf{v}_{\infty 2} = \mathbf{V}_{dep}^{(2)} - \mathbf{V}_2 \quad (5.5b)$$

Una vez conocido φ , se calcula el radio de periapsis según:

$$r_p = \frac{K_2(\frac{1}{\sin \varphi/2} - 1)}{(v_{arr}^2)^2} \quad (5.6)$$

siendo K_2 el parámetro gravitacional del planeta escala.

En el caso en que $r_p > r_{planeta}$, la maniobra de fly-by rotará el vector velocidad de entrada de manera que a la salida tendrá la **misma dirección** que el vector velocidad heliocéntrica calculado mediante Lambert.

$$\mathbf{v}_{\infty 2} \parallel \mathbf{V}_{dep}^{(2)} \quad (5.7)$$

Sin embargo, el módulo de este vector (en coordenadas heliocéntricas) será menor que el del vector definido por Lambert para iniciar la transferencia hacia el planeta destino, siendo por tanto necesario un impulso ΔV_{flyby} de módulo:

$$\Delta V = V_{dep}^{(2)} - V_{arr}^{(2)} \quad (5.8)$$

En el caso en que $r_p < r_{planeta}$, la maniobra de fly-by no rotará el vector entrada el ángulo φ calculado, pues supondría atravesar el planeta, si no que producirá una **rotación máxima** φ_{max} que corresponderá a una altura r_p igual al radio del planeta más un margen establecido para su atmósfera. Por tanto, modificando la ecuación 3.17, se obtiene:

$$\varphi_{max} = 2 \arcsin \frac{K_2}{K_2 + (r_{max} v_{\infty 1}^2)} \quad (5.9)$$

donde

$$r_{max} = r_2 + h_{atmosfera} \quad (5.10)$$

En este caso, el vector salida del fly-by tendrá tanto un **módulo** como una **dirección** diferente al vector velocidad requerido para la siguiente transferencia. Por ello, el impulso $\Delta\mathbf{V}_{flyby}$ será necesario tanto en módulo como en dirección:

$$\Delta\mathbf{V}_{flyby} = \mathbf{V}_{dep}^{(2)} - \mathbf{V}_{flyby} \quad (5.11)$$

Donde \mathbf{V}_{flyby} es calculado de acuerdo a la ecuación 3.3b y empleando φ_{max} en la matriz de rotación definida por la ecuación 3.4.

En definitiva, el problema matemático desarrollado puede ser resumido en el siguiente algoritmo.

1. Obtener los vectores descriptivos para cada planeta en los instantes establecidos mediante las expresiones 5.1 y 5.3.
 2. De acuerdo al modelo de ajuste de cónicas y recurriendo al algoritmo de Lambert, definir la trayectoria heliocéntrica entre el planeta de origen y el planeta escala. Se obtienen aquí los vectores $\mathbf{V}_{dep}^{(1)}$ y $\mathbf{V}_{arr}^{(2)}$.
 3. Realizar el paso anterior para la trayectoria entre el planeta escala y el planeta destino, obteniendo en este caso los vectores $\mathbf{V}_{dep}^{(2)}$ y $\mathbf{V}_{arr}^{(3)}$.
 4. Emplear las ecuaciones 5.2 y 5.4 para obtener el impulso ΔV_{dep} necesario para abandonar la gravisfera del planeta origen, y ΔV_{arr} para quedar en órbita en torno al planeta destino.
 5. Calcular los vectores velocidad de llegada y velocidad de salida del fly-by en coordenadas planetocéntricas $\mathbf{v}_{arr}^{(2)}$ y $\mathbf{v}_{dep}^{(2)}$ de acuerdo a las ecuaciones 5.5.
 6. Obtener el ángulo φ que definen los dos vectores anteriores, y calcular r_p según la ecuación 5.6.
 7. En función de que condición se encuentre ($r_p > r_{planeta}$ o $r_p < r_{planeta}$), calcular ΔV_{flyby} según las ecuaciones 5.8 o 5.11.
-

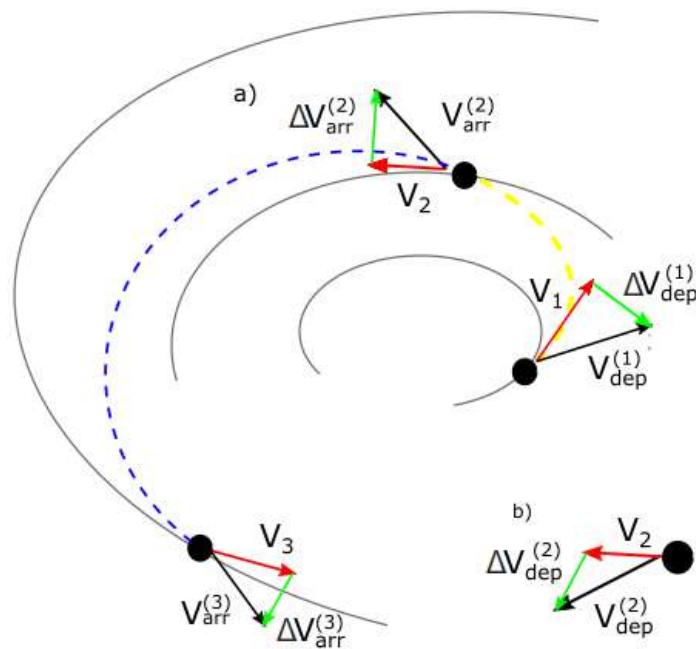


Figura 5.1: Definición de una trayectoria interplanetaria. El caso a) refiere la transferencia entre el planeta 1 y 2, y el caso b) refiere la transferencia entre el planeta 2 y 3.

5.2. Búsqueda de la transferencia de menor ΔV_{total}

Una vez todos los impulsos han sido calculados, la suma de todos ellos será el ΔV_{total} que definirá la solución del problema expuesto para una combinación dada T_0 , T_{tof1} , T_{tof2} .

Como la serie de combinaciones posibles es elevada, el resultado es una malla de posibles ΔV_{total} que son solución para la trayectoria deseada entre el planeta origen y el planeta de destino.

En este caso, la solución que se busca es aquella que minimice el valor final de ΔV_{total} , pues supone que el gasto energético adicional o artifical requerido es mínimo.

Sin embargo, tras resolver todas las posibles combinaciones y encontrar el valor mínimo entre todas ellas, puede que esta primera solución sea una mera aproximación al mínimo absoluto que pueda obtenerse. Es decir, el valor calculado como mínimo del problema en la primera iteración puede ser un **mínimo local** en lugar del mínimo absoluto buscado.

Para buscar, por tanto, una trayectoria que minimice la solución del problema, se puede emplear el resultado encontrado como semilla o punto de partida para buscar una mejor solución.

En este caso, dado que el mallado contiene las soluciones de una primera iteración, se llevará a cabo un **refinamiento de la malla** generada en torno al valor mínimo encontrado.

Dado que los rangos de fechas de partida y tiempos de vuelo definidos son muy amplios, al generar todas las posibles combinaciones, el delta de tiempo Δt que sigue el programa puede dejar fuera de la iteración la combinación de fecha de partida y tiempos de vuelo que resuelvan el mínimo absoluto del problema.

Por tanto, después de una primera iteración del problema y ya con un valor ΔV_{total} mínimo inicial, se analiza qué fecha de partida T_0 , y qué tiempos de vuelos establecidos T_{tof1} y T_{tof2} son los que conducen a esta primera solución.

Tomando como referencia la fecha T_0 y los tiempos T_{tof1} y T_{tof2} , a continuación se reajustan los rangos de posibles valores para estos parámetros, y se reduce el Δt del programa, con el objetivo de crear una malla más reducida en torno a la solución previa.

De esta forma, al interar de nuevo el problema, como la nueva malla incluye valores y rangos próximos a la solución previa, la nueva solución ΔV_{total} se aproximará más al mínimo absoluto del problema, siendo menor que la solución generada en la primera iteración.

En caso de que la malla devuelva valores mayores que la solución previa, el ΔV_{total} obtenido en la primera etapa se correspondería con el mínimo absoluto.

Tanto el problema matemático expuesto en la sección 5.1, como su algoritmo de resolución y optimización han sido llevados a la práctica mediante el desarrollo paralelo de un programa empleando **Python**.

La implementación de este programa será desarrollada en el próximo capítulo.

6. Implementación Práctica del Problema

A la hora de llevar a la práctica la resolución del problema expuesto en el capítulo anterior se ha recurrido a su implementación empleando **Python** como lenguaje de programación.

Python es un lenguaje de **código abierto** que permite una programación funcional, imperativa y orientada a objetos, que lo convierte en la actualidad en uno de los lenguajes más empleados en los ámbitos científico, matemático y para desarrollos de big-data.

En este entorno, Python dispone de una serie de **librerías** orientadas a facilitar el trabajo con órbitas, transferencias, transformaciones entre vectores de estado y elementos orbitales, etc. Ejemplos de estas librerías son **pykep** o **astropy**.

Sin embargo, para este trabajo se ha decidido desarrollar por completo el programa que conducirá a la minimización del ΔV_{total} solución de la trayectoria deseada.

Para ello, se han empleado las librerías **NumPy** y **Matplotlib**, las cuáles están orientadas a la programación científica, permitiendo manejar arrays multimensionales y representar gráficamente datos y funciones.

A continuación se expondrán las principales partes del código desarrollado para implementar el problema expuesto.

El código completo puede ser encontrado en el ANEXO B, y la validación del mismo en el ANEXO C.

6.1. Implementación y código

De acuerdo al algoritmo resumido que aparece al final de la sección 5.1.1, en primer lugar es necesario obtener los **vectores de estado** correspondientes a los planetas elegidos en los instantes T_0 , T_{tof1} y T_{tof2} respectivamente.

Esta primera parte queda definida mediante el código 6.1.

Código 6.1: Función que retorna el vector de estado de cada planeta

```
def planet_state(planet, T_ephemeris, t): # t in days
    t_julian = julian_date(T_ephemeris, t)

    h_p, a_p, e_p, I_p, L_p, w1_p, w_p, O_p, M_p = planets_elements(planet, ↵
        ↵ t_julian)

    E_p, theta_p = kepler_eliptic(e_p, M_p)

    R_planet, V_planet = coe_to_vstate(h_p, e_p, O_p, I_p, w_p, theta_p)
```

```
    return R_planet, V_planet
```

Este código emplea a su vez las funciones definidas *t Julian*, *planet elements*, *kepler elliptic* y *coe to vstate*.

Estas funciones, cuyo encabezado y retorno se definen el Código 6.2, devuelven los siguientes parámetros:

- *t Julian*: devuelve los días que han pasado desde la época J2000 hasta el día juliano definido por el parámetro *t ephemeris*. El parámetro *t* permite calcular los días pasados desde J2000 partiendo de la referencia fijada por *t ephemeris*.
- *planet elements*: devuelve los elementos órbitales keplerianos que definen la órbita del planeta introducido mediante el parámetro *planet* para un tiempo juliano dado.
- *kepler elliptic*: resuelve la ecuación de Kepler a partir de la **excentricidad** *e* de la órbita y la **anomalía media** *M* según el algoritmo expuesto en la sección 2.3.1.
- *coe to vstate*: retorna el vector de estado descriptivo del planeta para el instante *t* de acuerdo al algoritmo de la sección 2.2.2.

Código 6.2: Funciones referidas en planet_state

```
def julian_date(t_ephemeris, t): # t in days

    t_julian = (t_ephemeris - 2451545.0) / 36525.0
    t_julian = t_julian + (t / 36525.0)

    return t_julian

def planets_elements(planet, julian_time):
    .
    .
    .
    return h_p, a_p, e_p, I_p, L_p, w1_p, w_p, O_p, M_p

def kepler_elliptic(e, M):
    .
    .
    .
    return E, theta

def coe_to_vstate(h, e, RA, inc, w, theta):
    .
    .
    .
    return R, V
```

Una vez los vectores de estado de cada planeta han sido calculados, el siguiente paso consistía en **definir los subsegmentos heliocéntricos** entre el planeta de partida y el planeta escala, y entre el planeta escala y el planeta destino.

Para llevar a cabo este cálculo, se ha desarrollado una función genérica que, incluyendo el cálculo de las efemérides planetarias, permite definir por completo una **transferencia heliocéntrica** entre **dos planetas cualesquiera**, dados una fecha juliana de referencia

T_{ephem_begin} , un instante inicial t_0 , un segundo instante t_1 y un tiempo de vuelo cualquiera fly_time .

Código 6.3: Función que devuelve la transferencia heliocéntrica entre dos planetas.

```
def segment_trajectory(planet_1, planet_2, T_ephem_begin, t_0, t_1, fly_time):
    R_planet1, V_planet1 = planet_state(planet_1, T_ephem_begin, t_0) # [km]
    R_planet2, V_planet2 = planet_state(planet_2, T_ephem_begin, t_1) # [km]

    V_departure, V_arrival = lambert_problem(R_planet1, R_planet2, fly_time) # ↪
    ↪ [km/s]

    DV_departure = V_departure - V_planet1 # [km/s]
    DV_arrival = V_arrival - V_planet2 # [km/s]

    return R_planet1, V_planet1, R_planet2, V_planet2, V_departure, ↪
    ↪ DV_departure, V_arrival, DV_arrival
```

Mediante el parámetro T_{ephem_begin} se establece la fecha de referencia de **inicio de la trayectoria**, y mediante t_0 y t_1 se definen los instantes en los que se quiere conocer el vector de estado del planeta de partida y del planeta llegada respectivamente. Además, estos instantes permiten hacer un registro del tiempo que dura la misión.

Por último, el parámetro fly_time establece el tiempo de vuelo, en días, que debe durar la transferencia desde el planeta de partida hasta el planeta de llegada definidos para este subsegmento.

El cálculo de los vectores de estado de los planetas se lleva a cabo empleando la función **planet_state**, definida en el Código 6.1, para los planetas seleccionados en los instantes t_0 y t_1 . Una vez los valores de posición y velocidad de ambos quedan almacenados, se emplea la función **lambert_problem** que resuelve el problema de Lambert de acuerdo al algoritmo 4.3, devolviendo las velocidades de salida y llegada heliocéntricas de la nave ($\mathbf{V}_{departure}$, $\mathbf{V}_{arrival}$).

Las dos sentencias siguientes definen las velocidades de partida y llegada en coordenadas planetocéntricas ($\mathbf{DV}_{departure}$, $\mathbf{DV}_{arrival}$).

Por último, la sentencia retorno de la función **segment_trajectory** devuelve como variables las posiciones y velocidades para cada uno de los planetas ($\mathbf{R}_{planet1}$, $\mathbf{V}_{planet1}$, $\mathbf{R}_{planet2}$, $\mathbf{V}_{planet2}$), las velocidades heliocéntricas de partida y llegada de la nave ($\mathbf{V}_{departure}$, $\mathbf{V}_{arrival}$), y estas mismas velocidades pero en el sistema de referencia planetocéntrico ($\mathbf{DV}_{departure}$, $\mathbf{DV}_{arrival}$).

En definitiva, esta función devuelve todo lo necesario para calcular tanto los impulsos a la salida del primer planeta ΔV_{dep} como a la llegada al planeta destino ΔV_{arr} de acuerdo al punto 4 del algoritmo 5.1.1.

Igualmente, se tienen todas las variables necesarias para calcular la maniobra de asistencia y el ΔV_{flyby} resultante.

Dado que estos cálculos deben ser realizados para cada posible combinación de fecha de partida y tiempos de vuelo establecidos, se empleará un **bucle for** de manera que todas las

combinaciones sean recorridas y para cada una de ellas se efectúen los cálculos necesarios (Código 6.4).

Código 6.4: Bucle for que retornará los valores de los impulsos necesarios.

```

1   for i in range(len(X[0, :, 0])):
2       for j in range(len(Y[:, 0, 0])):
3
4           R_1dep, V_1dep, R_2arr, V_2arr, V_hdep, DV_dep, V_harr_1, DV_arr_1 =←
5               ↪ segment_trajectory(planet_1, planet_2, X[0, i, 0], t_0, Y[j, ←
6               ↪ 0, 0], Y[j, 0, 0])
7
8           DV0_departure[i, j] = np.linalg.norm(DV_dep)
9           DV_arrival_planet_1[i, j] = np.linalg.norm(DV_arr_1)
10
11          for k in range(len(Z[0, 0, :])):
12              R_2dep, V_2dep, R_3arr, V_3arr, V_hdep_1, DV_dep_1, V_harr_2, ←
13                  ↪ DV_arr_2 = segment_trajectory(planet_2, planet_3, X[0, i, ←
14                  ↪ 0], t_0 + Y[j, 0, 0], t_0 + Y[j, 0, 0] + Z[0, 0, k], Z[0, ←
15                  ↪ 0, k])
16
17              DV0_departure_1[j, k] = np.linalg.norm(DV_dep_1)
18              DV_arrival_planet_2[j, k] = np.linalg.norm(DV_arr_2)
19
20              fi, r_p = r_periapse(DV_arr_1, DV_dep_1, K_planet2)
21              ratio_radius[i, j, k] = r_p / r_planet2
22              fi_flyby[i, j, k] = fi
23              if r_p > r_planet2:
24                  DV_flyby = V_hdep_1 - V_harr_1
25                  DV_flyby_mag = np.linalg.norm(DV_flyby)
26                  DV_1_flyby[i, j, k] = DV_flyby_mag
27                  fi_max[i, j, k] = 0
28              elif r_p <= r_planet2:
29                  r_angle, V_flyby = fly_by(R_2arr, DV_arr_1, V_2dep, K_planet2←
30                                  , r_planet2)
31                  DV_flyby = V_hdep_1 - V_flyby
32                  DV_flyby_mag = np.linalg.norm(DV_flyby)
33                  DV_1_flyby[i, j, k] = DV_flyby_mag
34                  fi_max[i, j, k] = r_angle
35                  fi_flyby[i, j, k] = r_angle
36
37              DV_total[i, j, k] = DV0_departure[i, j] + DV_1_flyby[i, j, k] + ←
38                  ↪ DV_arrival_planet_2[j, k]

```

Esta secuencia de bucles anidados permite obtener los impulsos requeridos según la combinación que quede definida mediante las variables locales de control (**i**, **j**, **k**) que fijan la fecha de partida T_0 y los tiempos de vuelo T_{tof1} y T_{tof2} .

Las sentencias definidas en las líneas 6 y 14 calculan y almacenan el valor de los impulsos ΔV_{dep} y ΔV_{arr} . Se observa que la magnitud de este impulso coincide con la magnitud de la

velocidad de partida y llegada en el sistema de referencia planetocéntrico que devuelve la resolución del problema de Lambert.

Las sentencias comprendidas entre las líneas 16 y 30 resuelven el problema del fly-by siguiendo los algoritmos expuestos en la sección 5.1.1.

En primer lugar se calcula el valor del radio de pericentro de la hipérbola mediante la función definida ***r_periapse***, para después, mediante una construcción condicional comparar este valor con el radio del planeta sobre el cuál se llevará a cabo la maniobra.

Código 6.5: Función que evalúa el valor del parámetro r_p .

```
def r_periapse(v_arrival, v_departure, K_planet): # v_arrival, v_departure are ←
    ↪ given in planetocentric coor

    # v_arrival defines the planetocentric velocity of the spacecraft at the ←
    ↪ inbound crossing point of the flyby planet
    # v_departure defines the planetocentric velocity of the spacecraft at the ←
    ↪ outbound crossing point of this planet
    rotated_angle = np.dot(v_arrival, v_departure) / (np.linalg.norm(v_arrival) ←
        ↪ * np.linalg.norm(v_departure))
    rotated_angle = np.arccos(rotated_angle)

    r_p = (K_planet * ((1 / np.sin(rotated_angle / 2)) - 1)) / (np.linalg.norm(←
        ↪ v_arrival)) ** 2

    return rotated_angle, r_p
```

Una vez la comparación es realizada, en función de que rama condicional sea necesario abordar, el valor del ΔV_{flyby} será calculado acorde a lo expuesto en el capítulo anterior.

En el caso de que $r_p < r_{planeta}$ será necesario llevar a cabo el cálculo de la maniobra de fly-by mediante la función definida a tal efecto ***fly_by***, que devolverá tanto el ángulo máximo que se rotará en la maniobra φ_{max} , definido por la variable ***r_angle***, como la dirección del vector impulso necesario ***DV_{flyby}***.

Código 6.6: Función que evalúa la maniobra de asistencia por gravedad.

```
def fly_by(r_1, v_1, V_planet, K_planet, r_planet):
    v_infinity = np.linalg.norm(v_1)

    v_2 = rotation_vel_matrix(r_1, v_1, K_planet, r_planet) @ v_1.T
    V_2 = V_planet + v_2

    rot_angle = rot_vel_angle(v_infinity, K_planet, r_planet)

    return rot_angle, V_2
```

El significado de los parámetros de entrada de la función ***fly_by*** es el siguiente:

- ***r_1*** es la posición de la nave en el momento que cruza la frontera de la gravisfera planetaria. De acuerdo al modelo de ajuste de cónicas este valor se approxima a la posición del planeta en ese instante de tiempo. Es decir, en este caso, y de acuerdo a

la sentencia 25 del código 6.4, la posición de la nave vendrá aproximada por el vector R_{2arr} que define la posición del planeta sobre el que se está realizando el fly-by para el instante definido $T_0 + T_{tof1}$.

- v_1 define la velocidad planetocéntrica de llegada de la nave. En este caso, se corresponde con el vector que retorna el problema de Lambert DV_arr_1 .
- V_planet se corresponde con el vector velocidad del planeta escala en el instante $T_0 + T_{tof1}$, representado por el vector V_2dep devuelto por la segunda ejecución del problema de Lambert.
- r_planet , K_planet son el radio y el parámetro gravitacional del planeta sobre el que se está realizando el fly-by.

Las funciones ***rotation_vel_matrix*** y ***rot_vel_angle*** son funciones que calculan respectivamente la **matriz de rotación** definida en la ecuación 3.4 y el **ángulo de rotación** de acuerdo a la ecuación 3.17.

Tras todo lo anterior, en última instancia, la sentencia 32 del Código 6.4 calcula y almacena en su array correspondiente el valor del ΔV_{total} que implica la trayectoria definida por la fecha de partida **i**, el tiempo de vuelo T_{tof1} **j** y el tiempo de vuelo T_{tof2} **k**.

Una vez todo el bucle ha sido ejecutado, los arrays que almacenan los valores calculados para ΔV_{dep} , ΔV_{arr} , ΔV_{fly-by} , ΔV_{total} , r_p^1 y φ^2 (sentencias 6, 14, 22 o 28, 32, 17, 18 o 30, 23 o 29 respectivamente) son empleados para generar las gráficas.

Con esta simple ejecución de la serie de bucles, se encuentra una primera solución para ΔV_{total} que será empleada para refinar los rangos definidos al principio y que permitieron generar los arrays X, Y, Z que encabezan los bucles.

Estos arrays, tanto los generados para la primera iteración, como los ahora generados para llevar a cabo una segunda iteración, son definidos en función de una serie de constantes que establecen los límites temporales de la misión:

Código 6.7: Ejemplo de creación de los arrays de fechas y tiempos.

```

T_begin = 2457300.0 # 4th October 2015
T_end = 2457600.0 # 30th July 2016

T_tof_1 = 200 # 200 days of flight
T_tof_1_end = 550 # 500 days of flight

T_tof_2 = 600 # 600 days of flight
T_tof_2_end = 1000 # 1000 days of flight

```

¹El array *ratio_radius* representa el ratio entre el radio de la maniobra y el radio planetario, de forma que la gráfica mostrará aquellas combinaciones donde el ratio es mayor que uno de un color y aquellas donde es menor en otro color.

²Si el radio de la maniobra es menor que el radio planetario, el fly-by producirá un rotación φ_{max} del vector velocidad, que será menor que la teóricamente necesaria. Es por ello que se registran los valores de φ_{max} para representar gráficamente aquellas combinaciones donde esto ocurre.

```

...
x = np.arange(T_beg, T_fin, 5)
y = np.arange(T_tof_1_beg, T_tof_1_fin, 5)
z = np.arange(T_tof_2_beg, T_tof_2_fin, 5)
X, Y, Z = np.meshgrid(x, y, z)

```

Los arrays x , y , z definen **arrays unidimensionales** con los extremos fijados por las constantes establecidas, y cuyas posiciones están ocupadas por fechas crecientes con una discretización de 5 días.

Sin embargo, lo que en realidad es necesario no son arrays unidimensionales individuales, sino un **mallado** que defina posiciones en función de tres coordenadas, que serán las fechas establecidas por cada combinación de x , y , z .

La última sentencia del código resuelve esta necesidad, creando un mallado tridimensional a partir de los arrays unidimensionales x , y , z .

Cuando se obtiene la primera solución que minimiza ΔV_{total} , se extraen las posiciones $[i, j, k]$ del array **DV_total** que almacena este valor.

Código 6.8: Cálculo del mínimo ΔV_{total} y las fechas en las que se obtiene.

```

min_1 = np.min(DV_total)
position_min_1 = np.unravel_index(np.argmin(DV_total), DV_total.shape)

```

Con estas dos sentencias se extrae, en primer lugar, el valor mínimo de ΔV_{total} y, en segundo lugar, una lista de tres posiciones donde se almacenan los índices del array **DV_total** en los que se encontró el valor mínimo.

Con esto, se re-ajustan las constantes que definían los arrays iniciales de tiempos según:

Código 6.9: Re-ajuste de las constantes de fechas.

```

eps = 30

T_begin_opt = X[0, position_min_1[0], 0] - eps
T_end_opt = X[0, position_min_1[0], 0] + eps

T_tof_1_opt = Y[position_min_1[1], 0, 0] - eps
T_tof_1_end_opt = Y[position_min_1[1], 0, 0] + eps

T_tof_2_opt = Z[0, 0, position_min_1[2]] - eps
T_tof_2_end_opt = Z[0, 0, position_min_1[2]] + eps

```

En primer lugar se establece un margen, **eps** , que fijará los días en torno a los cuáles atender en función de la solución mínima previa.

Después, las constantes definidas se modifican de acuerdo a las sentencias anteriores: se toma el array correspondiente y en él se busca la posición i , j , k , según corresponda, donde se encontró el valor mínimo en la iteración anterior. Este valor i , j , k corresponderá a una posición del array X , Y , Z , la cuál almacena una fecha definida.

Por tanto, se extrae la fecha correspondiente almacenada en el índice i, j, k en la que se encontró el mínimo, y a esta se le suma y resta un valor de días estimado definido por eps, generando una malla reducida en torno a esta solución.

Finalmente, para llevar a cabo una nueva iteración, se ejecuta a continuación el Código 6.4 de nuevo, y una vez finalizado, las sentencias definidas en el Código 6.10 devolverán el mínimo ΔV_{total} solución del problema y las fechas en que se encuentra.

Código 6.10: Búsqueda de una nueva solución mínima.

```
min_opt = np.min(DV_total)
position_min_opt = np.unravel_index(np.argmin(DV_total), DV_total.shape)
print("Minimum after optimization", min_opt)
print("Dates of minimum value after optimization")
print("Departure Date:", X[0, position_min_opt[0], 0], "Time of Flight 1:", ↵
      ↵ Y[position_min_opt[1], 0, 0], "Time of Flight 2:", Z[0, 0, ↵
      ↵ position_min_opt[2]])
```

Tanto el código desarrollado como su validación pueden consultarse en los ANEXOS B y C ya referidos, y a continuación se expone un diagrama de flujo para reflejar la relación establecida entre las funciones que definidas.

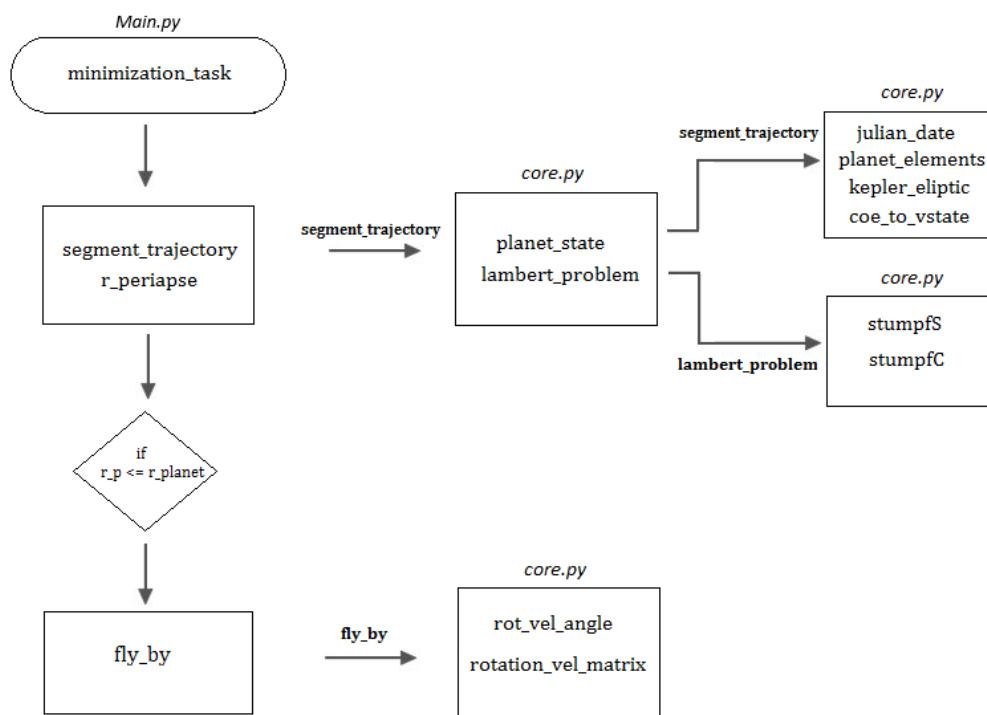


Figura 6.1: Diagrama de flujo que relaciona las funciones implementadas.

7. Caso Práctico

Es cierto que a pesar de que ya se cerró la etapa de las grandes misiones, y que las agencias espaciales en las últimas décadas han optado por misiones menos llamativas, la exploración espacial ha continuado su creciente experiencia en misiones interplanetarias no sólo a los planetas internos más próximos, sino también a los planetas exteriores e incluso asteroides.

Actualmente, debido a la proximidad existente entre la Tierra y Marte, el número de misiones con este destino están copando los objetivos de las agencias espaciales y entidades privadas. Sin embargo, en el contexto que aquí es de interés, existen también una serie de misiones a planetas interiores y exteriores que emplean maniobras de asistencia gravitacional en otros cuerpos del Sistema Solar.

Un ejemplo es la **sonda Juno** lanzada en dirección a Júpiter en Agosto de 2011, y que realizó un **fly-by** sobre la Tierra con el objetivo de llegar a su destino en Julio de 2016.

En esta dirección se enfoca este último capítulo: el diseño de una misión interplanetaria a un cuerpo exterior empleando una maniobra asistida por gravedad en un planeta intermedio.

Empleando el código desarrollado, el diseño de estas misiones consistirá en elegir tanto aquellas trayectorias como la fecha de partida y los tiempos de vuelo que minimizan el impulso ΔV_{total} necesario.

En la sección 7.1 se llevará a cabo un análisis completo de la **misión Tierra-Venus-Júpiter**, donde además de encontrar una solución, se analizarán otros parámetros que pueden resultar de interés. A continuación, en la sección 7.2 se compararán diferentes misiones a cuerpos del Sistema Solar, empleando fly-bys en diferentes planetas intermedios.

7.1. Tierra - Venus - Júpiter

A lo largo de la historia de la exploración espacial son múltiples las misiones que emplearon una maniobra de asistencia gravitacional en Venus: las misiones **Mariner**, **Galileo** o **Cassini-Huygens** son uno de los muchos ejemplos.

Para el cálculo y optimización de esta trayectoria, el código buscará la solución óptima dentro del siguiente rango de fechas establecidas:

- Ventana de inicio de la misión: 04 de Julio de 2021 hasta 11 de Marzo de 2022.
- Rango de tiempo de vuelo T_{tof1} para la trayectoria entre Tierra y Venus: la transferencia puede ser realizada entre 80 y 325 días desde la fecha de partida.

- Rango de tiempo de vuelo T_{tof2} para la trayectoria entre Venus y Júpiter: la transferencia puede ser realizada entre 650 y 950 días desde la salida del fly-by en Venus.

Con las condiciones iniciales ya fijadas y una vez todo el código fue ejecutado, a continuación se expondrá la solución encontrada, apoyada en una serie de gráficos que permitirán una mejor visualización de la misión.

7.1.1. Trayectoria Tierra - Venus. Porkchop plots

Una herramienta interesante a la hora de diseñar una misión interplanetaria es la conocida como **pork-chop plot**, que representa curvas de nivel, habitualmente del **parámetro C_3** , en función de un conjunto de fechas de partida y llegada al planeta de destino.

El C_3 permite comparar diferentes trayectorias en **términos energéticos**, pues es una medida de la **energía por unidad de masa**, obtenida como $C_3 = v_\infty^2$.

Sin embargo, el parámetro que es de interés para el estudio que aquí se lleva a cabo es la **velocidad hiperbólica excedente v_∞** en la frontera de la gravisfera terrestre, que definirá el $\Delta V_{departure}$ requerido para iniciar la transferencia.

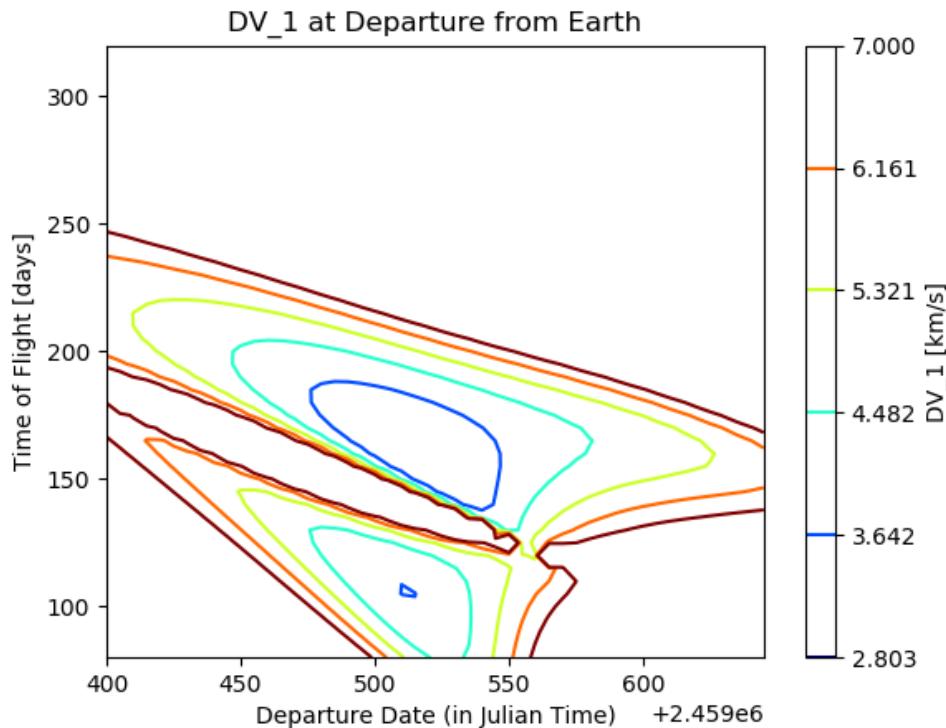


Figura 7.1: Pork-chop plot de v_∞ para la transferencia Tierra-Venus

Este parámetro puede representarse de igual manera recurriendo a un pork-chop plot (Fi-

gura 7.1). De estas gráficas pueden extraerse **ventanas de lanzamiento** compatibles con la capacidad de los actuales vehículos de lanzamiento.

En un pork-chop plot las zonas de **mínima energía**, es decir, aquellas donde el ΔV_{dep} será mínimo, son aquellas que se encuentran hacia el interior de las curvas de nivel.

De forma general, se distinguen dos zonas de mínima energía, que se corresponden con las denominadas **trayectorias de Tipo I**, donde los tiempos de vuelo son menores y el ángulo entre los vectores de salida y llegada es menor que 180° , y las **trayectorias de Tipo II**, donde los tiempos de vuelo son mayores y el ángulo entre los vectores salida y llegada es mayor que 180° .

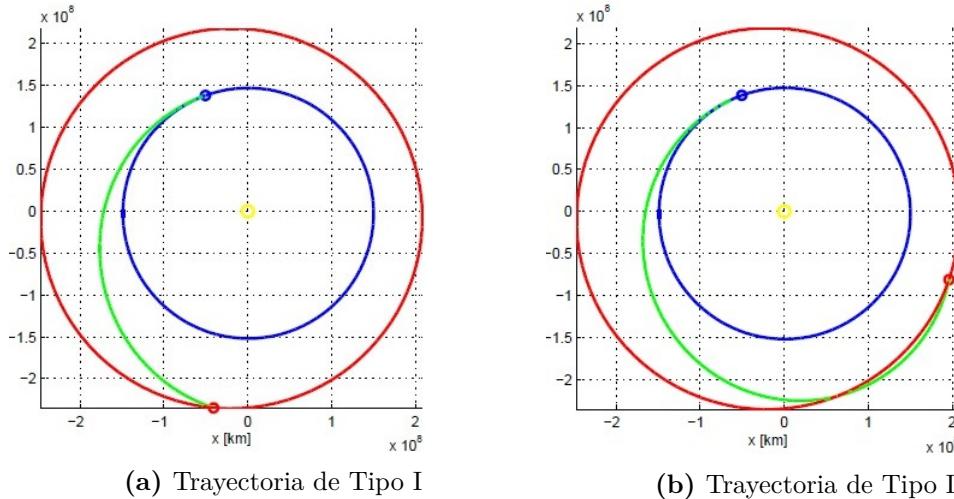


Figura 7.2: Tipos de trayectorias. Figurada tomada de (Fernández (2014)).

De acuerdo a la Figura 7.1, las transferencias que implican el mínimo ΔV_{dep} para este subsegmento tienen lugar para fechas de partida comprendidas entre el 12 de Septiembre y 21 de Diciembre de 2021, y para tiempos de vuelo de entorno a 120-200 días.

Sin embargo, una transferencia óptima entre Tierra y Venus, no implica que esta trayectoria sea la óptima a la hora de considerar la misión en su conjunto (Tierra-Venus-Júpiter). Para ello, es necesario encontrar una solución de compromiso entre los diferentes impulsos a proporcionar durante el fly-by y a la llegada a Júpiter.

7.1.2. Fly-by en Venus y llegada a Júpiter

En este caso, la pork-chop plot es poco útil, pues habría que fijar una serie de valores de partida y llegada, lo cuál no es posible, puesto que habrá un amplio rango de posibilidades en función de qué fecha quede definida como partida para la trayectoria Tierra-Venus.

Dado que las condiciones inciales y finales del fly-by vienen pre-definidas por los segmentos heliocéntricos Tierra-Venus y Venus-Júpiter para todas las posibles combinaciones de fechas de partida T_0 y tiempos de vuelo T_{tof1} y T_{tof2} , el código generará una malla de valores para el parámetro ΔV_{flyby} .

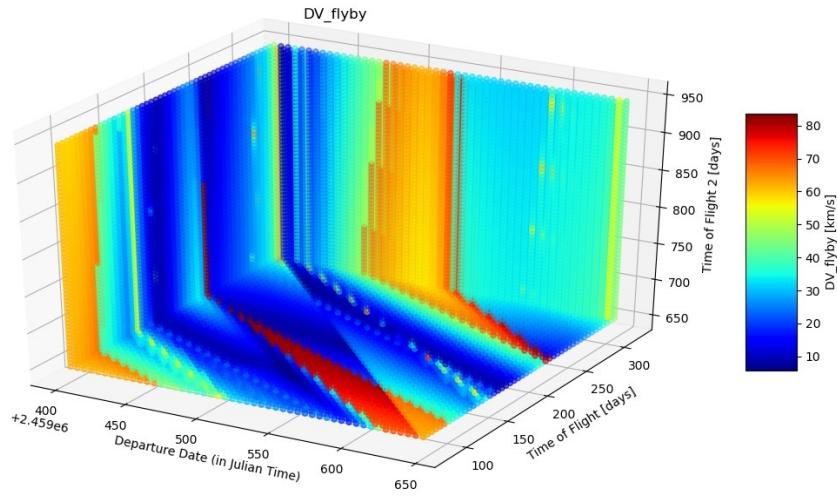


Figura 7.3: Mallado que representa la magnitud ΔV_{flyby} para cada posible trayectoria

En este gráfico (Figura 7.3), cada punto representa la **magnitud del impulso** ΔV_{flyby} que es necesario proporcionar durante la misión de fechas correspondientes a las coordenadas (T_0 , T_{tof1} , T_{tof2}) de forma que la nave pase de la transferencia definida entre la Tierra y Venus a la transferencia deseada entre Venus y Júpiter.

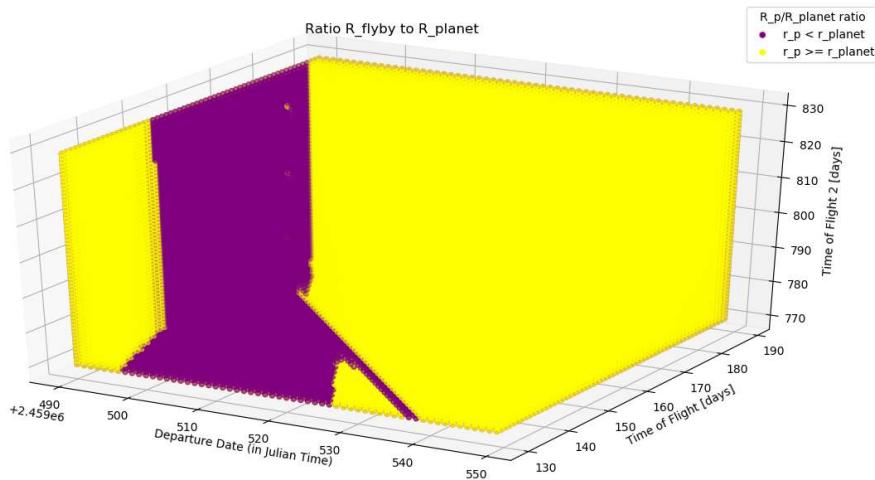
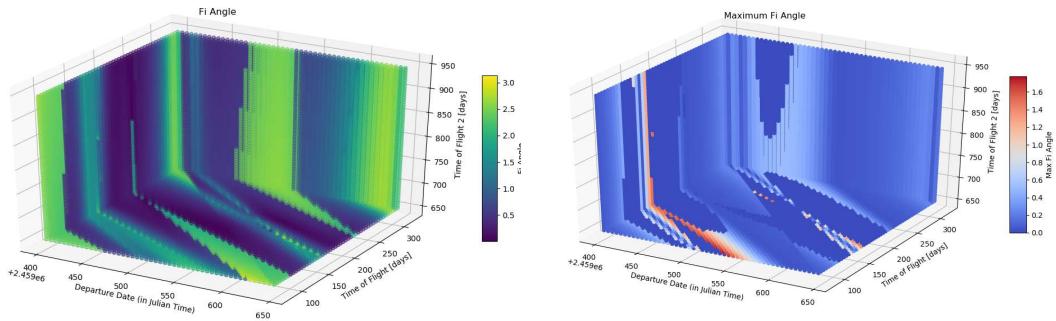


Figura 7.4: Ratio $r_p/r_{planeta}$ generado durante la maniobra de fly-by

Según lo mencionado en la sección 5.1, la forma en que ΔV_{flyby} era calculado dependía de si el radio de pericentro del fly-by r_p era mayor o menor que el radio planetario.



(a) Ángulo φ que rotaría el vector velocidad (b) Magnitud máxima del ángulo φ rotado durante la maniobra de asistencia

Figura 7.5: Variación del ángulo φ durante la maniobra de asistencia

La Figura 7.4 muestra el ratio r_p / r_{Venus} , de forma que en función de si este es mayor o menor que cero, el fly-by bien rotará el vector velocidad el ángulo φ definido por las condiciones iniciales y finales fijadas por los subsegmentos heliocéntricos, o, en cambio, rotará este vector hasta un ángulo φ_{max} (Figura 7.5).

Por último, tras la llegada a Júpiter, y el consecuente cálculo del impulso ΔV_{arr} , es posible generar una malla cuyos puntos representan la magnitud del ΔV_{total} necesario para cada posible solución de la trayectoria Tierra-Venús-Júpiter (Figura 7.6).

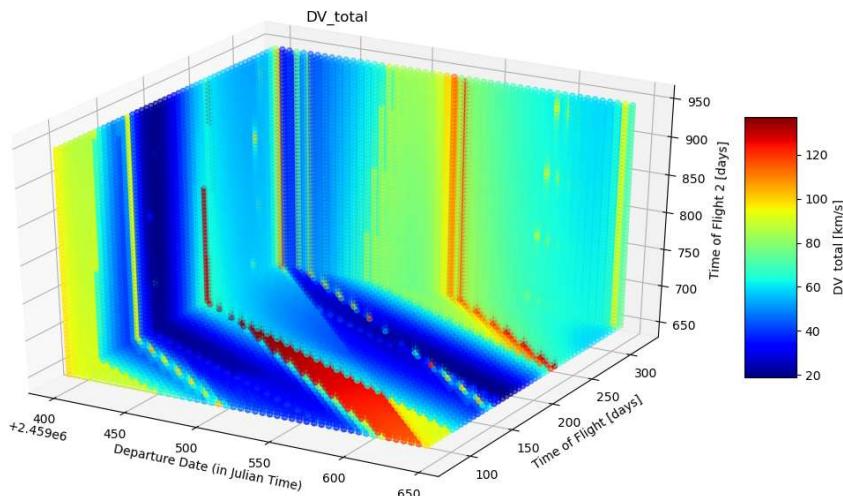


Figura 7.6: Magnitud ΔV_{total} para la misión Tierra-Venús-Júpiter

7.1.3. Búsqueda del mínimo ΔV_{total} para la misión Tierra - Venus - Júpiter

En la búsqueda del valor mínimo del parámetro ΔV_{total} , la primera iteración devuelve un resultado que servirá como aproximación inicial en la búsqueda de una mejor solución.

Esta aproximación para la misión Tierra-Venus-Júpiter puede encontrarse en la tabla 7.1.

En base a la solución encontrada, se refina el mallado en torno a las fechas obtenidas y se procede a realizar una nueva iteración, cuyas gráficas para una mejor visualización aparecen en el ANEXO A.

Buscando ahora un nuevo resultado, se encontrará una solución que minimiza el impulso ΔV_{total} para la trayectoria Tierra-Venus-Júpiter. Finalmente, la fecha de partida y los tiempos de vuelo que hacen mínimo este ΔV_{total} se presentan en la Tabla 7.1.

Tabla 7.1: Resultados para una misión Tierra-Venus-Júpiter

	T_0	T_{tof1}	T_{tof2}	ΔV_{total}
1a Iteración	01/11/2021	160 días	800 días	18.93108 km/s
2a Iteración	03/11/2021	159 días	807 días	18.91859 km/s

7.2. Otros Itinerarios

A continuación se presentan otras misiones estudiadas con el objetivo de estimar qué trayectorias pueden resultar más propicias a la hora de emprender una misión a planetas exteriores.

Las misiones estudiadas siguen algunas de las trayectorias más habituales en la exploración espacial, como aquellas que emplean fly-bys sobre la misma Tierra, u otras que emplean una transferencia directa a Júpiter para llegar a los límites del Sistema Solar.

A continuación se muestran los resultados obtenidos en las tablas que siguen:

Tabla 7.2: Primera aproximación para ΔV_{total}

Iteracion	T_0	T_{tof1} [días]	T_{tof2} [días]	ΔV_{total} [km/s]
Tierra-Tierra-Júpiter	29/06/2021	345	795	15.26725
Tierra-Marte-Júpiter	18/08/2020	405	795	19.52629
Tierra-Venus-Neptuno	7/10/2021	170	4995	23.12813
Tierra-Tierra-Neptuno	9/06/2021	330	4995	20.52637
Tierra-Júpiter-Neptuno	26/04/2021	550	3798	31.68201

Tabla 7.3: Optimización de ΔV_{total}

Iteracion	T_0	T_{tof1} [días]	T_{tof2} [días]	ΔV_{total} [km/s]
Tierra-Tierra-Júpiter	27/07/2021	320	824	15.08249
Tierra-Marte-Júpiter	20/08/2020	407	824	18.80089
‘ Tierra-Venus-Neptuno	5/10/2021	170	5024	23.00408
Tierra-Tierra-Neptuno	fecha	fecha	fecha	delta
Tierra-Júpiter-Neptuno	25/04/2021	520	3827	30.31377

De acuerdo a estos resultados, parece claro que las maniobras de asistencia por gravedad en la propia Tierra son las que proporcionan mejores efectos a la hora de plantear trayectorias a planetas exteriores.

Esta conclusión está en línea con lo visto en las últimas misiones de exploración espacial, en las que han predominado las maniobras asistidas por gravedad en la Tierra, también conocida como maniobras **EGA** (Earth Gravity Assist).

Esto es así no sólo por la mayor masa de la Tierra con respecto a Marte o Venus, si no también por la disponibilidad de ventanas de lanzamiento óptimas de forma más periódica, y en menor medida, también por las limitaciones de los actuales vehículos de lanzamiento, los cuáles limitan la posibilidad de una transferencia inicial directa hacia planetas exteriores de mayor masa donde llevar a cabo este tipo de maniobras.

A pesar de lo aquí visto, en la práctica, este tipo de misiones emplean no sólo una asistencia gravitacional, sino que utilizan diferentes combinaciones de asistencias que permiten obtener un ΔV_{total} mucho mayor que realizando una única maniobra de fly-by en cualquiera de los planetas del Sistema Solar.

Al final, el diseño de una misión interplanetaria puede asemejarse a un juego de billar, tratando de encontrar la combinación óptima para cumplir con el objetivo, sea cuál sea la combinación idónea.

8. Conclusiones y Trabajos Futuros

8.1. Conclusiones

Finalmente, en este capítulo se recogen las conclusiones extraídas a lo largo de esta memoria:

- Las maniobras EGA resultan ser las que mejores resultados devuelven si un único fly-by hacia planetas exteriores es requerido. Sin embargo, no sólo son útiles en cuanto al impulso proporcionado, si no también en cuanto a cuestiones técnicas, pues, enviar una sonda de exploración a planetas externos fríos pasando primero por Venus, mucho más cercano al Sol, puede dañar la sensible instrumentación científica.
- Marte y Venus parecen proporcionar un impulso similar a la nave en su camino en el espacio, sin embargo, en la práctica, un único fly-by llevado a cabo en estos planetas no resulta interesante. Las maniobras que emplean Venus, suelen venir acompañadas de otros fly-bys en la Tierra (maniobras VEGA, VEEGA) que resultan ser las mejores en cuanto a misiones a planetas exteriores se refiere. Por otra parte, Marte no suele ser empleado en estas misiones, pues su baja masa no proporciona un impulso elevado por si solo sin incluir otra asistencias, pero, en cambio, es muy utilizado para lograr órbitas de amplia excentricidad que permitan llegar a asteroides o cometas.
- En cuanto a Júpiter, aunque las misiones emprendidas a los límites del Sistema Solar lo emplearon como lanzadera, en la práctica no es muy útil si se quieren enviar sondas pesadas, pues alcanzarlo en un lanzamiento directo resulta complicado con las actuales lanzaderas disponibles.
- La modelización del Sistema Solar y las transferencias heliocéntricas basado en la resolución del problema de Lambert y el modelo de ajuste de cónicas con el objetivo de emprender el diseño preliminar de misiones interplanetarias proporciona unos resultados orientativos válidos de acuerdo a las validaciones que han sido llevadas a cabo.
- Un propagador planetario lineal basado en la formulación de los elementos keplerianos supone una aproximación bastante conveniente para la precisión y los tiempos de vuelo aquí estimados. Además, su implementación incurre en tiempos no muy elevados en comparación con los tiempos que una formulación más completa requeriría.
- Como última conclusión, destacar el potencial de Python como lenguaje de programación orientado a los ámbitos científico-matemático. Su variedad de librerías y sus posibilidades hacen de él una excelente herramienta a la hora de iterar problemas de esta magnitud.

8.2. Líneas de trabajo futuro

Como punto final a esta memoria, a continuación se exponen una serie de posibles tareas a incluir en un trabajo futuro que amplien y mejoren los estudios aquí realizados:

- En este trabajo se mostró como la ecuación de Kepler puede formularse de forma diferente según si el tipo de órbita considerada es elíptica o hiperbólica. Sin embargo, existe una formulación que emplea la variable universal χ , de forma que es indiferente el tipo de órbita que se esté considerando. Sería conveniente implementar esta formulación para posibles casos en los sea necesario diferenciar estos tipos de órbita.
- La implementación del método de ajuste de cónicas permite obtener aproximaciones válidas para un análisis preliminar implicando tiempos de ejecución no muy elevados. Sin embargo, existen otros métodos que no alcanzan un nivel computacional demasiado alto y que supondrían reducir el grado de error debido a la presencia de terceros cuerpos. Entre estos se encuentra el método conocido como **Pseudo-State Theory**. Una posible línea de trabajo, por tanto, sería implementar este método a la hora de llevar a cabo un diseño preliminar más preciso.
- En esta memoria, se ha considerado que la nave ya estaba en una órbita en el seno de la esfera de influencia terrestre, de forma que directamente se ha procedido a calcular qué condiciones serían las mejores para emprender la trayectoria hacia el planeta en el que realizar el fly-by. Una amplia línea de mejora podría incluir llevar a cabo una estimación de las ventanas de lanzamiento más óptimas para iniciar la misión, fijando, por ejemplo, una base de lanzamiento concreta, de forma que la trayectoria hasta abandonar la atmósfera planetaria sea tenida en cuenta.
- En cuanto a la optimización llevada a cabo se podrían plantear una serie de mejoras interesantes. En primer lugar, es posible sustituir el método basado en un re-ajuste del mallado en torno a una primera solución por un optimizador que emplee esta solución como semilla del mismo. Por otra parte, a lo largo de la memoria, los itinerarios planteados venían fijados previamente por los planetas introducidos como parámetros, por lo que una línea de mejora podría suponer generar aleatoriamente una serie de combinaciones planetarias y que la solución de la iteración devuelva el itinerario planetario que supone los mínimos requerimientos.
- Ya se ha visto que las trayectorias más efectivas y empleadas en la práctica son aquellas que emplean más de un fly-by en diferentes cuerpos (Venus-Tierra, Venus-Tierra-Tierra, incluso Venus-Tierra-Venus-Tierra), por lo que una importante mejora sería incorporar la posibilidad de llevar a cabo más de un fly-by que mejore los resultados.
- Por último, se podría plantear un diseño más preciso de la trayectoria, integrando numéricamente métodos que permitan incluir las perturbaciones generadas por la presencia de terceros o más cuerpos.

Bibliografía

- Anónimo. (2010). *La anomalía excéntrica*. Descargado 20/08/2019, de <https://esacademic.com/dic.nsf/eswiki/87337>
- Curtis, H. (2005). *Orbital mechanics for engineering students*. Elsevier.
- Fernández, A. L. (2014). *Análisis y diseño de una misión interplanetaria a un planeta exterior*, Trabajo Final de Grado . Universidad de Sevilla.
- Gim, J. D. (2011). *The superior Lambert algorithm*. Descargado de <https://www.amostech.com/TechnicalPapers/2011/Poster/DER.pdf>
- Kemble, S. (2006). *Interplanetary mission analysis and design*. Springer Science-Business Media.
- Labunsky, A. V., Papkov, O. V., y Sukhanov, K. G. (1998). *Multiple gravity assist interplanetary trajectories*. CRC Press.
- Montenbruck, O., y Gill, E. (2000). *Satellite orbits: Models, methods and applications*. Springer Science-Business Media.
- NASA. (1998). *Interplanetary mission design handbook: Earth-to-Mars mission opportunities and Mars-to-Earth return opportunities 2009-2024*. Createspace Independent Pub.
- Ng, E. W. (1979). A general algorithm for the solution of Kepler's equation for elliptic orbits. *Celestial Mechanics*, 20, 243-249.
- Petropoulos, A. E., Longuski, J. M., y Bonfiglio, E. P. (2000). Trajectories to Jupiter via gravity assists from Venus, Earth, and Mars. *Journal of Spacecraft and Rockets*, 37(6), 776-787.
- Shank, B. S. (2013). *Development of an optimized Lambert problem solver for targeting elliptical orbits* . Pennsylvania State University.
- Siddiqi, A. A. (2018). *Beyond earth: A chronicle of deep space exploration*. National Aeronautics and Space Administration, NASA.
- Standish, E. M., y Williams, J. G. (s.f.). *Chapter 8. Orbital ephemerides of the Sun, Moon and planets*. Descargado de <ftp://ssd.jpl.nasa.gov/pub/eph/planets/ioms/ExplSupplChap8.pdf>
- Vallado, D. A., y McClain, W. D. (2013). *Fundamentals of astrodynamics and applications*. Microcosm Press.

Varios. (2013). *An example of Kepler's equation solver*. Descargado 20/08/2019, de <https://smallsts.org/2013/01/17/keplers-equation-solver/>

Williams, D. R. (s.f.). *Planetary fact sheet - metric*. Descargado 10/07/2019, de <https://nssdc.gsfc.nasa.gov/planetary/factsheet/>

A. Anexo I - Gráficas Capítulo 7

A.1. Fechas y tiempos de vuelo fijados para las misiones

Tabla A.1: Fechas y tiempos de vuelo para la misión Tierra-Tierra-Júpiter

	Desde	Hasta
Misión Tierra-Tierra-Júpiter		
T_0	7/09/2020	4/07/2021
T_{tof1} [días]	300	600
T_{tof2} [días]	550	800
Misión Tierra-Marte-Júpiter		
T_0	10/04/2020	16/12/2020
T_{tof1} [días]	160	420
T_{tof2} [días]	500	800
Misión Tierra-Venus-Neptuno		
T_0	4/07/2021	11/03/2022
T_{tof1} [días]	90	300
T_{tof2} [días]	4000	5000
Misión Tierra-Tierra-Neptuno		
T_0	7/09/2020	4/07/2021
T_{tof1} [días]	300	600
T_{tof2} [días]	4000	5000
Misión Tierra-Júpiter-Neptuno		
T_0	21/12/2020	14/07/2021
T_{tof1} [días]	550	850
T_{tof2} [días]	3000	3800

A.2. Gráficas Misiones

A.2.1. Gráficas Optimizadas Tierra-Venus-Júpiter

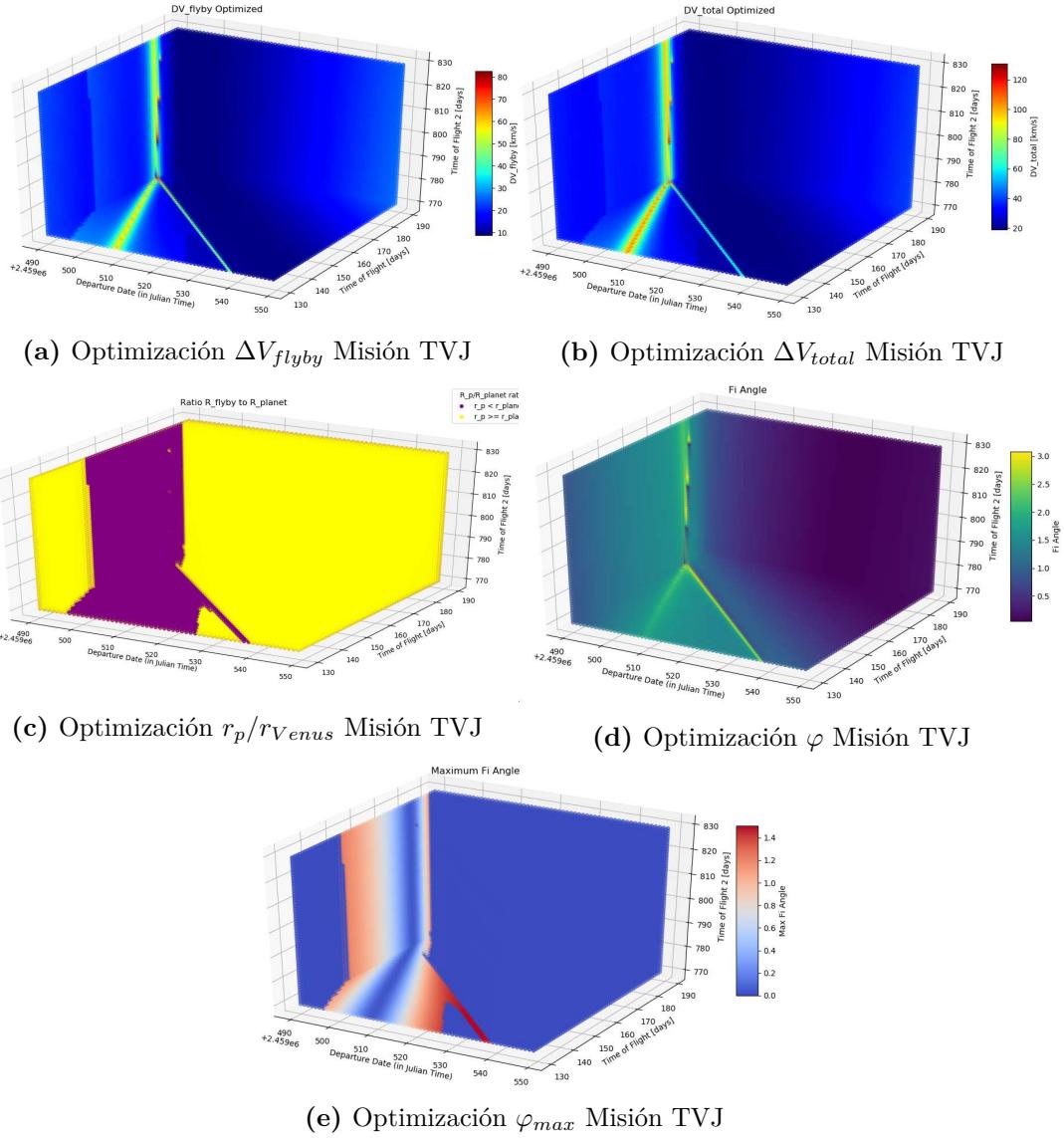


Figura A.1: Gráficas de la optimización de la misión Tierra-Venus-Júpiter

A.2.2. Gráficas Tierra-Tierra-Júpiter

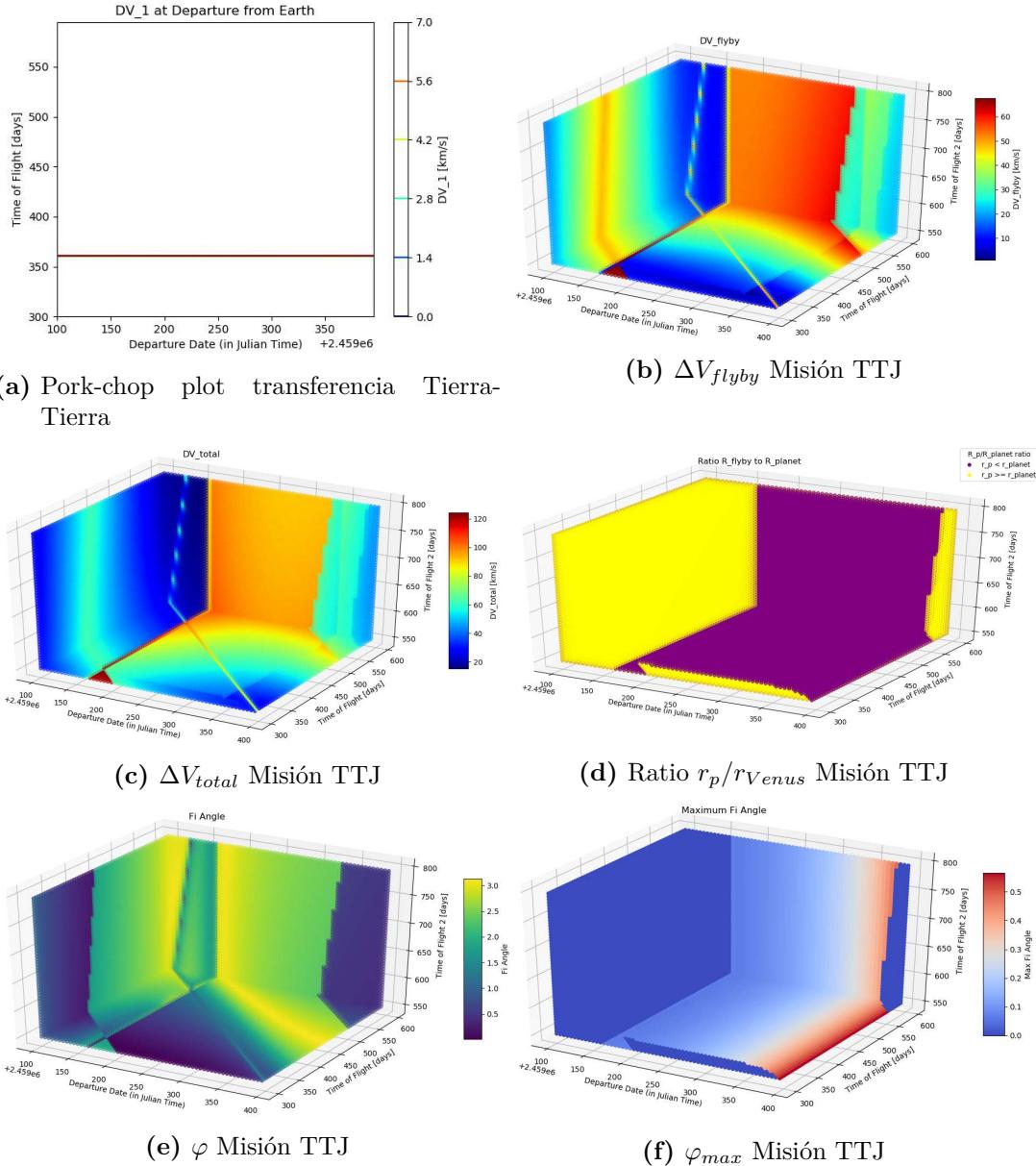


Figura A.2: Gráficas Misión Tierra-Tierra-Júpiter

A.2.3. Gráficas Tierra-Marte-Júpiter

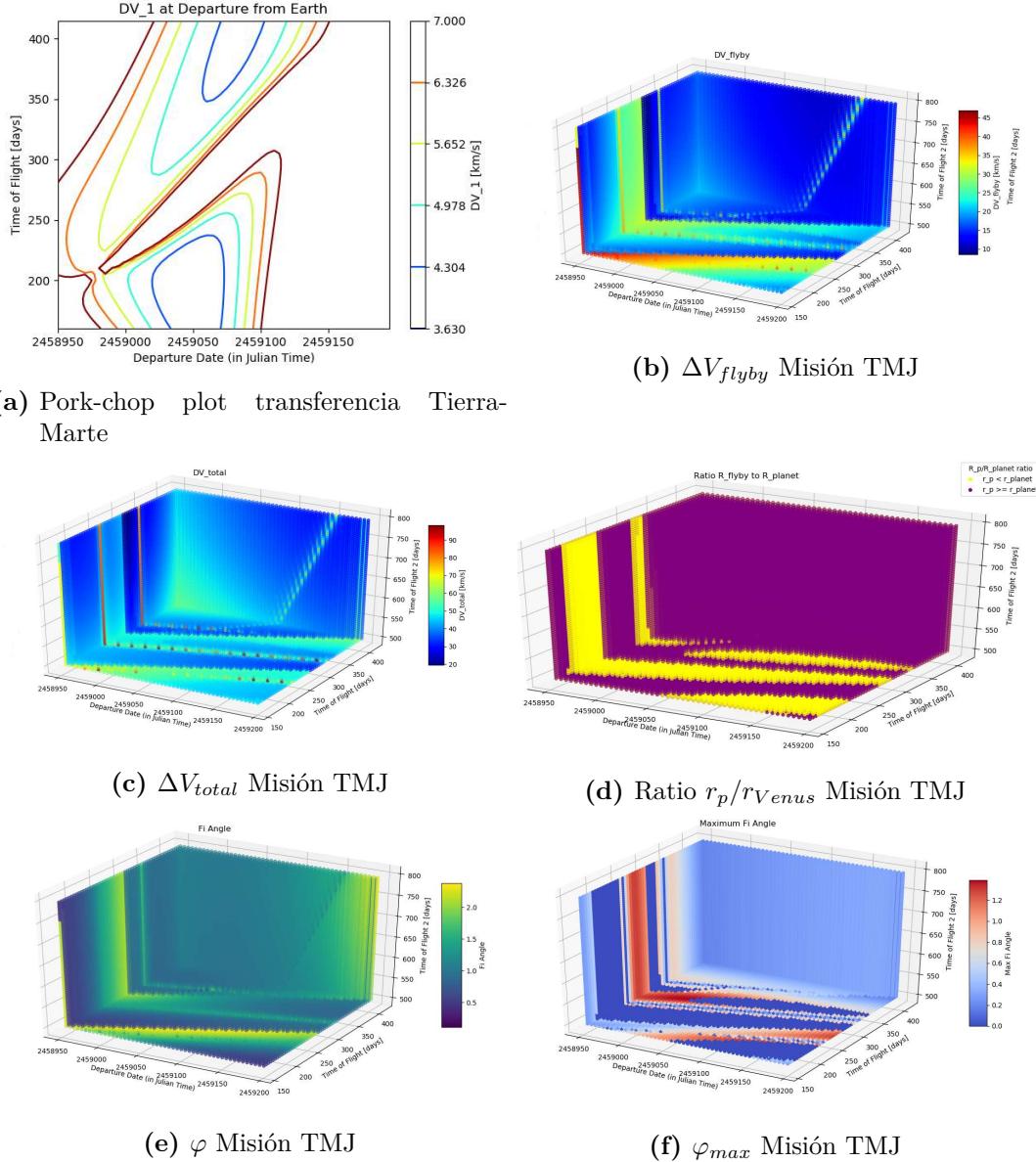


Figura A.3: Gráficas Misión Tierra-Marte-Júpiter

A.2.4. Gráficas Tierra-Venus-Neptuno

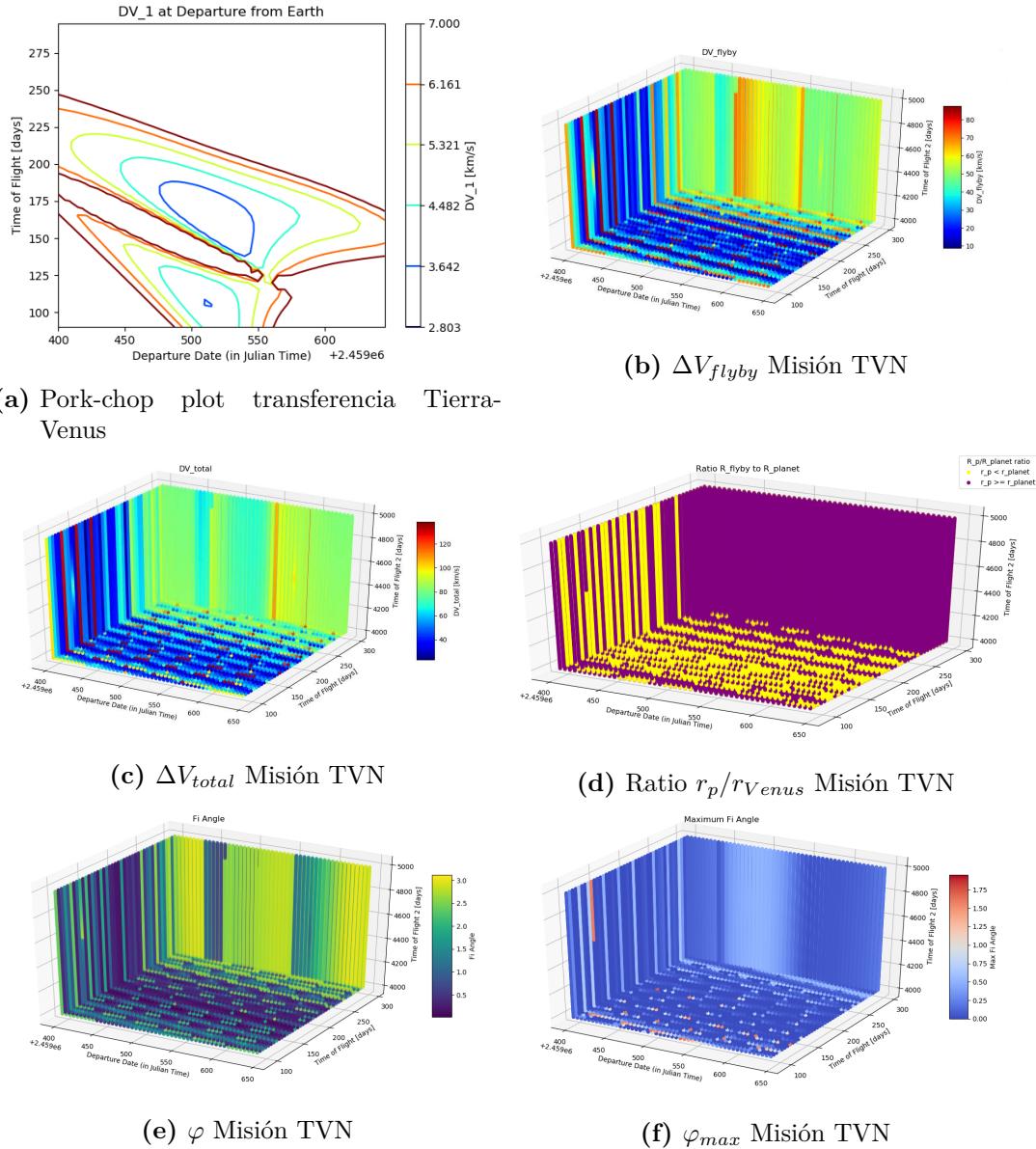


Figura A.4: Gráficas Misión Tierra-Venus-Neptuno

A.2.5. Gráficas Tierra-Tierra-Neptuno

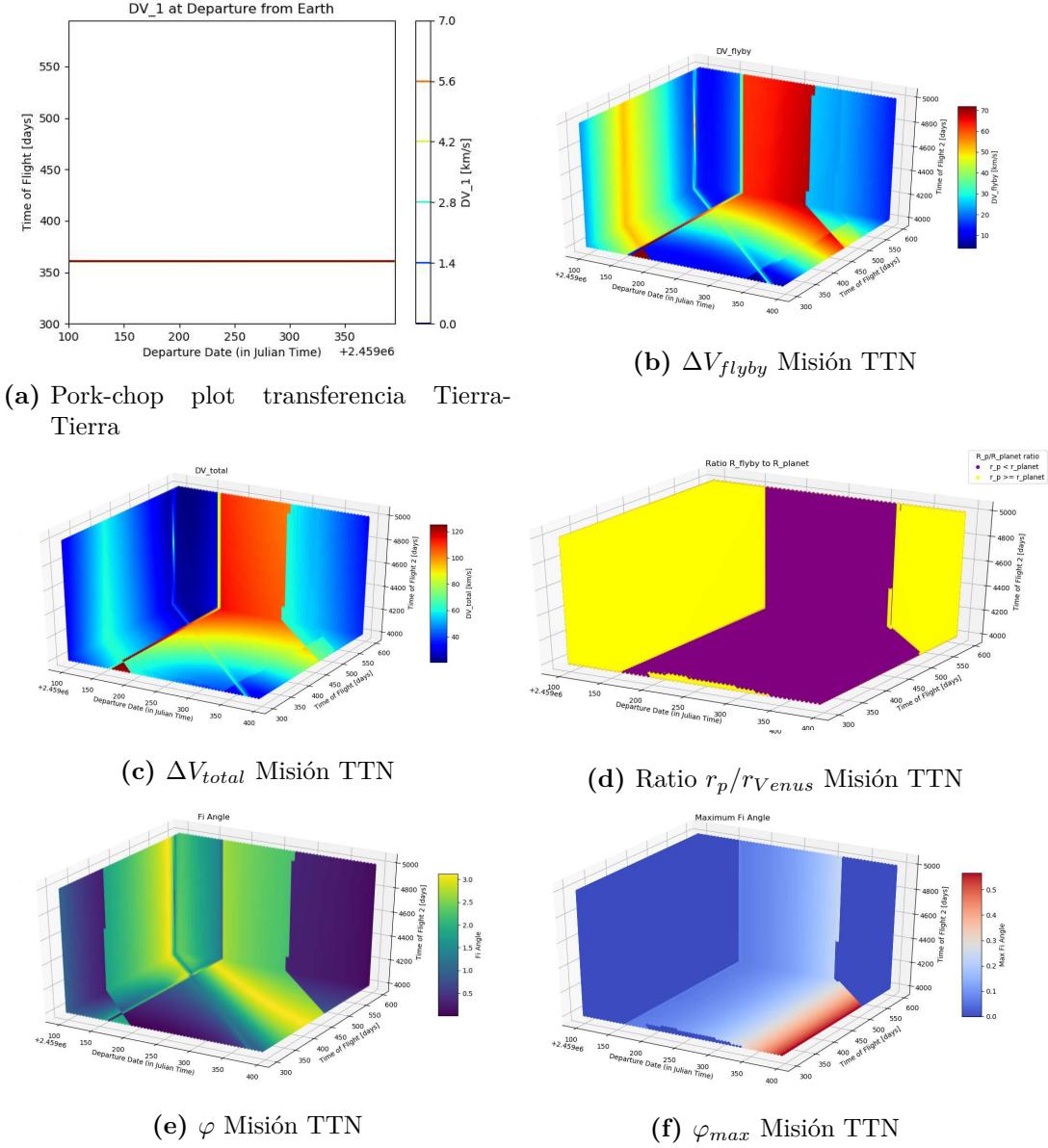


Figura A.5: Gráficas Misión Tierra-Tierra-Neptuno

A.2.6. Gráficas Tierra-Júpiter-Neptuno

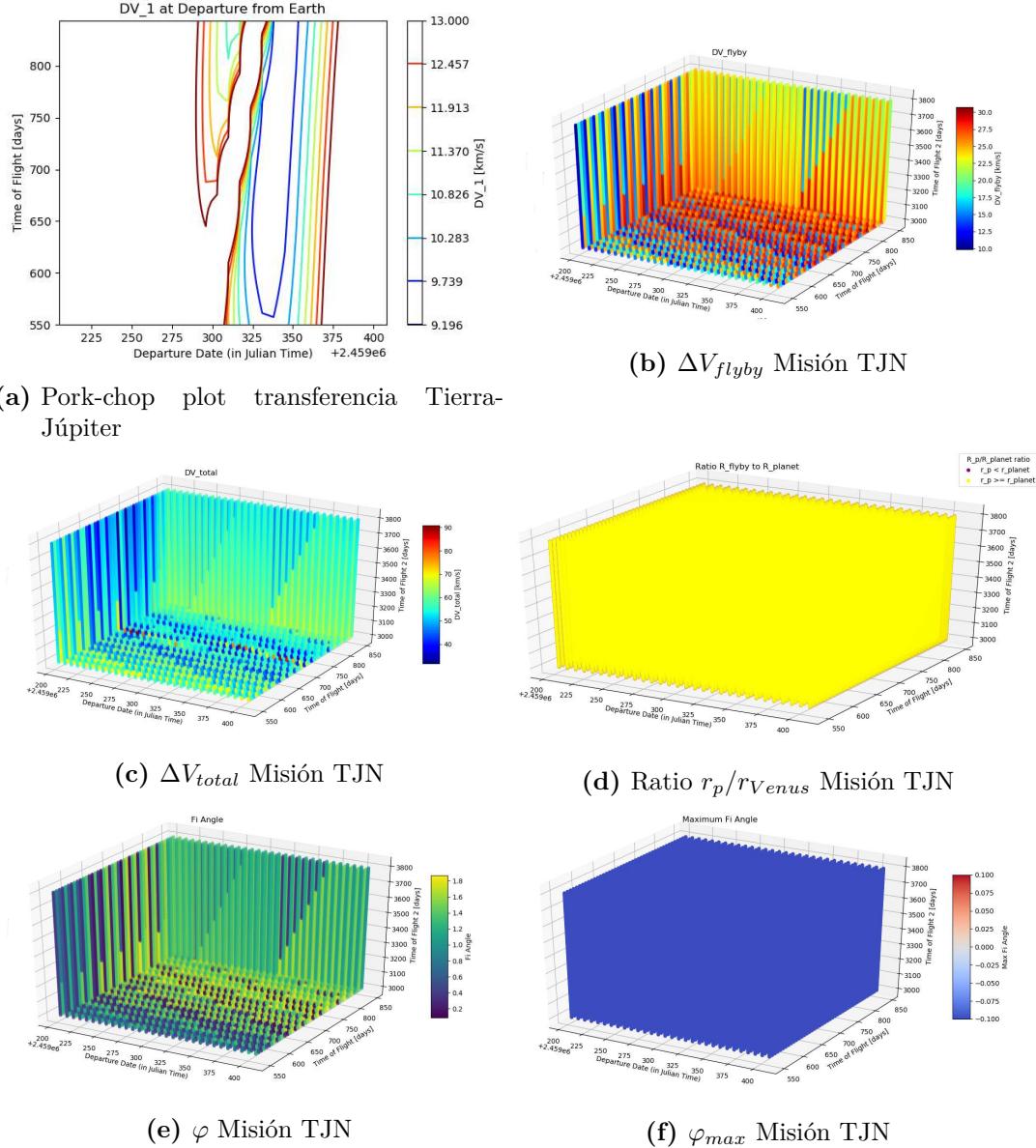


Figura A.6: Gráficas Misión Tierra-Júpiter-Neptuno

A.3. Optimización ΔV_{total}

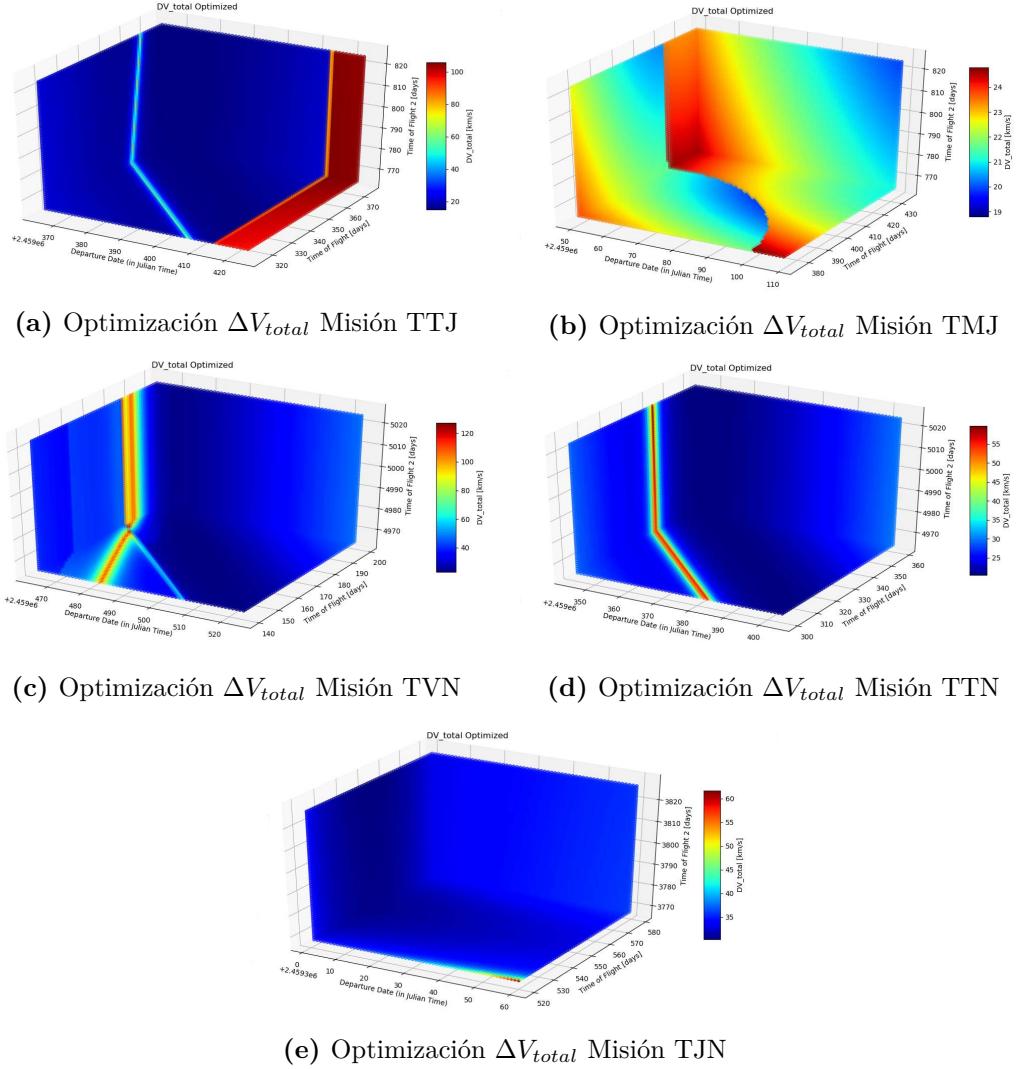


Figura A.7: Optimización Δv_{total}

B. Anexo II - Código

Este anexo incluye el **código completo** que ha sido desarrollado para llevar a cabo el análisis preliminar de misiones interplanetarias.

Cada sección se corresponde a un archivo de extensión **.py**, de forma que el archivo principal no tenga una extensión demasiado larga y compleja.

Las sección B.1 define, en un único archivo, todas las **constantes** que serán necesarias a lo largo del código. La sección B.2 incluye el **cuerpo del programa**, donde son desarrolladas todas las funciones necesarias para llevar a cabo la resolución del problema expuesto: funciones para transformar coordenadas cartesianas en elementos orbitales y viceversa, la función que resuelve el problema de Lambert, una función que devolverá el vector de estado planetario en un momento dado... Por último, la sección B.3 incluye la **parte principal del programa**, donde son definidos los bucles que permitirán evaluar el problema para las diferentes combinaciones de tiempos posibles.

La sección B.4 expone el fichero que fue empleado para **validar** el código implementado.

Dado que los valores para una misma constante difieren en función de la solución de referencia a la que se atiende a la hora de llevar a cabo la validación, el fichero *validation.py* simplemente ejecuta las funciones y devuelve unos valores que han de ser contrastados con las referencias empleadas.

Esta validación será expuesta en el Anexo C.

B.1. planet_consts.py

```
1 """
2 SUN Consts -----
3 """
4
5 # General Consts
6
7 M_Sun = 1.98 * 10 ** 30 # Mass [kg]
8 K_Sun = 1.327124 * 10 ** 11 # Gravitational Parameter [km3/s2]
9
10 """
11 MERCURY Consts -----
12 """
13
14 # Keplerian Elements
15
16 ao_M = 0.38709843 # [au]
```

```

17 da_M = 0.0 # [au/cty]
18 eo_M = 0.20563661 # [rad]
19 de_M = 0.00002123 # [rad/cty]
20 Io_M = 7.00559432 # [°]
21 dI_M = -0.00590158 # [°/cty]
22 Lo_M = 252.25166724 # [°]
23 dL_M = 149472.67486623 # [°/cty]
24 wo_M = 77.45771895 # [°]
25 dw_M = 0.15940013 # [°/cty]
26 Oo_M = 48.33961819 # [°]
27 dO_M = -0.12214182 # [°/cty]
28 b_M = 0.0 # [°/cty2]
29 c_M = 0.0 # [°]
30 s_M = 0.0 # [°]
31 f_M = 0.0 # [°/cty]
32
33 Mercury = [ao_M, da_M, eo_M, de_M, Io_M, dI_M, Lo_M, dL_M, wo_M, dw_M, Oo_M, ←
   ↘ dO_M, b_M, c_M, s_M, f_M]
34
35 # General Consts
36
37 m_Mercury = 3.30 * 10 ** 23 # Mass [kg]
38 r_Mercury = 2439.7 # Equatorial Radius [km]
39 K_Mercury = 0.022032 * 10 ** 6 # Gravitational Parameter [km3/s2]
40
41 """
42 VENUS Consts -----
43 """
44
45 # Keplerian Elements
46
47 ao_V = 0.72332102 # [au]
48 da_V = -0.00000026 # [au/cty]
49 eo_V = 0.00676399 # [rad]
50 de_V = -0.00005107 # [rad/cty]
51 Io_V = 3.39777545 # [°]
52 dI_V = 0.00043494 # [°/cty]
53 Lo_V = 181.97970850 # [°]
54 dL_V = 58517.81560260 # [°/cty]
55 wo_V = 131.76755713 # [°]
56 dw_V = 0.05679648 # [°/cty]
57 Oo_V = 76.67261496 # [°]
58 dO_V = -0.27274174 # [°/cty]
59 b_V = 0.0 # [°/cty2]
60 c_V = 0.0 # [°]
61 s_V = 0.0 # [°]
62 f_V = 0.0 # [°/cty]
63
64 Venus = [ao_V, da_V, eo_V, de_V, Io_V, dI_V, Lo_V, dL_V, wo_V, dw_V, Oo_V, ←
   ↘ dO_V, b_V, c_V, s_V, f_V]
65

```

```

60 # General Consts
61
62 m_Venus = 4.87 * 10 ** 24 # Mass [kg]
63 r_Venus = 6051.8 # Equatorial radius [km]
64 K_Venus = 0.32486 * 10 ** 6 # Gravitational Parameter [km3/s2]
65
66 """
67 EARTH Consts -----
68 """
69
70 # Keplerian Elements
71
72 ao_E = 1.00000018 # [au]
73 da_E = -0.00000003 # [au/cty]
74 eo_E = 0.01673163 # [rad]
75 de_E = -0.00003661 # [rad/cty]
76 Io_E = -0.00054346 # [°]
77 dI_E = -0.01337178 # [°/cty]
78 Lo_E = 100.46691572 # [°]
79 dL_E = 35999.37306329 # [°/cty]
80 wo_E = 102.93005885 # [°]
81 dw_E = 0.31795260 # [°/cty]
82 Oo_E = -5.11260389 # [°]
83 d0_E = -0.24123856 # [°/cty]
84 b_E = 0.0 # [°/cty2]
85 c_E = 0.0 # [°]
86 s_E = 0.0 # [°]
87 f_E = 0.0 # [°/cty]
88
89 Earth = [ao_E, da_E, eo_E, de_E, Io_E, dI_E, Lo_E, dL_E, wo_E, dw_E, Oo_E, ↪
90   ↪ d0_E, b_E, c_E, s_E, f_E]
91
92 # General Consts
93
94 m_Earth = 5.97 * 10 ** 24 # Mass [kg]
95 r_Earth = 6378.1 # Equatorial radius [km3/s2]
96 K_Earth = 0.3986 * 10 ** 6 # Gravitational Parameter [km3/s2]
97
98 """
99 MARS Consts -----
100 """
101
102 # Keplerian Elements
103
104 ao_Mars = 1.52371243 # [au]
105 da_Mars = 0.00000097 # [au/cty]
106 eo_Mars = 0.09336511 # [rad]
107 de_Mars = 0.00009149 # [rad/cty]
108 Io_Mars = 1.85181869 # [°]
109 dI_Mars = -0.00724757 # [°/cty]
110 Lo_Mars = -4.56813164 # [°]

```

```

116 dL_Mars = 19140.29934243 # [°/cty]
117 wo_Mars = -23.91744784 # []
118 dw_Mars = 0.45223625 # [°/cty]
119 Oo_Mars = 49.71320984 # [°]
120 d0_Mars = -0.26852431 # [°/cty]
121 b_Mars = 0.0 # [°/cty2]
122 c_Mars = 0.0 # [°]
123 s_Mars = 0.0 # [°]
124 f_Mars = 0.0 # [°/cty]
125
126 Mars = [ao_Mars, da_Mars, eo_Mars, de_Mars, Io_Mars, dI_Mars, Lo_Mars, dL_Mars ←
127   ↪ , wo_Mars, dw_Mars, Oo_Mars, d0_Mars,
128     b_Mars, c_Mars, s_Mars, f_Mars]
129
130 # General Consts
131
132 m_Mars = 6.42 * 10 ** 23 # Mass [kg]
133 r_Mars = 3396.2 # Equatorial Radius [km]
134 K_Mars = 0.042828 * 10 ** 6 # Gravitational Parameter [km3/s2]
135 """
136 JUPITER Consts -----
137 """
138 # Keplerian Elements
139
140 ao_J = 5.20248019 # [au]
141 da_J = -0.00002864 # [au/cty]
142 eo_J = 0.04853590 # [rad]
143 de_J = 0.00018026 # [rad/cty]
144 Io_J = 1.29861416 # [°]
145 dI_J = -0.00322699 # [°/cty]
146 Lo_J = 34.33479152 # []
147 dL_J = 3034.90371757 # [°/cty]
148 wo_J = 14.27495244 # [°]
149 dw_J = 0.18199196 # [°/cty]
150 Oo_J = 100.29282654 # []
151 d0_J = 0.13024619 # [°/cty]
152 b_J = -0.00012452 # [°/cty2]
153 c_J = 0.06064060 # []
154 s_J = -0.35635438 # [°]
155 f_J = 38.35125000 # [°/cty]
156
157 Jupiter = [ao_J, da_J, eo_J, de_J, Io_J, dI_J, Lo_J, dL_J, wo_J, dw_J, Oo_J, ←
158   ↪ d0_J, b_J, c_J, s_J, f_J]
159
160 # General Consts
161
162 m_Jupiter = 1.89 * 10 ** 27 # Mass [kg]
163 r_Jupiter = 71492 # Equatorial radius [km]
164 K_Jupiter = 126.687 * 10 ** 6 # Gravitational Parameter [km3/s2]
165

```

```

165 """
166 SATURN Consts -----
167 """
168
169 # Keplerian Elements
170
171 ao_S = 9.54149883 # [au]
172 da_S = -0.00003065 # [au/cty]
173 eo_S = 0.05550825 # [rad]
174 de_S = -0.00032044 # [rad/cty]
175 Io_S = 2.49424102 # [°]
176 dI_S = 0.00451969 # [°/cty]
177 Lo_S = 50.07571329 # [°]
178 dL_S = 1222.11494724 # [°/cty]
179 wo_S = 92.86136063 # [°]
180 dw_S = 0.54179478 # [°/cty]
181 Oo_S = 113.63998702 # [°]
182 d0_S = -0.25015002 # [°/cty]
183 b_S = 0.00025899 # [°/cty2]
184 c_S = -0.13434469 # [°]
185 s_S = 0.87320147 # [°]
186 f_S = 38.35125000 # [°/cty]
187
188 Saturn = [ao_S, da_S, eo_S, de_S, Io_S, dI_S, Lo_S, dL_S, wo_S, dw_S, Oo_S, ←
189     ↢ d0_S, b_S, c_S, s_S, f_S]
190
191 # General Consts
192
193 m_Saturn = 5.68 * 10 ** 26 # Mass [kg]
194 r_Saturn = 60268 # Equatorial radius [km]
195 K_Saturn = 37.931 * 10 ** 6 # Gravitational Parameter [km3/s2]
196 """
197 URANUS Consts -----
198 """
199
200 # Keplerian Elements
201
202 ao_U = 19.18797948 # [au]
203 da_U = -0.00020455 # [au/cty]
204 eo_U = 0.04685740 # [rad]
205 de_U = -0.00001550 # [rad/cty]
206 Io_U = 0.77298127 # [°]
207 dI_U = -0.00180155 # [°/cty]
208 Lo_U = 314.20276625 # [°]
209 dL_U = 428.49512595 # [°/cty]
210 wo_U = 172.43404441 # [°]
211 dw_U = 0.09266985 # [°/cty]
212 Oo_U = 73.96250215 # [°]
213 d0_U = 0.05739699 # [°/cty]
214 b_U = 0.00058331 # [°/cty2]

```

```

215 c_U = -0.97731848 # [°]
216 s_U = 0.17689245 # [°]
217 f_U = 7.67025000 # [°/cty]
218
219 Uranus = [ao_U, da_U, eo_U, de_U, Io_U, dI_U, Lo_U, dL_U, wo_U, dw_U, Oo_U, ←
    ↪ dO_U, b_U, c_U, s_U, f_U]
220
221 # General Consts
222
223 m_Uranus = 8.68 * 10 ** 25 # Mass [kg]
224 r_Uranus = 2559 # Equatorial radius [km]
225 K_Uranus = 5.7940 * 10 ** 6 # Gravitational Parameter [km3/s2]
226
227 """
228 NEPTUNE Consts -----
229 """
230
231 # Keplerian Elements
232
233 ao_N = 30.06952752 # [au]
234 da_N = 0.00006447 # [au/cty]
235 eo_N = 0.00895439 # [rad]
236 de_N = 0.00000818 # [rad/cty]
237 Io_N = 1.77005520 # [°]
238 dI_N = 0.00022400 # [°/cty]
239 Lo_N = 304.22289287 # [°]
240 dL_N = 218.46515314 # [°/cty]
241 wo_N = 46.68158724 # [°]
242 dw_N = 0.01009938 # [°/cty]
243 Oo_N = 131.78635853 # [°]
244 dO_N = -0.00606302 # [°/cty]
245 b_N = -0.00041348 # [°/cty2]
246 c_N = 0.68346318 # [°]
247 s_N = -0.10162547 # [°]
248 f_N = 7.67025000 # [°/cty]
249
250 Neptune = [ao_N, da_N, eo_N, de_N, Io_N, dI_N, Lo_N, dL_N, wo_N, dw_N, Oo_N, ←
    ↪ dO_N, b_N, c_N, s_N, f_N]
251
252 # General Consts
253
254 m_Neptune = 1.02 * 10 ** 26 # Mass [kg]
255 r_Neptune = 24764 # Equatorial radius [km]
256 K_Neptune = 6.8351 * 10 ** 6 # Gravitational Parameter [km3/s2]
257
258 """
259 PLUTO Consts -----
260 """
261
262 # Keplerian Elements
263 ao_P = 39.48686035 # [au]

```

```

264 da_P = 0.00449751 # [au/cty]
265 eo_P = 0.24885238 # [rad]
266 de_P = 0.00006016 # [rad/cty]
267 Io_P = 17.14104260 # [°]
268 dI_P = 0.00000501 # [°/cty]
269 Lo_P = 238.96535011 # [°]
270 dL_P = 145.18042903 # [°/cty]
271 wo_P = 224.09702598 # [°]
272 dw_P = -0.00968827 # [°/cty]
273 Oo_P = 110.30167986 # [°]
274 d0_P = -0.00809981 # [°/cty]
275 b_P = -0.01262724 # [°/cty2]
276 c_P = 0.0 # [°]
277 s_P = 0.0 # [°]
278 f_P = 0.0 # [°/cty]
279
280 Pluto = [ao_P, da_P, eo_P, de_P, Io_P, dI_P, Lo_P, dL_P, wo_P, dw_P, Oo_P, ←
281     ↢ d0_P, b_P, c_P, s_P, f_P]
282
283 # General Consts
284
285 m_Pluto = 1.3 * 10 ** 22 # Mass [kg]
286 r_Pluto = 1187 # Equatorial radius [km]
287 K_Pluto = 0.000870 * 10 ** 6 # Gravitational Parameter [km3/s2]

```

B.2. core.py

```

1 import numpy as np
2 from planet_consts import *
3
4 # Consts
5 mu = K_Sun
6 tolerance = 10 ** (-8)
7 au = 1.496 * 10 ** 8
8 h_atm = 600 # [km]
9
10 # Decision parameter for Lambert's Problem
11 kind_orbit = 1 # 1: Prograde Orbit / 2: Retrograde Orbit
12
13 """
14 Some useful generic functions
15 """
16
17 def romin(r_planet):
18     return r_planet + h_atm
19
20 """
21 State Vector to Orbital Elements
22

```

```

23     h [km2 / s]
24     i [rad]
25     RA [rad]
26     w [rad]
27     theta [rad]
28     e []
29     rp [km]
30     ra [km]
31 """
32
33 def vstate_to_coe(vect1, vect2):
34     r = np.linalg.norm(vect1)
35     # v = np.linalg.norm(vect2)
36     v_rad = np.dot(vect1, vect2) / r
37
38     H = np.cross(vect1, vect2)
39     h = np.linalg.norm(H)
40
41     inc = np.arccos(H[2] / h)
42
43     N = np.cross([0, 0, 1], H)
44     n = np.linalg.norm(N)
45
46     if n != 0:
47         RA = np.arccos(N[0] / n)
48         if N[1] < 0:
49             RA = 2 * np.pi - RA
50     else:
51         RA = 0
52
53     e_vect = (1 / mu) * (np.cross(vect2, H) - mu * (vect1 / r))
54     e = np.linalg.norm(e_vect)
55
56     if n != 0:
57         w = np.arccos(np.dot(N, e_vect) / (n * e))
58         if e_vect[2] < 0:
59             w = 2 * np.pi - w
60     else:
61         w = 0
62
63     theta = np.arccos(np.dot(e_vect, vect1) / (e * r))
64     if v_rad < 0:
65         theta = 2 * np.pi - theta
66
67     T = (2.0 * np.pi / mu ** 2.0) * (h / np.sqrt(1 - e ** 2.0)) ** 3.0
68
69     return h, e, RA, inc, w, theta, T
70
71 """
72
73 Orbital Elements to State Vector

```

```

74
75 """
76
77 def coe_to_vstate(h, e, RA, inc, w, theta):
78     R_pf = (h ** 2.0 / mu) * (1 / (1 + e * np.cos(theta))) * np.array([np.cos(←
79         ↪ theta), np.sin(theta), 0.0])
80     V_pf = (mu / h) * np.array([-np.sin(theta), e + np.cos(theta), 0.0])
81
82     R3_RA = np.array([[np.cos(RA), np.sin(RA), 0],
83                       [-np.sin(RA), np.cos(RA), 0],
84                       [0, 0, 1]])
85
86     R1_inc = np.array([[1, 0, 0],
87                        [0, np.cos(inc), np.sin(inc)],
88                        [0, -np.sin(inc), np.cos(inc)]])
89
90     R3_w = np.array([[np.cos(w), np.sin(w), 0],
91                      [-np.sin(w), np.cos(w), 0],
92                      [0, 0, 1]])
93
94     conv_matrix = R3_w @ R1_inc @ R3_RA
95     conv_matrix = np.transpose(conv_matrix)
96
97     R = conv_matrix @ R_pf
98     V = conv_matrix @ V_pf
99
100    R = np.transpose(R)
101    V = np.transpose(V)
102
103    return R, V
104 """
105
106 Kepler Equation
107
108
109 """
110
111 def kepler_eliptic(e, M):
112     if M < np.pi:
113         E = M + e / 2.0
114     else:
115         E = M - e / 2.0
116
117     ratio = 1
118     while abs(ratio) > tolerance:
119         fE = E - e * np.sin(E) - M
120         df_E = 1 - e * np.cos(E)
121         ratio = fE / df_E
122         E = E - ratio
123

```

```

124     theta = 2.0 * np.arctan(np.sqrt((1.0 + e) / (1.0 - e)) * np.tan(E / 2.0))
125     if theta < 0:
126         theta = 2 * np.pi + theta
127
128     return E, theta
129
130 """
131 SOLAR SYSTEM
132 """
133 """
134 def julian_date(t_ephemeris, t): # t in days
135     t Julian = (t_ephemeris - 2451545.0) / 36525.0
136     t Julian = t Julian + (t / 36525.0)
137
138     return t Julian
139
140 """
141 """
142 """
143
144 planets_elements function will return Keplerian Elements for each planet
145
146     h_p [km2/s]
147     a_p [km]
148     e_p [rad]
149     I_p [rad]
150     L_p [rad]
151     w1_p [rad]
152     w_p [rad]
153     o_p [rad]
154     M_p [rad]
155
156 """
157
158 def planets_elements(planet, julian_time):
159     a_p = planet[0] + (planet[1] * julian_time) # [au]
160     e_p = planet[2] + (planet[3] * julian_time) # [rad]
161     I_p = planet[4] + (planet[5] * julian_time) # [°]
162     L_p = planet[6] + (planet[7] * julian_time) # [°]
163     w1_p = planet[8] + (planet[9] * julian_time) # [°]
164     O_p = planet[10] + (planet[11] * julian_time) # [°]
165
166     # As every parameter should be given in radians, all the previous ones ↪
167     # ↪ should be transformed into radians
168     # Also a_p should be transformed to [km] in order to have the same units
169
170     a_p = a_p * au # [km]
171     h_p = np.sqrt(mu * a_p * (1 - e_p ** 2)) # [km2/s]
172     I_p = np.radians(I_p) # [rad]
173     L_p = np.radians(L_p) # [rad]
174     w1_p = np.radians(w1_p) # [rad]

```

```

174     0_p = np.radians(0_p) # [rad]
175
176     w_p = w1_p - 0_p # [rad]
177
178     M_p = L_p - w1_p + np.radians(planet[12] * (julian_time ** 2.0)) + np.←
179         ↪ radians(planet[13]) * (
180             np.cos(np.radians(planet[15] * julian_time))) + np.radians(planet[14]) * ←
181             ↪ (
182                 np.sin(np.radians(planet[15] * julian_time)))
183
184     return h_p, a_p, e_p, I_p, L_p, w1_p, w_p, 0_p, M_p
185
186
187 def planet_state(planet, T_ephemeris, t): # t in days
188     t_julian = julian_date(T_ephemeris, t)
189
190     h_p, a_p, e_p, I_p, L_p, w1_p, w_p, 0_p, M_p = planets_elements(planet, ←
191         ↪ t_julian)
192
193     E_p, theta_p = kepler_eliptic(e_p, M_p)
194
195     R_planet, V_planet = coe_to_vstate(h_p, e_p, 0_p, I_p, w_p, theta_p)
196
197     return R_planet, V_planet
198
199 """
200
201 LAMBERT'S PROBLEM: Calculation of the orbit between two given points
202
203
204     R1 [km] Initial position of spacecraft in its trajectory. Coincides with←
205         ↪ planet position at time t1
206     R2 [km] Desired position of spacecraft at the end. Coincides with planet←
207         ↪ position at time t2
208
209 """
210
211
212 # Stumpff Functions
213
214
215 def stumpfS(z):
216     if z > 0:
217         S = (np.sqrt(z) - np.sin(np.sqrt(z))) / (np.sqrt(z)) ** 3
218     elif z < 0:
219         S = (np.sinh(np.sqrt(-z)) - np.sqrt(-z)) / (np.sqrt(-z)) ** 3
220     else:
221         S = 1 / 6
222
223     return S
224
225
226 def stumpfC(z):
227     if z > 0:
228         C = (1 - np.cos(np.sqrt(z))) / z
229     elif z < 0:

```

```

220     C = (np.cosh(np.sqrt(-z)) - 1) / (-z)
221 else:
222     C = 1 / 2
223
224 return C
225
226def lambert_problem(R1, R2, fly_time): # fly_time is given in days
227     R1_norm = np.linalg.norm(R1) # [km]
228     R2_norm = np.linalg.norm(R2) # [km]
229
230     cross_product = np.cross(R1, R2)
231     theta_lambert = np.arccos(np.dot(R1, R2) / (R1_norm * R2_norm)) # [rad]
232
233     if kind_orbit == 1: # 1: prograde orbit
234         if cross_product[2] < 0:
235             theta_lambert = 2 * np.pi - theta_lambert # [rad]
236     elif kind_orbit == 2: # 2: retrograde orbit
237         if cross_product[2] >= 0:
238             theta_lambert = 2 * np.pi - theta_lambert # [rad]
239
240     A = np.sin(theta_lambert) * np.sqrt((R1_norm * R2_norm) / (1 - np.cos(←
241             ↪ theta_lambert)))
242
242     # Additional Functions needed to calculate the variables involved in ←
243     # ↪ Lambert Problems
244
243     def y_equation(z):
244         S = stumpfS(z)
245         C = stumpfC(z)
246
247         y_z = R1_norm + R2_norm + (A * (z * S - 1)) / np.sqrt(C)
248
249     return y_z
250
251     def Fz(z):
252         y = y_equation(z)
253         S = stumpfS(z)
254         C = stumpfC(z)
255
256         if y < 0:
257             F = -1
258         else:
259             F = (((y / C) ** 1.5) * S) + (A * np.sqrt(y)) - (np.sqrt(mu) * (←
260                     ↪ fly_time*24*3600))
261
262     return F
263
263     def dFz(z):
264         y = y_equation(z)
265         S = stumpfS(z)
266         C = stumpfC(z)
267

```

```

268     if z == 0:
269         dF = (np.sqrt(2) / 40 * (y ** 1.5)) + A / 8 * (np.sqrt(y) + A * np.  

270             ↪ sqrt(1 / (2 * y)))
271     elif z != 0:
272         dF = ((y / C) ** 1.5) * ((1 / (2 * z)) * (C - ((3 * S) / (2 * C))) +  

273             ↪ ((3 * S ** 2) / (4 * C))) + (A / 8) * (  

274                 (3 * S / C * np.sqrt(y)) + (A * np.sqrt(C / y)))
275     return dF
276
277
278     z = -100
279     F = Fz(z)
280
281
282     while F < 0:
283         z = z + 0.1
284         F = Fz(z)
285
286     tol = 10 ** (-8)
287     nmax = 5000
288
289     ratio = 1
290     n = 0
291
292     while (np.absolute(ratio) > tol) and (n <= nmax):
293         n = n + 1
294         F = Fz(z)
295         dF = dFz(z)
296         ratio = F / dF
297         z = z - ratio
298
299     if n >= nmax:
300         print("Number of iterations exceeds", nmax)
301
302     f = 1 - y_equation(z) / R1_norm
303     g = A * np.sqrt(y_equation(z) / mu)
304     dg = 1 - y_equation(z) / R2_norm
305
306     V1 = 1 / g * (R2 - f * R1)
307     V2 = 1 / g * (dg * R2 - R1)
308
309 Function which returns one segment of the mission
310
311     R_planet1, V_planet1 Defines the planet state vector at the beginning of ↪
312         ↪ the transference
313     R_planet2, V_planet2 Defines the planet state vector at the end of the ↪
314         ↪ transference
315     V_departure Defines the spacecraft heliocentric velocity when it leaves the ↪
316         ↪ planet at the beginning

```

```

314                     of the transference
315 V_arrival Defines the spacecraft heliocentric velocity when it arrives the ↵
316     ↪ planet at the end of the
317             transference
318 DV_departure Defines the spacecraft planetocentric velocity when it leaves ↵
319     ↪ the planet at the beginning
320             of the transference
321 DV_arrival Defines the spacecraft planetocentric velocity when it arrives ↵
322     ↪ the planet at the end of
323         the transference
324 """
325
326
327 def segment_trajectory(planet_1, planet_2, T_ephem_begin, t_0, t_1, fly_time):
328     R_planet1, V_planet1 = planet_state(planet_1, T_ephem_begin, t_0) # [km]
329     R_planet2, V_planet2 = planet_state(planet_2, T_ephem_begin, t_1) # [km]
330
331     V_departure, V_arrival = lambert_problem(R_planet1, R_planet2, fly_time) # ↵
332         ↪ [km/s]
333
334     DV_departure = V_departure - V_planet1 # [km/s]
335     DV_arrival = V_arrival - V_planet2 # [km/s]
336
337     return R_planet1, V_planet1, R_planet2, V_planet2, V_departure, ↵
338         ↪ DV_departure, V_arrival, DV_arrival
339 """
340
341
342 """ GRAVITY ASSIST MANEUVER: rotation matrix for velocity and position vectors are ↵
343     ↪ calculated
344 """
345
346
347 # Function which returns the periapse radius
348
349 def r_periapse(v_arrival, v_departure, K_planet):
350
351     # v_arrival defines the planetocentric velocity of the spacecraft at the ↵
352         ↪ inbound crossing point of the flyby planet
353     # v_departure defines the planetocentric velocity of the spacecraft at the ↵
354         ↪ outbound crossing point of this planet
355     rotated_angle = np.dot(v_arrival, v_departure) / (np.linalg.norm(v_arrival) ↵
356         ↪ * np.linalg.norm(v_departure))
357     rotated_angle = np.arccos(rotated_angle)
358
359     r_p = (K_planet * ((1 / np.sin(rotated_angle / 2)) - 1)) / (np.linalg.norm(↵
360         ↪ v_arrival)) ** 2
361
362     return rotated_angle, r_p
363
364
365 def rot_vel_angle(v_infinity, K_planet, r_planet): # r, v are given in ↵
366     ↪ planetocentric coordinates

```

```

354     r_min = romin(r_planet)
355
356
357     rotation_v_angle = 2 * np.arcsin(K_planet / (K_planet + r_min * (v_infinity
358         ** 2))) # [rad]
359
360     return rotation_v_angle
361
362 def rotation_vel_matrix(r, v, K_planet, r_planet):
363     c1 = r[1] * v[2] - r[2] * v[1] # [km2/s]
364     c2 = r[2] * v[0] - r[0] * v[2] # [km2/s]
365     c3 = r[0] * v[1] - r[1] * v[0] # [km2/s]
366     c = np.sqrt(c1 ** 2 + c2 ** 2 + c3 ** 2) # [km2/s]
367
368     v_infinity = np.linalg.norm(v)
369
370     fi = rot_vel_angle(v_infinity, K_planet, r_planet) # [rad]
371
372     return np.array([[np.cos(fi), (c3 / c) * np.sin(fi), (-c2 / c) * np.sin(fi)],
373                     [-(c3 / c) * np.sin(fi), np.cos(fi), (c1 / c) * np.sin(fi)],
374                     [(c2 / c) * np.sin(fi), -(c1 / c) * np.sin(fi), np.cos(fi)]]) # no units
375 """
376
377 fly_by function will describe the fly by
378
379     r_1 defines spacecraft position at the beginning of the fly by in ↪
380         ↪ planetocentric coors
381     v_1 defines spacecraft velocity at the beginning of the fly by in ↪
382         ↪ planetocentric coors
383
384     r_2 defines spacecraft position at the end of the fly by in ↪
385         ↪ planetocentric coors
386     v_2 defines spacecraft position at the end of the fly by in ↪
387         ↪ planetocentric coors
388
389 R_2 defines spacecraft position at the end of the fly by in heliocentric ↪
390         ↪ coors
391 V_2 defines spacecraft velocity at the end of the fly by in heliocentric ↪
392         ↪ coors
393
394 """
395
396 def fly_by(r_1, v_1, V_planet, K_planet, r_planet):
397     v_infinity = np.linalg.norm(v_1)
398
399     v_2 = rotation_vel_matrix(r_1, v_1, K_planet, r_planet) @ v_1.T
400
401     V_2 = V_planet + v_2

```

```

396     rot_angle = rot_vel_angle(v_infinity, K_planet, r_planet)
397
398     return rot_angle, V_2

```

B.3. main.py

```

1 import matplotlib as mpl
2 from mpl_toolkits.mplot3d import Axes3D
3 import matplotlib.pyplot as plt
4 from core import *
5
6 t_0 = 0.0
7
8 T_begin = 2459400.0 # 4th July 2021
9 T_end = 2459650.0 # 11th Mars 2022
10
11 T_tof_1 = 80 # 200 days of flight
12 T_tof_1_end = 325 # 550 # 500 days of flight
13
14 T_tof_2 = 650 # 600 days of flight
15 T_tof_2_end = 950 # 1200 days of flight
16
17
18 def optimization_task(planet_1, planet_2, r_planet2, K_planet2, planet_3, ↪
19   ↪ T_beg, T_fin, T_tof_1_beg, T_tof_1_fin,
20   ↪ T_tof_2_beg,
21   ↪ T_tof_2_fin):
22
23   x = np.arange(T_beg, T_fin, 5) # Point every 5 days
24   y = np.arange(T_tof_1_beg, T_tof_1_fin, 5) # Point every 5 days
25   z = np.arange(T_tof_2_beg, T_tof_2_fin, 5) # Point every 5 days
26   X, Y, Z = np.meshgrid(x, y, z)
27
28   # Definition of arrays which will store the results
29
30   DV0_departure = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0])))
31   DV_arrival_planet_1 = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0])))
32   DV0_departure_1 = np.zeros(shape=(len(Y[:, 0, 0]), len(Z[0, 0, :])))
33   DV_arrival_planet_2 = np.zeros(shape=(len(Y[:, 0, 0]), len(Z[0, 0, :])))
34
35   DV_1_flyby = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↪
36   ↪ :])))
37
38   DV_total = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↪
39   ↪ :])))
40
41   ratio_radius = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↪
42   ↪ 0, :])))

```

```

39     fi_flyby = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↵
40         ↵ :])))
40     fi_max = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, :]))↵
40         ↵ )
41
42     for i in range(len(X[0, :, 0])):
43         for j in range(len(Y[:, 0, 0])):
44             # print("FECHA DE LANZAMIENTO -----")
45             # print(X[0, i, 0])
46             # print("TIEMPO DE VUELO 1 -----")
47             # print(Y[j, 0, 0])
48             R_1dep, V_1dep, R_2arr, V_2arr, V_hdep, DV_dep, V_harr_1, DV_arr_1 =↵
48                 ↵ segment_trajectory(planet_1, planet_2, X[0, i, 0], t_0, Y[j, ↵
48                 ↵ 0, 0], Y[j, 0, 0])
49
50             DV0_departure[i, j] = np.linalg.norm(DV_dep)
51             DV_arrival_planet_1[i, j] = np.linalg.norm(DV_arr_1)
52
53             for k in range(len(Z[0, 0, :])):
54                 # print("TIEMPO DE VUELO 2")
55                 # print(Z[0, 0, k])
56                 R_2dep, V_2dep, R_3arr, V_3arr, V_hdep_1, DV_dep_1, V_harr_2, ↵
56                     ↵ DV_arr_2 = segment_trajectory(planet_2, planet_3, X[0, i, ↵
56                     ↵ 0], t_0 + Y[j, 0, 0], t_0 + Y[j, 0, 0] + Z[0, 0, k], Z[0, ↵
56                     ↵ 0, k])
57
57             DV0_departure_1[j, k] = np.linalg.norm(DV_dep_1)
58             DV_arrival_planet_2[j, k] = np.linalg.norm(DV_arr_2)
59
60
61             fi, r_p = r_periapse(DV_arr_1, DV_dep_1, K_planet2)
62             ratio_radius[i, j, k] = np.absolute(r_p / r_planet2)
63             fi_flyby[i, j, k] = fi
64
64             if r_p > r_planet2:
65                 DV_flyby = V_hdep_1 - V_harr_1
66                 DV_flyby_mag = np.linalg.norm(DV_flyby)
67                 DV_1_flyby[i, j, k] = DV_flyby_mag
68                 fi_max[i, j, k] = 0
69                 # print("Pericentral radius was BIGGER than planet radius")
70             elif r_p <= r_planet2:
71                 r_angle, V_flyby = fly_by(R_2arr, DV_arr_1, V_2dep, K_planet2↵
71                     , r_planet2)
72                 DV_flyby = V_hdep_1 - V_flyby
73                 DV_flyby_mag = np.linalg.norm(DV_flyby)
74                 DV_1_flyby[i, j, k] = DV_flyby_mag
75                 fi_max[i, j, k] = r_angle
76                 # print("Pericentral radus was SMALLER than planet radius")
77
78             DV_total[i, j, k] = DV0_departure[i, j] + DV_1_flyby[i, j, k] + ↵
78                 ↵ DV_arrival_planet_2[j, k]
79
80

```

```
81 plt.figure()
82 plt.subplot(1, 1, 1)
83 vmin = np.min(DV0_departure)
84 v = np.linspace(vmin, 5, 6)
85 cs = plt.contour(X[0, :, 0], Y[:, 0, 0], np.transpose(DV0_departure), v, ↪
86     cmap='jet')
87 plt.colorbar(cs, label="DV_1 [km/s]")
88 plt.title("DV_1 at Departure from Earth")
89 plt.xlabel("Departure Date (in Julian Time)")
90 plt.ylabel("Time of Flight")
91 plt.show()
92
92 fig = plt.figure(figsize=(35.0, 35.0))
93 ax = fig.add_subplot(111, projection='3d')
94 scat = ax.scatter3D(X, Y, Z, c=DV_1_flyby.ravel(), cmap='jet', depthshade↪
95     =0.2)
95 plt.title("DV_flyby")
96 ax.set_xlabel("Departure Date (in Julian Time)")
97 ax.set_ylabel("Time of Flight")
98 ax.set_zlabel("Time of Flight 2")
99 fig.colorbar(scat, shrink=0.5, aspect=5, label='DV_flyby [km/s]')
100 plt.show()
101
102 fig = plt.figure(figsize=(35.0, 35.0))
103 ax = fig.add_subplot(111, projection='3d')
104 scat = ax.scatter3D(X, Y, Z, c=DV_total.ravel(), cmap='jet', depthshade↪
105     =0.1)
105 plt.title("DV_total")
106 ax.set_xlabel("Departure Date (in Julian Time)")
107 ax.set_ylabel("Time of Flight")
108 ax.set_zlabel("Time of Flight 2")
109 fig.colorbar(scat, shrink=0.5, aspect=5, label='DV_total [km/s]')
110 plt.show()
111
112 fig = plt.figure(figsize=(35.0, 35.0))
113 ax = fig.add_subplot(111, projection='3d')
114 label = np.where(ratio_radius < 1, 'yellow', 'purple')
115 label1 = np.where(ratio_radius < 1, 'purple', 'yellow')
116 ax.scatter3D(X, Y, Z, c=label.ravel(), label='r_p <= r_planet')
117 ax.scatter3D(X, Y, Z, c=label1.ravel(), label='r_p > r_planet')
118 plt.title("Ratio R_flyby to R_planet")
119 ax.set_xlabel("Departure Date (in Julian Time)")
120 ax.set_ylabel("Time of Flight")
121 ax.set_zlabel("Time of Flight 2")
122 # fig.colorbar(scat, shrink=0.5, aspect=5, label='R_flyby / R_planet')
123 ax.legend(title='R_p/R_planet ratio')
124 plt.show()
125
126 fig = plt.figure(figsize=(35.0, 35.0))
127 ax = fig.add_subplot(111, projection='3d')
```

```

128     scat = ax.scatter3D(X, Y, Z, c=fi_flyby.ravel(), cmap='viridis', depthshade=0.1)
129     plt.title("Fi Angle")
130     ax.set_xlabel("Departure Date (in Julian Time)")
131     ax.set_ylabel("Time of Flight")
132     ax.set_zlabel("Time of Flight 2")
133     fig.colorbar(scat, shrink=0.5, aspect=5, label='Fi Angle')
134
135     fig = plt.figure(figsize=(35.0, 35.0))
136     ax = fig.add_subplot(111, projection='3d')
137     scat = ax.scatter3D(X, Y, Z, c=fi_max.ravel(), cmap='coolwarm', depthshade=0)
138     plt.title("Maximum Fi Angle")
139     ax.set_xlabel("Departure Date (in Julian Time)")
140     ax.set_ylabel("Time of Flight")
141     ax.set_zlabel("Time of Flight 2")
142     fig.colorbar(scat, shrink=0.5, aspect=5, label='Max Fi Angle')
143     plt.show()
144
145     min_1 = np.min(DV_total)
146     position_min_1 = np.unravel_index(np.argmin(DV_total), DV_total.shape)
147     print("Minimum before optimization:", min_1)
148     print("Dates of minimum position before optimization")
149     print("Departure Date:", X[0, position_min_1[0], 0], "Time of Flight 1:", Y[0, position_min_1[1], 0], "Time of flight 2:", Z[0, 0, position_min_1[2]])
150
151     """
152
153     Optimization of the solution by grid search: Taking the previous minimum, next we reduce the grid search redefining
154     the ranges to look at
155
156     """
157
158     eps = 30
159
160     T_begin_opt = X[0, position_min_1[0], 0] - eps
161     T_end_opt = X[0, position_min_1[0], 0] + eps
162
163     T_tof_1_opt = Y[position_min_1[1], 0, 0] - eps
164     T_tof_1_end_opt = Y[position_min_1[1], 0, 0] + eps
165
166     T_tof_2_opt = Z[0, 0, position_min_1[2]] - eps
167     T_tof_2_end_opt = Z[0, 0, position_min_1[2]] + eps
168
169     x = np.arange(T_begin_opt, T_end_opt, 1) # Point every day
170     y = np.arange(T_tof_1_opt, T_tof_1_end_opt, 1) # Point every day
171     z = np.arange(T_tof_2_opt, T_tof_2_end_opt, 1) # Point every day
172     X, Y, Z = np.meshgrid(x, y, z)
173

```

```

174 DV0_departure = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0])))
175 DV_arrival_planet_1 = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0])))
176
177 DV0_departure_1 = np.zeros(shape=(len(Y[:, 0, 0]), len(Z[0, 0, :])))
178 DV_arrival_planet_2 = np.zeros(shape=(len(Y[:, 0, 0]), len(Z[0, 0, :])))
179
180 DV_1_flyby = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↵
181     ↵ :, :])))
182
183 DV_total = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↵
184     ↵ :, :])))
185
186 ratio_radius = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↵
187     ↵ 0, :])))
188 fi_flyby = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, ↵
189     ↵ :, :])))
190 fi_max = np.zeros(shape=(len(X[0, :, 0]), len(Y[:, 0, 0]), len(Z[0, 0, :])))
191
192 for i in range(len(X[0, :, 0])):
193     for j in range(len(Y[:, 0, 0])):
194         # print("FECHA DE LANZAMIENTO -----")
195         # print(X[0, i, 0])
196         # print("TIEMPO DE VUELO 1 -----")
197         # print(Y[j, 0, 0])
198         R_1dep, V_1dep, R_2arr, V_2arr, V_hdep, DV_dep, V_harr_1, DV_arr_1 = ↵
199             ↵ segment_trajectory(planet_1, planet_2, X[0, i, 0], t_0, Y[j, ↵
200                 ↵ 0, 0], Y[j, 0, 0])
201
202         DV0_departure[i, j] = np.linalg.norm(DV_dep)
203         DV_arrival_planet_1[i, j] = np.linalg.norm(DV_arr_1)
204
205         for k in range(len(Z[0, 0, :])):
206             # print("TIEMPO DE VUELO 2")
207             # print(Z[0, 0, k])
208             R_2dep, V_2dep, R_3arr, V_3arr, V_hdep_1, DV_dep_1, V_harr_2, ↵
209                 ↵ DV_arr_2 = segment_trajectory(planet_2, planet_3, X[0, i, ↵
210                     ↵ 0], t_0 + Y[j, 0, 0], t_0 + Y[j, 0, 0] + Z[0, 0, k], Z[0, ↵
211                         ↵ 0, k])
212
213         DV0_departure_1[j, k] = np.linalg.norm(DV_dep_1)
214         DV_arrival_planet_2[j, k] = np.linalg.norm(DV_arr_2)
215
216         fi, r_p = r_periapse(DV_arr_1, DV_dep_1, K_planet2)
217         ratio_radius[i, j, k] = r_p / r_planet2
218         fi_flyby[i, j, k] = fi
219
220         if r_p > r_planet2:
221             DV_flyby = V_hdep_1 - V_harr_1
222             DV_flyby_mag = np.linalg.norm(DV_flyby)
223             DV_1_flyby[i, j, k] = DV_flyby_mag

```

```

215         fi_max[i, j, k] = 0
216         # print("Pericentral radius was BIGGER than planet radius")
217     elif r_p <= r_planet2:
218         r_angle, V_flyby = fly_by(R_2arr, DV_arr_1, V_2dep, K_planet2←
219             ↪ , r_planet2)
220         DV_flyby = V_hdep_1 - V_flyby
221         DV_flyby_mag = np.linalg.norm(DV_flyby)
222         DV_1_flyby[i, j, k] = DV_flyby_mag
223         fi_max[i, j, k] = r_angle
224         # print("Pericentral radus was SMALLER than planet radius")
225
226         DV_total[i, j, k] = DV0_departure[i, j] + DV_1_flyby[i, j, k] + ←
227             ↪ DV_arrival_planet_2[j, k]
228
229     fig = plt.figure(figsize=(35.0, 35.0))
230     ax = fig.add_subplot(111, projection='3d')
231     scat = ax.scatter3D(X, Y, Z, c=DV_1_flyby.ravel(), cmap='jet', depthshade←
232             ↪ =1)
233     plt.title("DV_flyby")
234     ax.set_xlabel("Departure Date (in Julian Time)")
235     ax.set_ylabel("Time of Flight")
236     ax.set_zlabel("Time of Flight 2")
237     fig.colorbar(scat, shrink=0.5, aspect=5, label='DV_flyby [km/s]')
238     plt.show()
239
240     fig = plt.figure(figsize=(35.0, 35.0))
241     ax = fig.add_subplot(111, projection='3d')
242     scat = ax.scatter3D(X, Y, Z, c=DV_total.ravel(), cmap='jet', depthshade=1)
243     plt.title("DV_total")
244     ax.set_xlabel("Departure Date (in Julian Time)")
245     ax.set_ylabel("Time of Flight")
246     ax.set_zlabel("Time of Flight 2")
247     fig.colorbar(scat, shrink=0.5, aspect=5, label='DV_total [km/s]')
248     plt.show()
249
250     fig = plt.figure(figsize=(35.0, 35.0))
251     ax = fig.add_subplot(111, projection='3d')
252     label = np.where(ratio_radius < 1, 'yellow', 'purple')
253     label1 = np.where(ratio_radius < 1, 'purple', 'yellow')
254     ax.scatter3D(X, Y, Z, c=label.ravel(), label='r_p <= r_planet')
255     ax.scatter3D(X, Y, Z, c=label1.ravel(), label='r_p > r_planet')
256     plt.title("Ratio R_flyby to R_planet")
257     ax.set_xlabel("Departure Date (in Julian Time)")
258     ax.set_ylabel("Time of Flight")
259     ax.set_zlabel("Time of Flight 2")
260     # fig.colorbar(scat, shrink=0.5, aspect=5, label='R_flyby / R_planet')
261     ax.legend(title='R_p/R_planet ratio')

```

```

262     scat = ax.scatter3D(X, Y, Z, c=fi_flyby.ravel(), cmap='viridis', depthshade=←
263     ↪ =1)
264     plt.title("Fi Angle")
265     ax.set_xlabel("Departure Date (in Julian Time)")
266     ax.set_ylabel("Time of Flight")
267     ax.set_zlabel("Time of Flight 2")
268     fig.colorbar(scat, shrink=0.5, aspect=5, label='Fi Angle')
269
270     fig = plt.figure(figsize=(35.0, 35.0))
271     ax = fig.add_subplot(111, projection='3d')
272     scat = ax.scatter3D(X, Y, Z, c=fi_max.ravel(), cmap='coolwarm', depthshade=←
273     ↪ =1)
274     plt.title("Maximum Fi Angle")
275     ax.set_xlabel("Departure Date (in Julian Time)")
276     ax.set_ylabel("Time of Flight")
277     ax.set_zlabel("Time of Flight 2")
278     fig.colorbar(scat, shrink=0.5, aspect=5, label='Max Fi Angle')
279     plt.show()
280
281     min_opt = np.min(DV_total)
282     position_min_opt = np.unravel_index(np.argmin(DV_total), DV_total.shape)
283     print("Minimum after optimization", min_opt)
284     print("Minimum position after optimization", position_min_opt)
285     print("Dates of minimum value after optimization")
286     print("Departure Date:", X[0, position_min_opt[0], 0], "Time of Flight 1:", ←
287           ↪ Y[position_min_opt[1], 0, 0], "Time of Flight 2:", Z[0, 0, ←
288           ↪ position_min_opt[2]])
289
290     return X[0, position_min_opt[0], 0], Y[position_min_opt[1], 0, 0], Z[0, 0, ←
291           ↪ position_min_opt[2]]

```

B.4. validation.py

```

import numpy as np
from core import vstate_to_coe, coe_to_vstate, kepler_eliptic, mu
from core import julian_date, planets_elements, lambert_problem, planet_state,←
    ↪ segment_trajectory
from planet_consts1 import *

"""
Mutual Validation of the function vstate_to_coe/coe_to_vstate
"""

R1 = np.array([-6045.0, -3490.0, 2500.0])
V1 = np.array([-3.457, 6.618, 2.533])

R2 = np.array([-4039.9, 4814.56, 3628.62])
V2 = np.array([-10.386, -4.77192, 1.74388])

```

```
R3 = np.array([5000, 10000, 2100])
V3 = np.array([-5.99249, 1.92536, 3.24564])

R4 = np.array([-14600, 2500, 7000])
V4 = np.array([-3.31246, -4.19662, -0.385288])

R5 = np.array([3830.68, -2216.47, 6605.09])
V5 = np.array([1.50357, -4.56099, -0.291536])

def valid_vstate_to_coe_coe_tovstate():
    val1 = vstate_to_coe(R1, V1)
    rval1 = coe_to_vstate(val1[0], val1[1], val1[2], val1[3], val1[4], val1[5])
    val2 = vstate_to_coe(R2, V2)
    rval2 = coe_to_vstate(val2[0], val2[1], val2[2], val2[3], val2[4], val2[5])
    val3 = vstate_to_coe(R3, V3)
    rval3 = coe_to_vstate(val3[0], val3[1], val3[2], val3[3], val3[4], val3[5])
    val4 = vstate_to_coe(R4, V4)
    rval4 = coe_to_vstate(val4[0], val4[1], val4[2], val4[3], val4[4], val4[5])
    val5 = vstate_to_coe(R5, V5)
    rval5 = coe_to_vstate(val5[0], val5[1], val5[2], val5[3], val5[4], val5[5])

    validation = [val1, val2, val3, val4, val5]
    rvalidation = [rval1, rval2, rval3, rval4, rval5]

    for i in range(len(validation)):
        print("Cartesian to Orbital Elements Case", i + 1, ":")
        print(validation[i])
        print("")
        print("Orbital Elements to Cartesian Case", i + 1, ":")
        print(rvalidation[i])
        print("")

"""
Validation of Kepler Function through kepler_equation function
"""

e1 = 0.37255
M1 = 3.6029

e2 = 0.0167088
M2 = 4.054539479

e3 = 0.98
M3 = 2.53072

e4 = 0.1
M4 = 0.3458

e5 = 0.10234
M5 = 3.307398933
```

```
def validation_keplereq():

    val1 = kepler_eliptic(e1, M1)
    val2 = kepler_eliptic(e2, M2)
    val3 = kepler_eliptic(e3, M3)
    val4 = kepler_eliptic(e4, M4)
    val5 = kepler_eliptic(e5, M5)

    validation = [val1, val2, val3, val4, val5]

    for i in range(len(validation)):
        print("Validation of Kepler Equation", i + 1, ":")
        print("Eccentric Anomaly, True Anomaly =", validation[i])
        print("")

    ...
    Validation of Julian Date code which returns the time pass since the beginning
    ↪ in the Julian Calendar
    t is given in seconds
    ...

T_eph = 2458485.0

tJ1 = 100

tJ2 = 1504.5

tJ3 = 43200

tJ4 = 86400

T_eph_2 = 2455050.0
tJ5 = 900576.25

def validation_juliantime():
    val1 = julian_date(T_eph, tJ1)
    val2 = julian_date(T_eph, tJ2)
    val3 = julian_date(T_eph, tJ3)
    val4 = julian_date(T_eph, tJ4)
    val5 = julian_date(T_eph_2, tJ5)

    validation = [val1, val2, val3, val4, val5]

    for i in range(len(validation)):
        print("Julian time validation Case", i, ":")
        print(validation[i])

T_eph1 = 2452879.0
T_eph2 = 2504068.5
T_eph3 = 2453719.5
T_eph4 = 2457158.0
```

```
T_eph5 = 2451545.0

tjulian1 = julian_date(T_eph1, 0)
tjulian2 = julian_date(T_eph2, 0)
tjulian3 = julian_date(T_eph3, 0)
tjulian4 = julian_date(T_eph5, 0)
tjulian5 = julian_date(T_eph5, 0)

def validation_kepelements_equation():
    val1 = planets_elements(Earth, tjulian1)
    equ1 = kepler_eliptic(val1[2], val1[8])
    val2 = planets_elements(Mars, tjulian2)
    equ2 = kepler_eliptic(val2[2], val2[8])
    val3 = planets_elements(Mars, tjulian3)
    equ3 = kepler_eliptic(val3[2], val3[8])
    val4 = planets_elements(Jupiter, tjulian4)
    equ4 = kepler_eliptic(val4[2], val4[8])
    val5 = planets_elements(Pluto, tjulian5)
    equ5 = kepler_eliptic(val5[2], val5[8])

    validation = [val1, val2, val3, val4, val5]
    validequation = [equ1, equ2, equ3, equ4, equ5]

    for i in range(len(validation)):
        print("Validation", i + 1, ":")
        print(validation[i])
        print("Validation of the Kepler Equation", i + 1, ":")
        print(validequation[i])

planet1 = Earth
T_try_1 = 2452879.0

planet2 = Earth
T_try_2 = 2450394.5

planet3 = Mars
T_try_3 = 2450703.5

def validation_planet_state():
    val1 = planet_state(planet1, T_try_1, 0)
    val2 = planet_state(planet2, T_try_2, 0)
    val3 = planet_state(planet3, T_try_2, 309) # 309 days

    validation = [val1, val2, val3]

    for i in range(len(validation)):
        print("Validation", i + 1, ":")
        print(validation[i])

R_1 = np.array([5000, 10000, 2100])
R_2 = np.array([-14600, 2500, 7000])
```

```

fly_time1 = 3600 / 86400 # days

R2_1 = np.array([1.04994*10**8, 1.04655*10**8, 988.331])
R2_2 = np.array([-2.08329*10**7, -2.18404*10**8, -4.06287*10**6])
fly_time2 = 309 # days

R3_1 = np.array([7231.58074563487, 218.02523761425, 11.79251215952])
R3_2 = np.array([7357.06485698842, 253.55724281562, 38.81222241557])
fly_time3 = 12300 / 86400 # days

R4_1 = np.array([22592.145603, -1599.915239, -19783.950506])
R4_2 = np.array([1922.067697, 4054.157051, -8925.727465])
fly_time4 = 36000 / 86400 # days

def validation_lambert():
    v1 = lambert_problem(R_1, R_2, fly_time1)
    v2 = lambert_problem(R2_1, R2_2, fly_time2)
    v3 = lambert_problem(R3_1, R3_2, fly_time3)
    v4 = lambert_problem(R4_1, R4_2, fly_time4)

    validation = [v1, v2, v3, v4]

    for i in range(len(validation)):
        print("validation", i + 1, ":")
        print(validation[i])

# Validation of function which returns a segment trajectory. This functions ↪
# ↪ uses other defined and validated functions
# This is just a try to verify it works

def validation_segment_trajectory():
    t_0 = 0
    R_1p, V_1p, R_2p, V_2p, V_d, DV_d, V_a, DV_a = segment_trajectory(Earth, ↪
        ↪ Mars, 2450394.500, t_0, 309, 309)
    print("Earth position at beginning,", R_1p)
    print("Earth velocity at beginning,", V_1p)
    print("Mars position at the end,", R_2p)
    print("Mars velocity at the end,", V_2p)
    print("Heliocentric Velocity at departure;", V_d)
    print("Planetocentric Velocity at departure;", DV_d)
    print("Heliocentric Velocity at arrival;", V_a)
    print("Planetocentric Velocity at arrival;", DV_a)

# print("Validation of function which transforms Cartesian Elements into ↪
# ↪ Orbital Elements and viceversa")
# print(valid_vstate_to_coe_coe_tovstate())
# print("")
# print("")
print("Validation of function which solve Kepler Equation")
validation_keplereq()
# print("")

```

```
# print("")  
# print("Validation of function which return Julian time passed")  
# validation_juliantime()  
# print("")  
# print("")  
# print("Validation of function which returns Planets Elements")  
# validation_kepelements_equation()  
# print("")  
# print("")  
# print("Validation of function which returns planet position and velocity")  
# print(validation_planet_state())  
# print("")  
# print("")  
# print("Validation of funtcion which solve Lamberts problem")  
# print(validation_lambert())  
# print("")  
# print("")  
# print("Validation of function which returns a trajectory segment of the ↵  
↪ mission")  
# print(validation_segment_trajectory())
```

C. Anexo III - Validación del Código

La validación del código desarrollado es una de las partes más importantes de la implementación, pues permite garantizar que las funciones que se constituyen como elementales, efectivamente llevan a cabo los cálculos y procedimientos de la forma correcta. Con esto garantizado y validado paso a paso, se puede estar seguro de haber desarrollado un código que ejecutará correctamente la solución al problema planteado.

En este caso, a la hora de llevar cabo la validación, dado que las soluciones de referencia han sido tomadas de diversas fuentes, estas difieren en los valores asignados a las constantes definidas, lo cuál implica que estas deberán ser modificadas en función de los parámetros y funciones que se estén validando.

A continuación se muestra la validación del código que ha sido llevada a cabo, teniendo, en primer lugar, el valor que retorna la función implementada tras ejecutar *validation.py* y después la solución de referencia empleada para la validación.

La validación de la maniobra de fly-by será llevada a cabo manualmente y expuesta en la última sección del anexo, tanto para el caso en que $r_p > r_{planeta}$ como cuando $r_p < r_{planeta}$.

C.1. Validación de las funciones *vstate_to_coe* y *coe_to_vstate*

Ambas funciones serán validadas simultáneamente, de forma que, partiendo de un vector de estado conocido se hallarán los elementos orbitales correspondientes, y estos serán empleados como parámetros de entrada para volver al vector de estado introducido inicialmente.

Esta validación emplea como constante el parámetro gravitacional de la Tierra ($\mu = 398600 km^3/s^2$) y los resultados pueden ser consultados en (Curtis, 2005, Páginas 610, 613, 621 y 631).

Tabla C.1: Validación de las funciones que transforman coordenadas cartesianas en elementos orbitales y viceversa

Prueba	Inputs	vstate_to_coe	coe_to_vstate	Solución de referencia
1	$r = [-6045, -3490, 2500]$ $v = [-3.457, 6.618, 2.533]$	$h = 58311.66$ $e = 0.171212$ $RA = 4.45546$ $i = 2.67470$ $\omega = 0.350258$ $\theta = 0.496469$	$r = [-6045, -3490, 2500]$ $v = [-3.457, 6.618, 2.533]$	$h = 58311.7$ $e = 0.171212$ $RA = 4.45546$ $i = 2.674699$ $\omega = 0.350258$ $\theta = 0.496469$
2	$r = [-4039.9, 4814.56, 3628.62]$ $v = [-10.386, -4.77192, 1.74388]$	$h = 80000.04$ $e = 1.40000$ $RA = 0.698132$ $i = 0.523599$ $\omega = 1.04719$ $\theta = 0.523600$	$r = [-4039.9, 4814.56, 3628.62]$ $v = [-10.386, -4.77192, 1.74388]$	$h = 80000.0$ $e = 1.40$ $RA = 0.698132$ $i = 0.523599$ $\omega = 1.04719$ $\theta = 0.523599$
3	$r = [5000, 10000, 2100]$ $v = [-5.99249, 1.92536, 3.24546]$	$h = 80466.77$ $e = 0.433487$ $RA = 0.778421$ $i = 0.526934$ $\omega = 0.535925$ $\theta = 6.12313$	$r = [5000, 10000, 2100]$ $v = [-5.99249, 1.92536, 3.24564]$	$h = 80466.8$ $e = 0.433488$ $RA = 0.778421$ $i = 0.526932$ $\omega = 0.535925$ $\theta = 6.12313$
4	$r = [3830.68, -2216.47, 6605.09]$ $v = [1.50357, -4.56099, -0.291536]$	$h = 35621.39$ $e = 0.619756$ $RA = 1.91549$ $i = 1.97896$ $\omega = 5.40721$ $\theta = 2.88594$	$r = [3830.68, -2216.47, 6605.09]$ $v = [1.50357, -4.56099, -0.291536]$	$h = 35621.4$ $e = 0.619758$ $RA = 1.91549$ $i = 1.97896$ $\omega = 5.40721$ $\theta = 2.88594$

C.2. Validación de la función *kepler_eliptic*

En la siguiente tabla se muestran diferentes casos de prueba con los que la validación fue llevada a cabo. Las referencias consultadas pueden encontrarse en (Curtis, 2005, Páginas 598, 648) y Varios (2013).

La función *kepler_eliptic* devuelve como parámetros la **anomalía verdadera** y **anomalía excéntrica**.

Tabla C.2: Validación de la función *kepler_eliptic*

Prueba	Inputs	<i>kepler_eliptic</i>	Solución de referencia
1	$e = 0.37255$ $M = 3.6029$	$E = 3.47942$ $\theta = 3.37119$	$E = 3.47942$ $\theta = 3.371188$
2	$e = 0.0167088$ $M = 4.05454$	$E = 4.04145$ $\theta = 4.02843$	$E = 4.041449$ $\theta = 4.028429$
3	$e = 0.98$ $M = 2.53072$	$E = 2.83060$ $\theta = 3.11009$	$E = 2.83060$ $\theta = 3.11009$
4	$e = 0.1$ $M = 0.3458$	$E = 0.383188$ $\theta = 0.422492$	$E = 0.3832$ $\theta = 0.42250$

C.3. Validación de la función *planet_state*

La validación de la función *planet_state* conlleva la validación indirecta de las funciones *julian_date* y *planet_elements*.

Debido a esto, se expondrá principalmente la validación de *planet_state*, incluyendo un ejemplo de validación de las funciones que esta incluye.

En la práctica, las funciones *julian_date* y *planet_elements* fueron validadas de igual manera que el resto que aquí se incluyen, y con anterioridad a *planet_state* ya que constituyen las funciones elementales a las que refiere esta última.

Para esta validación el parámetro gravitacional empleado es el del Sol ($\mu = 1.327124 \cdot 10^{11} km^3/s^2$) y los elementos keplerianos son aquellos definidos en el archivo *planet_consts1.py*. Las soluciones referenciadas pueden encontrarse en (Curtis, 2005, Páginas 647, 648, 654).

Tabla C.3: Validación de la función *julian_date*

Prueba	Inputs	<i>julian_date</i>	Solución de referencia
1	$T_{eph} = 2458485.0$ $t = 100$ [días]	$t_{julian} = 0.192745$	$t_{julian} = 0.192745$
2	$T_{eph} = 2455050.0$ $t = 900576.25$ [días]	$t_{julian} = 24.7524$	$t_{julian} = 24.7524$

Tabla C.4: Validación de la función *planet_elements*

Prueba	Inputs	<i>planet_elements</i>	Solución de referencia
1	planet = Earth $T_{eph} = 2452879.0$ $t = 0$	$h = 4.45513 \cdot 10^9$ $a = 1.49600 \cdot 10^8$ $e = 0.016708$ $i = -7.43891 \cdot 10^{-6}$ $L = 5.85151$ $\omega = 1.99674$ $RA = -0.199763$ $M = 4.05453$ $E = 4.04144$ $\theta = 4.02842$	$h = 4.4551 \cdot 10^9$ $a = 1.49598 \cdot 10^8$ $e = 0.016708$ $i = -7.43891 \cdot 10^{-6}$ $L = 5.85151$ $\omega = 1.99674$ $RA = -0.199770$ $M = 4.05454$ $E = 4.04145$ $\theta = 4.02842$

Tabla C.5: Validación de la función *planet_state*

Prue- ba	Inputs	<i>planet_state</i>	Solución de referencia
1	planet = Earth $T_{eph} = 2452879.0$ $t = 0$	$r = [1.35591 \cdot 10^8, -6.68039 \cdot 10^7, 2.86913 \cdot 10^2]$ $v = [1.26803 \cdot 10^1, 2.66098 \cdot 10^1, -2.12729 \cdot 10^{-4}]$	$r = [1.35589 \cdot 10^8, -6.68029 \cdot 10^7, 286.909]$ $v = [12.6804, 26.61, -0.000212731]$
2	planet = Earth $T_{eph} = 2450394.5$ $t = 0$	$r = [1.04995 \cdot 10^8, 1.04656 \cdot 10^8, 9.88344 \cdot 10^2]$ $v = [-2.15148 \cdot 10^1, 2.09863 \cdot 10^1, 1.32282 \cdot 10^{-4}]$	$r = [1.04994 \cdot 10^8, 1.04655 \cdot 10^8, 988.33]$ $v = [-21.515, 20.9865, 0.000132284]$
3	planet = Mars $T_{eph} = 2450394.5$ $t = 309$	$r = [-2.0833 \cdot 10^7, -2.18407 \cdot 10^8, -4.06306 \cdot 10^6]$ $v = [25.03838, -0.22029, -0.62064]$	$r = [-2.08329 \cdot 10^7, -2.18404 \cdot 10^8, -4.06287 \cdot 10^6]$ $v = [25.0386, -0.220288, -0.620623]$

El pequeño error acumulado en las funciones previas repercute levemente en los resultados obtenidos al final.

C.4. Validación de la función *lambert_problem*

Para validar la función que lleva a cabo la resolución del problema de Lambert, las constantes utilizadas toman los valores definidos abajo:

Tabla C.6: Validación de la función *lambert_problem*

Prue- ba	Inputs	<i>lambert_problem</i>	Solución de referencia
1	$r_1 = [5000, 10000, 2100]$ $r_2 = [-14600, 2500, 7000]$ Órbita Prógrada	$v_1 = [-5.99249, 1.92536, 3.24564]$ $v_2 = [-3.31246, -4.19662, -0.38529]$	$v_1 = [-5.99249, 1.92536, 3.24564]$ $v_2 = [-3.31246, -4.19662, -0.385288]$
2	$r_1 = [1.04994 \cdot 10^8, 1.04655 \cdot 10^8, 988.331]$ $r_2 = [-2.08329 \cdot 10^7, -2.18404 \cdot 10^8, -4.06287 \cdot 10^6]$ Órbita Prógrada	$v_1 = [-24.42822, 21.78187, 0.94805]$ $v_2 = [22.15804, -0.19668, -0.45785]$	$v_1 = [-24.4282, 21.7819, 0.948049]$ $v_2 = [22.1581, -0.19666, -0.45784]$
3	$r_1 = [7231.5807, 218.0252, 11.7925]$ $r_2 = [7357.0648, 253.5572, 38.8122]$ Órbita Prógrada	$v_1 = [8.79257, 0.27868, 0.02581]$ $v_2 = [-8.68383, -0.28593, -0.03453]$	$v_1 = [8.79258, 0.27868, 0.02581]$ $v_2 = [-8.68383, -0.28593, -0.03453]$
4	$r_1 = [22592.1456, -1599.9152, -19783.9505]$ $r_2 = [1922.0676, 4054.1470, -8925.7274]$ Órbita Retrógrada	$v_1 = [2.96616, -1.27577, -0.75545]$ $v_2 = [5.84375, -0.20048, -5.48615]$	$v_1 = [2.96616, -1.27577, -0.75545]$ $v_2 = [5.84375, -0.20048, -5.48615]$

- $\mu = 398600 \text{ km}^3/\text{s}^2$ para las pruebas 1, 3 y 4.
- $\mu = 1.327124 \cdot 10^{11} \text{ km}^3/\text{s}^2$ para la prueba 2.
- El tipo de transferencia considerada para las pruebas 1, 2 y 3 es una órbita prógrada, mientras que para la prueba 4 es una órbita retrograda.

Las soluciones referenciadas pueden encontrarse en (Curtis, 2005, Páginas 621 y 654) y (Gim, 2011, Páginas 17 y 19).

C.5. Validación de las pork-chop plot

Para la validación de las pork-chop plot, a continuación se comparan las generadas mediante el código con las disponibles en *Interplanetary Mission Design Handbook* (NASA (1998)).

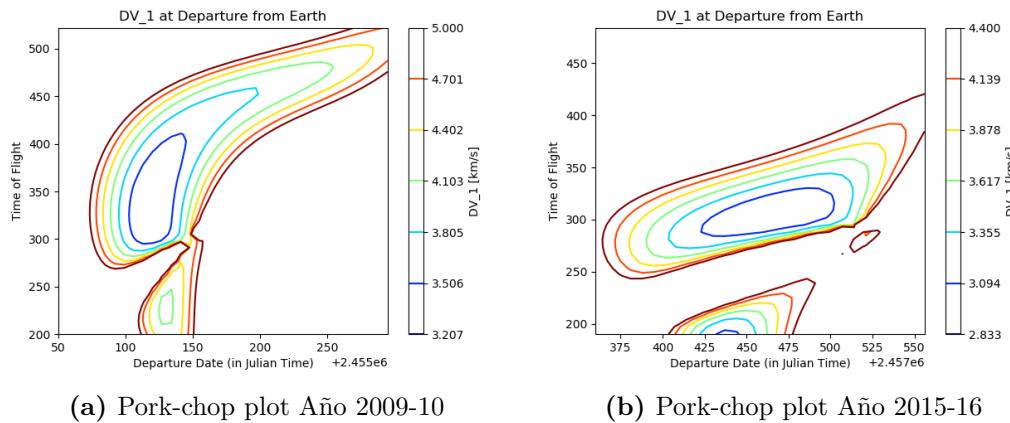


Figura C.1: Pork-chop plot generados

Las gráficas generadas mediante el código emplearon las nuevas efemérides DE405/LE405 descritas en la sección 4.4, a diferencia de las disponibles en el manual referenciado que corresponden a las efemérides previamente vigentes.

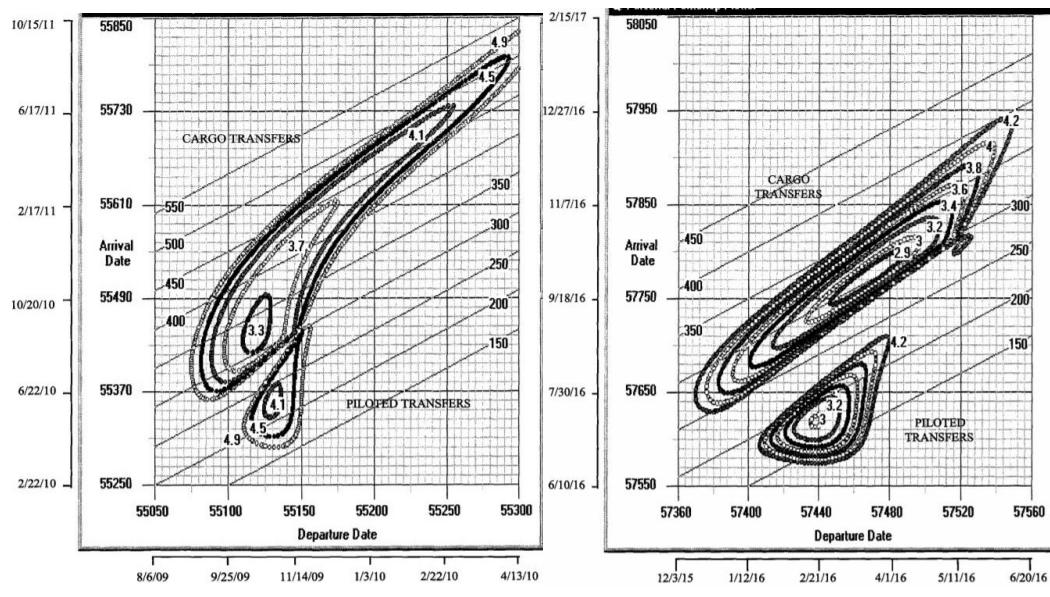


Figura C.2: Pork-chop plot de referencia

Las pork-chop plot consideradas son aquellas que se corresponden con las *cargo missions*,

puesto que las misiones tripuladas descritas en el manual corresponden a mayores energías para evitar una excesiva exposición a la radiación.

C.6. Validación manual de la mainobra de asistencia por gravedad

La validación de la maniobra asistida por gravedad, que en el código incluye las funciones *fly_by*, *rotation_vel_matrix*, *rot_vel_angle* y *r_periapse* será llevada a cabo manualmente.

A lo largo de los capítulos teóricos, se comprueba que una forma de validar que la maniobra está correctamente ejecutada es que al final de la misma se debe cumplir que:

$$|\mathbf{v}_\infty^1| = |\mathbf{v}_\infty^2| \quad (\text{C.1})$$

Por tanto, si se desarrolla manualmente este procedimiento a partir de unos valores extraídos de la ejecución del programa, y finalmente se comprueba que, en efecto, se cumple lo anterior, se puede considerar que la implementación es válida.

Se muestra a continuación el procedimiento llevado a cabo.

C.6.1. Maniobra de asistencia para el caso $r_p > r_{planeta}$

A partir de la ejecución del programa, se ha extraído la siguiente combinación de fechas en las que, teóricamente, la maniobra se encuadra en el caso en que $r_p > r_{planeta}$:

- $T_{eph} = 2457300.0$
- $t_{tof1} = 205$ días
- $t_{tof2} = 995$ días.

Estos parámetros permiten calcular el vector de estado de los planetas de interés para esta posible trayectoria:

$$\mathbf{R}_{\text{Tierra}} = [1.47039051 \cdot 10^8, 2.78667157 \cdot 10^7, -1.89445452 \cdot 10^3] \quad (\text{C.2a})$$

$$\mathbf{V}_{\text{Tierra}} = [-6.03229267, 2.91557541 \cdot 10^1, -1.31817730 \cdot 10^{-3}] \quad (\text{C.2b})$$

$$\mathbf{R}_{\text{Marte}} = [-1.54827456 \cdot 10^8, -1.73623417 \cdot 10^8, 1.82877092 \cdot 10^5] \quad (\text{C.3a})$$

$$\mathbf{V}_{\text{Marte}} = [18.99292931, -14.05193513, -0.7616692] \quad (\text{C.3b})$$

$$\mathbf{R}_{\text{Jupiter}} = [-3.03521691 \cdot 10^8, -7.39714964 \cdot 10^8, 9.76797652 \cdot 10^6] \quad (\text{C.4a})$$

$$\mathbf{V}_{\text{Jupiter}} = [11.93478514, -4.34950531, -0.24839739] \quad (\text{C.4b})$$

Una vez conocidos los vectores de estado planetarios, se recurre al problema de Lambert para calcular las velocidades de salida y llegada en cada una de las trayectorias heliocéntricas.

$$\mathbf{V}_{\text{departure}}^{\text{Tierra}} = [-16.83542925, 28.32797467, -0.03929746] \quad (\text{C.5a})$$

$$\mathbf{V}_{\text{arrival}}^{\text{Marte}} = [9.86916559, -18.86582951, 0.02586951] \quad (\text{C.5b})$$

$$\mathbf{V}_{\text{departure}}^{\text{Marte}} = [-14.24252851, -24.91423986, 0.22753191] \quad (\text{C.5c})$$

$$\mathbf{V}_{\text{arrival}}^{\text{Jupiter}} = [2.89405462, 2.49143185, 0.01434657] \quad (\text{C.5d})$$

En coordenadas planetocéntricas, estos vectores se definen como:

$$\mathbf{v}_{\text{departure}}^{\text{Tierra}} = [-10.80313658, -0.82777944, -0.03797928] \quad (\text{C.6a})$$

$$\mathbf{v}_{\text{arrival}}^{\text{Marte}} = [-9.12376372, -4.81389438, 0.78753871] \quad (\text{C.6b})$$

$$\mathbf{v}_{\text{departure}}^{\text{Marte}} = [-33.23545782, -10.86230473, 0.9892011] \quad (\text{C.6c})$$

$$\mathbf{v}_{\text{arrival}}^{\text{Jupiter}} = [-9.04073052, 6.84093717, 0.26274396] \quad (\text{C.6d})$$

El módulo de $\mathbf{v}_{\text{arrival}}^{\text{Marte}}$ definirá el vector \mathbf{v}_∞ , de forma que es posible calcular tanto el ángulo de rotación φ como el radio de pericentro de la hipérbola r_p .

Al considerar la maniobra de asistencia gravitacional en un plano, el ángulo de rotación φ puede calcularse como el ángulo definido por los vectores velocidad a la llegada y a la salida de Marte en coordenadas planetocéntricas:

$$\varphi = \arccos \frac{\mathbf{v}_{\text{arrival}}^{\text{Marte}} \cdot \mathbf{v}_{\text{departure}}^{\text{Marte}}}{v_{\text{arrival}}^{\text{Marte}} \cdot v_{\text{departure}}^{\text{Marte}}} = 0.1760128974 \text{ rad} \quad (\text{C.7})$$

$$r_p = \frac{K_{\text{Marte}} \cdot \left(\frac{1}{\sin \varphi/2} - 1 \right)}{v_\infty^2} = 4152.279052 \text{ km} \quad (\text{C.8})$$

Donde v_∞ es definido por el vector $\mathbf{v}_{\text{arrival}}^{\text{Marte}}$.

Ahora es posible calcular la maniobra de asistencia mediante la matriz de rotación definida en la Ecuación 3.4. El vector resultante de esta rotación deberá tener el mismo módulo que $\mathbf{v}_{\text{arrival}}^{\text{Marte}}$.

A la hora de calcular las componentes del vector de áreas \mathbf{c} , es necesario tener tanto un vector velocidad como su correspondiente vector posición.

Dado que el vector velocidad a emplear ($\mathbf{v}_{\text{arrival}}^{\text{Marte}}$) es un vector que se encuentra en el infinito, desde la perspectiva del planeta sobre el cuál se realiza el fly-by, el vector posición deberá estar por tanto, en el infinito. Para ello, es posible elegir cualquier vector en el infinito cuya dirección sea paralela al vector velocidad y su sentido sea el contrario.

El vector abajo definido es obtenido según $r = -R_{Marte} + 2 \cdot r_{SOI}$:

$$\mathbf{r} = [1.55983995 \cdot 10^8, 1.74779956 \cdot 10^8, 973662 \cdot 10^5] \quad (\text{C.9a})$$

$$\mathbf{v}_\infty = \mathbf{v}_{\text{arrival}}^{\text{Marte}} = [-9.12376372, -4.81389438, 0.78753871] \quad (\text{C.9b})$$

Con esto, ya se pueden definir las componentes del vector de áreas y la matriz de rotación según:

$$c_1 = y \cdot V_Z - z \cdot V_y = 1.42333087 \cdot 10^8 \quad (\text{C.10a})$$

$$c_2 = z \cdot V_x - x \cdot V_z = -1.31726896 \cdot 10^8 \quad (\text{C.10b})$$

$$c_3 = x \cdot V_y - y \cdot V_x = 8.43760544 \cdot 10^8 \quad (\text{C.10c})$$

$$c = 8.65761248 \cdot 10^8 \quad (\text{C.10d})$$

$$\Omega = \begin{bmatrix} 0.9845496801 & 0.170655699 & 0.02664256546 \\ -0.170655699 & 0.9845496801 & 0.02878773204 \\ -0.02664256546 & -0.02878773204 & 0.9845496801 \end{bmatrix} \quad (\text{C.11})$$

Y llevando a cabo la operación definida en la ecuación 3.3b, se obtiene el vector \mathbf{v}_∞^2 :

$$\mathbf{v}_\infty^2 = [-9.783335111, -3.159824443, 1.157032559] \quad (\text{C.12})$$

Cuyo módulo resulta ser:

$$v_\infty^2 = 10.34586201 = v_{\text{arrival}}^{\text{Marte}} \quad (\text{C.13})$$

Por tanto, la maniobra así planteada resulta válida.

A la hora de ejecutar el programa, en el caso en que $r_p > r_{\text{planeta}}$ la función a validar sería $r_periapse$ que retorna los valores de φ y r_p , resultando:

Tabla C.7: Validación de la función $r_periapse$

r_periapse	Cálculo manual
$\varphi = 0.1760128974$	$\varphi = 0.1760128974$
$r_p = 4152.2790496$	$r_p = 4152.279052$

C.6.2. Maniobra de asistencia para el caso $r_p < r_{\text{planeta}}$

Las fechas para las que se llevará a cabo la validación son las siguientes:

- $T_{eph} = 3457300.0$
- $t_{tof1} = 235$ días
- $t_{tof2} = 695$ días.

Con estos valores se calculan los vectores de estado planetarios:

$$\mathbf{R}_{\text{Tierra}} = [1.47039051 \cdot 10^8, 2.78667157 \cdot 10^7, -1.89445452 \cdot 10^3] \quad (\text{C.14a})$$

$$\mathbf{V}_{\text{Tierra}} = [-6.03229267, 2.91557541 \cdot 10^1, -1.31817730 \cdot 10^{-3}] \quad (\text{C.14b})$$

$$\mathbf{R}_{\text{Marte}} = [-1.00610471 \cdot 10^8, -2.03338557 \cdot 10^8, -1.77397172 \cdot 10^6] \quad (\text{C.15a})$$

$$\mathbf{V}_{\text{Marte}} = [22.62838728, -8.66727893, -0.73861062] \quad (\text{C.15b})$$

$$\mathbf{R}_{\text{Jupiter}} = [-5.54723752 \cdot 10^8, -5.89567578 \cdot 10^8, 1.47584975 \cdot 10^7] \quad (\text{C.16a})$$

$$\mathbf{V}_{\text{Jupiter}} = [9.36159565, -8.34508321, -0.17482832] \quad (\text{C.16b})$$

Empleando ahora los vectores posición para calcular Lambert, finalmente se obtienen las velocidades de llegada y salida a Marte en coordenadas planetocéntricas:

$$\mathbf{v}_{\text{arrival}}^{\text{Marte}} = [-6.29835679, -4.03435664, 0.58621671] \quad (\text{C.17a})$$

$$\mathbf{v}_{\text{departure}}^{\text{Marte}} = [-5.21553423, 32.26858265, 0.51053776] \quad (\text{C.17b})$$

De igual modo que en el caso anterior, ahora es posible calcular tanto el ángulo φ como r_p :

$$\varphi = \arccos \frac{\mathbf{v}_{\text{arrival}}^{\text{Marte}} \cdot \mathbf{v}_{\text{departure}}^{\text{Marte}}}{v_{\text{arrival}}^{\text{Marte}} \cdot v_{\text{departure}}^{\text{Marte}}} = 1.977542119 \text{ rad} \quad (\text{C.18})$$

$$r_p = \frac{K_{\text{Marte}} \cdot \left(\frac{1}{\sin \varphi/2} - 1 \right)}{v_{\infty}^2} = 149.96654110 \text{ km} \quad (\text{C.19})$$

Verificando, por tanto, que esta maniobra no es factible, siendo necesario recurrir a calcular un ángulo de rotación φ_{max} empleando como radio de pericentro de la hipérbola:

$$r_{min} = r_{\text{Marte}} + h_{atm} = 3396.2 \text{ km} + 600 \text{ km} = 3996.2 \text{ km} \quad (\text{C.20})$$

De esta forma, se tiene:

$$\varphi_{max} = 2 \cdot \arcsin \frac{K_{\text{Marte}}}{K_{\text{Marte}} + r_{min} \cdot (v_{\text{arrival}}^{\text{Marte}})^2} = 0.3212662596 \text{ rad} \quad (\text{C.21})$$

Definiendo ahora un vector \mathbf{r} en el infinito, y junto con el vector $\mathbf{v}_{\text{arrival}}^{\text{Marte}}$ se pueden desarrollar tanto las componentes del vector de áreas como el cálculo del vector \mathbf{v}_{∞}^2 .

$$\mathbf{r} = 1.01767010 \cdot 10^8, 2.04495096 \cdot 10^8, 2.930510 \cdot 10^6] \quad (\text{C.22a})$$

$$\mathbf{v}_{\text{arrival}}^{\text{Marte}} = [-6.29835679, -4.03435664, 0.58621671] \quad (\text{C.22b})$$

$$c_1 = y \cdot V_z - z \cdot V_y = 1.317011649 \cdot 10^8 \quad (\text{C.23a})$$

$$c_2 = z \cdot V_x - x \cdot V_z = -7.811491935 \cdot 10^7 \quad (\text{C.23b})$$

$$c_3 = x \cdot V_y - y \cdot V_x = 8.774186639 \cdot 10^8 \quad (\text{C.23c})$$

$$c = 8.90679880 \cdot 10^8 \quad (\text{C.23d})$$

Obteniendo finalmente:

$$v_\infty^2 = 7.502598336 \text{ km/s} = v_{arrival}^{Marte} \quad (\text{C.24})$$

Con estos resultados se puede validar la función *fly_by*, así como las funciones que esta incluye:

Tabla C.8: Validación de la función *fly_by*

fly_by	Cálculo manual
$\varphi_{max} = 0.3212662596$	$\varphi_{max} = 0.3212662596$
$V_2 = [15.41355739 \ -10.50864181 \ 0.18040742]$	$V_2 = [15.4135574, -10.3086418, 0.180407428]$