



UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE
INGEGNERIA ROBOTICA E DELL'AUTOMAZIONE

ROBOTICA AEROSPAZIALE

Missione EJE

Earth - Jupiter - Europa

20 Luglio 2021

Studenti:

Gazzanelli Niccolò

Incerpi Lorenzo

Lupi Marco

Vergara Francesca

Indice

1	Introduzione	3
1.1	Specifiche	3
1.2	Assunzioni	3
1.3	Fasi della missione	4
2	Codice MATLAB	5
2.1	Organizzazione del codice	5
2.2	Contenuto delle cartelle	6
2.2.1	computation_functions	6
2.2.2	animation_functions	8
3	Creazioni del Sistema Solare	10
4	Pianificazione della missione	13
4.1	Funzioni per la scelta delle date	13
4.2	Orbita di parcheggio e determinazione delle date di interesse	15
5	Svolgimento della missione	17
5.1	Fase iniziale: fuga dalla Terra	17
5.2	Prima parte del volo: fly-by	19
5.2.1	Primo Flyby - Marte	21
5.2.2	Secondo Flyby - Terra	22
5.3	Seconda parte del volo: Fase Interplanetaria Terra - Giove	25
5.4	Terza parte del volo: verso Europa	26
5.4.1	Ingresso SOI di Giove e parcheggio	26
5.4.2	Trasferimento su Europa	30
6	Missione alternativa	32
6.1	Fase interplanetaria	32
6.2	Fase di Fuga - Terra	33
6.3	Fase di cattura e parcheggio - Giove	34
7	Conclusioni	37

Capitolo 1

Introduzione

1.1 Specifiche

Il progetto consiste nel pianificare e simulare una missione interplanetaria con partenza dalla Terra e arrivo ad uno dei satelliti naturali di Giove, Europa.

In particolare, la sonda parte da un'orbita circolare geostazionaria ed equatoriale ad altezza di 200 km dalla superficie terrestre, per arrivare su un'orbita circolare ed equatoriale attorno ad Europa ad altezza di 100 km.

Prima di trasferirsi sul satellite inoltre, la sonda deve prima entrare in orbita attorno a Giove, con tipo di orbita e durata di permanenza su essa a nostra scelta.

1.2 Assunzioni

L'intero progetto è stato svolto adottando l'approssimazione delle *patched conics*, secondo cui la traiettoria complessiva della sonda risulta essere l'unione di diverse traiettorie che tengono conto della sfera di influenza (SOI) che viene assegnata ad ogni corpo celeste.

Quando la sonda si trova all'interno della SOI di un corpo celeste diverso dal Sole, viene considerata dominante la forza gravitazionale tra sonda e tale corpo, trascurando quindi le interazioni coi restanti corpi. In caso contrario, risulta dominante la forza gravitazionale tra sonda e corpo principale (il Sole), e di conseguenza tale forza viene considerata l'unica agente sulla sonda stessa. Si può dimostrare che

$$R_{SOI} = \left(\frac{m_p}{m_s}\right)^{2/5} d_p$$

con m_p massa del pianeta di cui si vuole calcolare la sfera d'influenza, m_s massa del Sole e d_p distanza del pianeta dal Sole. In generale, m_s potrà essere la massa del corpo rispetto al quale si vuole calcolare la sfera d'influenza nel caso in cui non sia il Sole ad essere dominante. Ad esempio, nel caso di Europa, m_s sarà la massa di Giove, e d_p la distanza di Europa da Giove stesso.

Questa tecnica ci risulterà utile per semplificare i calcoli della traiettoria della sonda, evitandoci di dover risolvere un *n-body problem* e anzi, riconducendoci a un *2-body problem*, di cui conosciamo la soluzione in forma chiusa (ovvero sezioni coniche delle orbite di Keplero).

Inoltre si sono considerate le manovre come impulsive, quindi eseguite tramite un Δv che può essere trattato come istantaneo; anche la manovra complessiva di flyby, nota come fionda gravitazionale, è stata considerata istantanea. Infine, si è scelto di assumere come date ammissibili

per la partenza tutte quelle successive al periodo in cui si è svolto il progetto, in particolare dal 01/05/2021.

Per questa missione di sola andata, è richiesto che la navicella sfrutti almeno una volta gli effetti della fionda gravitazionale di un pianeta di passaggio appropriato al caso, permettendo una variazione di energia cinetica a consumi di carburante nulli.

In ultimo, è anche richiesto che la missione si svolga sul piano dell'eclittica. E' importante osservare che si è deciso comunque di svolgere il progetto considerando le orbite reali tridimensionali, riservandosi di semplificare la pianificazione solo in casi di estrema necessità, come ad esempio la progettazione della manovra di Hohmann. In questa modo, è stato possibile ottenere traiettorie più realistiche, ed avere un'idea verosimile riguardo alla grandezza dei Δv in gioco.

1.3 Fasi della missione

La missione, come anticipato in precedenza, sfrutta l'approssimazione delle *patched conics*. Una delle difficoltà più grandi incontrata nelle fasi iniziali è stata la scelta di date che ci permettessero l'ottenimento di un Δv di missione soddisfacente e, allo stesso tempo, che rispettassero la specifica sulla manovra di flyby.

Dopo diversi tentativi *trial & error*, è stato deciso di prendere a riferimento alcune date indicative fornite dalla missione *Europa Clipper* in sviluppo dalla NASA , e impostare un problema di ottimizzazione in una finestra temporale attorno alle suddette date.

Alla fine, il risultato del design della missione interplanetaria è stata la sua suddivisione in:

1. Cambio del piano orbitale sull'eclittica
2. Uscita iperbolica dalla SOI della Terra
3. Trasferimento Terra - Marte
4. Fly-by su Marte
5. Trasferimento Marte - Terra
6. Fly-by su Terra
7. Trasferimento da Terra - Giove
8. Ingresso iperbolico nella SOI di Giove e parcheggio su un'orbita attorno ad esso
9. Permanenza per un periodo (circa 36 ore) sull'orbita di parcheggio
10. Trasferimento su un'orbita circolare attorno ad Europa

Nell'elaborato è riportato anche un confronto di tale missione con una alternativa che prevede il raggiungimento di Giove servendosi di un'unica manovra alla Hohmann con partenza dalla Terra. Nell'apposita sezione, il confronto verrà effettuato circa i tempi di volo e Δv .

Di seguito, verrà spiegato come si è svolto il progetto, con particolare attenzione all'organizzazione del codice utilizzato, partendo prima dalla creazione del sistema solare per poi passare alla pianificazione della missione e allo svolgimento del calcolo della rotta della navicella.

Capitolo 2

Codice MATLAB

Durante lo sviluppo della missione, ci siamo serviti di alcune funzioni che il libro *Howard D. Curtis - Orbital Mechanics for Engineering Students* mette a disposizione, e di altre create da noi. In particolare, abbiamo scelto di utilizzare le funzioni del Curtis che fornissero la risoluzione dei problemi di basso livello, quali ad esempio il calcolo del vettore di stato in un certo istante, il problema di Lambert, il passaggio da vettore di stato a coefficienti orbitali e viceversa.

2.1 Organizzazione del codice

Al fine di un'esposizione più chiara, l'implementazione su MATLAB viene presentata in accordo con l'organizzazione del codice.

All'interno della cartella principale `EJE_Mission`, si distinguono le seguenti sotto cartelle:

- `ephemerides`: contenente i files con i dati sulle effemeridi di Europa e Io;
- `computation_functions`: contenente funzioni per le manovre e funzioni di ausilio. I vari scripts sono ulteriormente suddivisi in altre sotto-cartelle, ognuna delle quali si occupa di una determinata classe di problemi:
 - `dates_choice`: vi si trovano le funzioni che consentono una scelta ottimale delle date per le manovre critiche;
 - `orbital_maneuvers`: vi si trovano le funzioni per l'esecuzione delle manovre;
 - `phasing`: vi si trovano le funzioni che calcolano i tempi di attesa per poter effettuare un trasferimento alla Hohmann o alla Lambert, e le tempistiche per il trasferimento su Europa;
 - `trajectory_generation`: vi si trovano le funzioni che computano la traiettoria della sonda in ogni fase della missione, e la traiettoria di Io ed Europa per poter eseguire il trasferimento su quest'ultimo tenendo conto anche della posizione del primo;
- `animation_functions`: contenente i files per implementare la parte grafica del progetto;
- `M_files_Curtis`: contenente gli scripts messi a disposizione dal libro *Howard D. Curtis - Orbital Mechanics for Engineering Students*;

Tra i file del libro in particolare, ci siamo appoggiati principalmente alle funzioni:

- `planet_elements_and_sv`: da noi modificata in `body_elements_and_sv`, in modo da contenere anche i dati relativi a Io e Europa;

- `lambert`: per la risoluzione del problema di Lambert;
- `rkf45`: per l'integrazione numerica effettuata a partire da condizioni iniziali a scelta dell'utente;
- `rv_from_r0v0`: per ricavare il vettore di stato a partire dalle condizioni iniziali r_0 e v_0 .

Maggiori dettagli implementativi sui file presenti nelle cartelle più importanti saranno forniti nel seguito del capitolo.

Infine, per una esecuzione più immediata è stato scelto di lasciar fuori da ogni sotto-cartella i file per l'esecuzione dell'intera missione, sia per la parte computazionale che per la parte grafica. In particolare, all'interno della cartella principale, sono presenti quattro file:

1. `initializeEJE`: file fondamentale per inizializzare il workspace di MATLAB, in cui sono state introdotte tutte le costanti del Sistema Solare (masse, raggi, costanti gravitazionali, distanze medie dal Sole, periodi orbitali e inclinazione assiale di ogni pianeta di interesse). È il primo file da mandare in *running* per permettere l'esecuzione dell'intera missione. Infatti, contiene tutta una serie di dati condivisi fra le varie funzioni, che per utilizzarle dovranno dichiararle `global`;
2. `entire_mission_plot`: file che si occupa di mandare in esecuzione l'intera missione con relativa interfaccia grafica. E' stato scelto che, per consentire all'utente inesperto di non dover eseguire comandi aggiuntivi, tale script inizializzi al suo interno tutte le variabili di cui ha bisogno.
3. `europa_transfer_plot`: file che si occupa di mandare in esecuzione soltanto la parte di missione che riguarda il trasferimento da Giove a Europa. Viene chiamato all'interno di `entire_mission_plot`, ma è stato scelto di mantenerlo comunque separato sia per dare la possibilità all'utente di visualizzare soltanto la seconda parte della missione che per motivi di debug.
4. `hohmann_transfer_plot`: file che si occupa di mandare in esecuzione la missione eseguita con la manovra di Hohmann.

2.2 Contenuto delle cartelle

2.2.1 `computation_functions`

Particolare attenzione merita la cartella `computation_functions`, che come anticipato in precedenza è a sua volta suddivisa in altre sotto-cartelle. Si tratta del cuore del progetto, in quanto contiene tutte le funzioni necessarie sia ad effettuare la scelta delle date che ad eseguire le manovre orbitali.

`dates_choice`

Questa cartella contiene tutti gli script necessari a risolvere un problema di ottimizzazione a partire dalle date fornite dalla missione Europa Clipper. Le sue funzionalità hanno coperto una parte fondamentale del progetto, e per questo sarà descritta in dettaglio nel paragrafo 4.1.

orbital_maneuver

Per quanto riguarda le manovre orbitali, sono presenti una serie di funzioni caratterizzate da una struttura modulare. Consideriamo le manovre che coinvolgono un solo corpo principale, ovvero iperbole di uscita, iperbole di cattura e flyby: tali funzioni prendono in ingresso indicazioni sul pianeta da considerare, in modo da poter prelevare le informazioni necessarie, e sulle velocità in gioco. In uscita, restituiscono l'informazione principale sul Δv necessario ad eseguire la manovra, e informazioni secondarie sulle caratteristiche orbitali. E' stata implementata anche la manovra di Hohmann, che presenta tuttavia diverse limitazioni non trascurabili che ci hanno obbligato alla ricerca di soluzioni alternative. Una limitazione significativa è la natura strettamente bidimensionale della manovra, che può comunque essere aggirata supponendo di lavorare sul piano dell'eclittica. Un'altra limitazione, probabilmente la più importante, è l'ipotesi di traiettorie circolari: si può notare che l'eccentricità delle orbite dei pianeti considerati nello svolgimento del progetto è prossima allo zero (ovvero traiettoria realmente circolare). Tuttavia, utilizzando i raggi medi nel calcolo dei vari Δv e i parametri orbitali "veri" nelle animazioni e nelle impostazioni (e risoluzioni) dei problemi di Lambert, si è verificata la situazione indesiderata ma non inaspettata in cui la sonda non riusciva a raggiungere correttamente il pianeta target. Ad esempio, tale distanza risulterà essere funzione della data in cui avviene la manovra. Questo perché il modulo della posizione del pianeta target dipende dalla particolare data di arrivo, visto che esso non segue una traiettoria perfettamente circolare.

phasing

Come anticipato all'inizio del capitolo, questa cartella contiene le funzioni che ci hanno permesso di gestire le tempistiche delle varie manovre.

E' presente la funzione `lambert_phasing`, utilizzata principalmente in una fase preliminare del progetto per acquisire sicurezza con gli strumenti a disposizione. Essa ci consente la risoluzione di problemi relativi al calcolo di un'orbita di trasferimento in cui, una volta preso a riferimento il tempo necessario ad eseguire una manovra di Hohmann equivalente, vengono risolti iterativamente una serie di problemi di Lambert confrontando i Δv risultanti. Tali problemi di Lambert hanno come input relativo al tempo di trasferimento un intorno di quello di Hohmann.

E' presente anche una funzione simile, `hohmann_phasing`, risultata molto utile per la progettazione della missione alternativa. Quest'ultima doveva necessariamente usare la manovra di Hohmann per il trasferimento da Terra a Giove. L'obiettivo di tale funzione, nota una certa data di riferimento da passare come input, è quello di restituire il tempo di attesa che deve trascorrere prima di poter iniziare la manovra. Si prenda a riferimento la figura 2.1; è necessario trovare l'angolo θ_H per cui, se la partenza avviene in quell'istante, il pianeta target si trova esattamente all'apoasse dell'orbita di trasferimento all'istante di arrivo.

E' presente infine la funzione `europa_phasing`, fondamentale per il trasferimento da Giove a Europa. Anche in questo caso vengono risolti diversi problemi di Lambert in un certo range temporale, in modo da ottenere quello che minimizzi il Δv necessario per iniziare la manovra. Il risultato dell'iterazione sarà utilizzato successivamente dagli script che si occupano dell'animazione. Inoltre, si vedrà in seguito che la manovra risultante è molto simile a quella di Hohmann.

trajectory_generation

Questa cartella contiene una serie di funzioni utili soprattutto per la parte grafica. Sono presenti tre funzioni strutturalmente molto simili: `gen_europa_traj`, `gen_io_traj` e

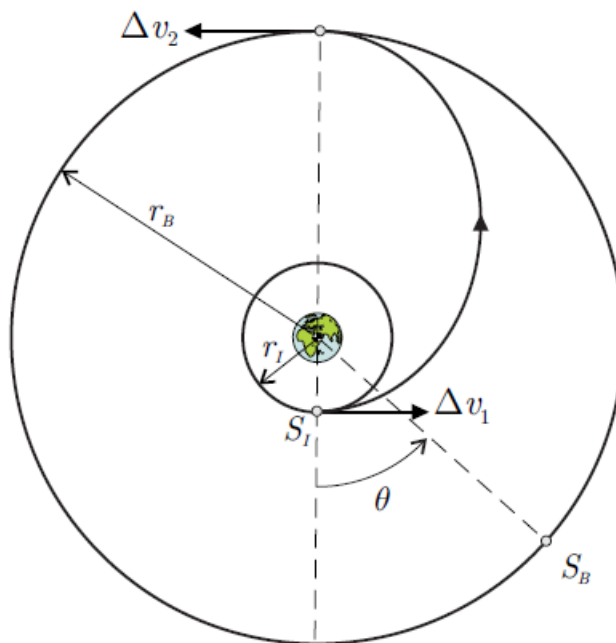


Figura 2.1: Rifasamento con manovra di Hohmann

`gen_jupiter_park_orbit` il cui obiettivo è quello di calcolare la traiettoria sia della sonda che dei satelliti in relazione alla fase di trasferimento da Giove a Europa.

Queste funzioni hanno una caratteristica fondamentale a livello implementativo che le distingue dalle altre: sono state ottenute utilizzando un algoritmo di integrazione a passo fisso. Questo si è reso necessario per ottenere delle animazioni che rispettassero la fisica del problema: infatti, utilizzando l'algoritmo di integrazione di Runge-Kutta a step variabile, durante l'animazione si ottenevano delle velocità non coerenti. In particolare, la sonda risultava avere una velocità proporzionale alla distanza dal pianeta; mano a mano che si allontanava dal centro del pianeta, la sonda accelerava. Un altro vantaggio dell'utilizzo di questo algoritmo, fornito dal Curtis, è stato quello di poter scegliere la dimensione dello step: questo ci ha permesso di ottenere una notevole precisione nel calcolo della posizione e della velocità dei corpi in gioco, togliendo di fatto la necessità di utilizzare eventuali approssimazioni. Un tentativo era stato fatto utilizzando la funzione `rv_from_r0v0`, già citata in precedenza, ma essendo l'ordine di grandezza delle quantità in gioco nettamente inferiore rispetto alla fase eliocentrica, in cui la funzione è stata utilizzata con successo, il risultato non è stato accettabile.

E' presente infine la funzione `gen_patched_conic_orbit`, utilizzata anch'essa principalmente nella parte grafica per il calcolo delle coniche raccordate della fase eliocentrica. Al suo interno, viene sfruttata la funzione del Curtis `interplanetary` che, dati in ingresso i pianeti di partenza e di arrivo con le rispettive date, restituisce le velocità risultanti dalla risoluzione del problema di Lambert associato agli input.

2.2.2 animation_functions

Questa cartella contiene le funzioni necessarie a fare il plot delle manovre. In generale, data una manovra `<maneuver>.m`, è presente il file `<maneuver>_plot.m`, che implementa una funzione che di default viene chiamata al termine della prima, ad essa associata, e che sarà per questo contenuta in `orbital_maneuver`. Queste due funzioni condividono delle variabili necessarie per

fare il plot, in modo da non doverne ripetere il calcolo.

Una delle maggiori difficoltà incontrate nella parte grafica è stata la scelta del sistema di riferimento in cui eseguire il plot.

Alla fine si è optato per un sistema di riferimento perifocale individuato dai versori :

- dell'asse x è diretto verso il pericentro, giacente sul piano orbitale
- dell'asse y perpendicolare e giacente sullo stesso piano
- dell'asse z di conseguenza

Tuttavia, per rendere l'animazione più interessante, si è deciso di ruotare tale sistema di riferimento in modo da poter cogliere la differenza tra il piano orbitale e quello equatoriale del pianeta considerato, ottenendo così un sistema di riferimento simil-perifocale.

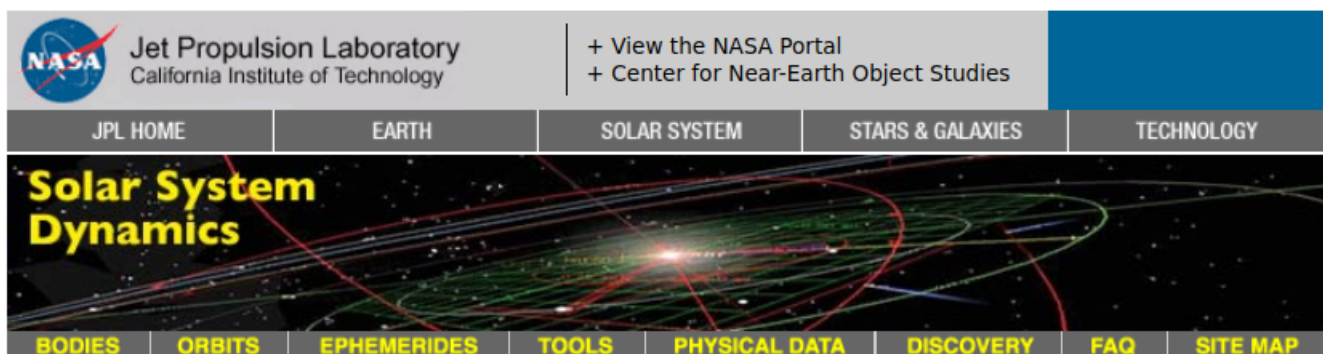
A livello implementativo, le orbite sono state generate prendendo a riferimento le grandezze calcolate dalle rispettive funzioni di manovra, integrando con l'algoritmo di Runge-Kutta a step variabile, a partire da determinate condizioni iniziali. Le animazioni, che sfruttano passo per passo la posizione risultante dall'integrazione, sono state realizzate tramite l'utilizzo della funzione MATLAB `animatedLine`. Per quanto riguarda il flyby, è stato aggiunto nel plot un vettore che permettesse la distinzione tra flyby da dietro e da davanti.

Infine, per quanto riguarda le manovre di cattura e di fuga, è stato aggiunto un controllo, effettuato tramite la funzione di utilità `compute_soi`, per verificare che le traiettorie ottenute dall'integrazione non uscissero dalla sfera d'influenza del pianeta considerato, ed è stato deciso di plottare anche le rispettive orbite di parcheggio scelte dall'utente.

Capitolo 3

Creazioni del Sistema Solare

Durante la scrittura del codice è stato necessario avere a disposizione lo stato dei corpi celesti di interesse. In particolare, come già anticipato, ci siamo appoggiati alla funzione `planet_elements_and_sv` che consente di ricavare gli elementi orbitali e il vettore di stato di un pianeta, fissata la data di interesse. Una volta modificata per i nostri scopi, è stata poi ribattezzata in `body_elements_and_sv`. Abbiamo infatti aggiunto le effemeridi dei satelliti gioviani, Europa e Io, tramite caricamento di due file `.m` generati a partire dal sito [Nasa Jet-PropulsionLaboratory Horizons](#).



HORIZONS Web-Interface

This tool provides a web-based *limited* interface to JPL's [HORIZONS system](#) which can be used to generate ephemerides for solar-system bodies. Full access to [HORIZONS](#) features is available via the primary [telnet interface](#). [HORIZONS system news](#) shows recent changes and improvements. A [web-interface tutorial](#) is available to assist new users.

Current Settings

Ephemeris Type [\[change\]](#) : **VECTORS**
Target Body [\[change\]](#) : **Europa (JII)** [502]
Coordinate Origin [\[change\]](#) : **Jupiter (body center)** [500@599]
Time Span [\[change\]](#) : Start=**2000-01-01**, Stop=**2050-01-01**, Step=**1 d**
Table Settings [\[change\]](#) : *defaults*
Display/Output [\[change\]](#) : **plain text**

[Generate Ephemeris](#)

Figura 3.1: Sito NASA per la generazione delle effemeridi

I vettori di posizione e velocità dei due corpi celesti sono stati generati selezionando come origi-

ne delle coordinate il centro di Giove anziché del Sole. Questo ci risulterà comodo quando sarà necessario plottare la traiettoria della sonda nel sistema gioviano. In caso contrario, avremmo infatti sempre dovuto tener conto di un *offset tempo-variante* :

- di posizione, pari alla distanza tra Sole e Giove ;
- di orientazione, descritto da una matrice di rotazione, fra le coordinate del frame gioviano e il frame solare.

Non a caso ora lo script viene usato come funzione che restituisce lo stato di un corpo *rispetto al fuoco della sua orbita*.

Tutti i valori ottenuti dal sito, sia per Europa che per Io, sono stati caricati all'interno di un file di testo da cui è stato derivato un file .m che riportasse tutti i dati, impilati in un array. In questo modo, nella funzione `body_elements_and_sv` è possibile estrarre il vettore posizione e velocità anche di Europa e Io ad una certa data.

Qui sotto, il caricamento dei nuovi dati all'interno di `body_element_and_sv`.

```

1      if(body_id == 10 || body_id == 12)
2          % Data loading
3          if(body_id == 10)
4              data = europe_rv();
5              mu_pl = pl_mu(10);
6          elseif(body_id == 12)
7              data = io_rv();
8              mu_pl = pl_mu(12);
9          end
10     end

```

Per quanto riguarda l'animazione del Sistema Solare, si è deciso di aggiornare la posizione dei vari pianeti giornalmente per la fase eliocentrica, mentre minuto per minuto per la fase di trasferimento da Giove ad Europa, essendo le grandezze in gioco notevolmente ridotte, soprattutto i periodi orbitali dei satelliti.

Per quanto riguarda la traiettoria della sonda, si è utilizzata la funzione `rv_from_r0v0`, in modo da poter sfruttare l'informazione relativa alle condizioni iniziali derivante dalla risoluzione del problema di Lambert (per la velocità) e dall'approssimazione delle *patched conics* (per la posizione).

La struttura per la generazione della traiettoria sarà dunque la seguente:

```

11
12 number_of_days = caldays( between(begin_date, end_date, 'Days'));
13 time_vector = generate_time(begin_date, end_date);
14
15 for days = 1 : 1 : number_of_days
16
17     [~, body_pos, ~] = body_elements_and_sv(body_id,
18                                             time_vector(days, 1),
19                                             time_vector(days, 2),
20                                             time_vector(days, 3)
21                                             );
22
23     [s_pos, ~] = rv_from_r0v0(r0, v0, days*60*60*24);
24
25 end

```

con `generate_time` funzione di utilità contenuta in `computation_functions` che serve a generare un vettore di date comprese tra una di inizio e una di fine a scelta dell'utente. Si tenga presente che, nelle animazioni, si è tenuto conto soltanto dei pianeti di interesse per la pianificazione della missione, dopo aver osservato che le traiettorie prodotte non potevano in alcun modo portare ad una collisione con nessuno degli altri pianeti del Sistema Solare. Inoltre, soltanto nella fase di animazione si è scelto di eseguire una scalatura delle grandezze in gioco in modo da rendere il plot meno dispersivo e più piacevole a livello visivo. Infine, una nota sulla funzione `body_elements_and_sv`: le effemeridi calcolate da quest'ultima sono valide per un periodo di tempo dal 1901 al 2050, e i vettori di stato dei pianeti del Sistema Solare sono restituiti in un sistema di riferimento eliocentrico con piano xy coincidente con l'eclittica, asse x diretto come il *vernal equinox*, e assi y e z di conseguenza.

Capitolo 4

Pianificazione della missione

Per la pianificazione della missione è stato necessario definire le date di ogni fase esposte nella sezione 1.3 .

A seguito di numerose prove *trial & error*, ci siamo appoggiati in un secondo momento alle date indicative della missione Europa Clipper, la quale ha previsto l'arrivo su Giove entro il 2030; in particolare la data prevista è l'11 Aprile 2030.

Per le restanti date caratteristiche della missione è stato deciso di implementare delle funzioni in MATLAB in accordo con il minimo consumo di carburante e la fattibilità della manovra.

4.1 Funzioni per la scelta delle date

Nella cartella `dates_choice` sono state create tre funzioni :

- `earth_flyby` :

```
26 %-----
27 function [t, dv, r_p] = earth_flyby(t1, r_park, e_park)
28 %-----
29 %Questa funzione restituisce data di partenza dalla Terra
30 %(post flyby), variazione di velocità dell'orbita
31 %interplanetaria ottima e raggio dell'orbita di
32 %parcheggio, data una data di arrivo t1 e un'eccentricità e.
33 %
34 %   Dati in ingresso:
35 %       t1      - data di arrivo su Giove [datetime]
36 %       r_park  - raggio al periasse dell'orbita di parcheggio
37 %                su Giove
38 %       e_park  - eccentricità desiderata per l'orbita di
39 %                parcheggio su Giove
40 %
41 %   Dati in uscita:
42 %       t       - vettore delle date di partenza dalla Terra
43 %       dv      - vettore delle variazioni ottime di velocità
44 %                richiesta per la manovra di parcheggio
45 %       r_p     - vettore dei raggi dell'orbita di parcheggio
```

Fissata la data di arrivo su Giove, l'eccentricità e il raggio al periasse dell'orbita di par-

cheggio, preso dal centro di Giove, otteniamo un vettore di date e di Δv ad esse associate. A partire da una data arbitrariamente fissata, la funzione esegue iterativamente, giorno dopo giorno, delle manovre di Lambert, andando ad ottenere le velocità di arrivo su Giove v_2 . Per ognuna di esse, viene eseguita la funzione `capture_hyp` per valutare i Δv della fase di cattura su Giove.

Fra questi, vengono salvati in un array solo quelli di valore inferiore a 4.75; tale valore è stato scelto empiricamente a seguito di una serie di test che ci hanno permesso di capire l'ordine di grandezza dei Δv in gioco.

- `mars_flyby`

```

47 %-----
48 function [t, dv, r] = mars_flyby(t1, t2)
49 %-----
50 %Questa funzione trova una data di partenza pre Lambert in
51 %modo da far coincidere le velocità relative di entrata
52 %e di uscita
53 %
54 %  Dati in ingresso:
55 %      t1 - data del flyby (fissata) su Terra
56 %      t2 - data di arrivo del Lambert post flyby su Giove
57 %
58 %  Dati in uscita:
59 %      t      - vettore dei tempi plausibili di partenza da
60 %               planet_id0 per il Lambert pre flyby
61 %
62 %      dv      - deltav del flyby su planet_id1
63 %      r      - raggio al periasse dell'iperbole di flyby

```

Vengono date in ingresso le date di flyby sulla Terra e di arrivo su Giove. Ciò implica che il vettore di velocità relativa in uscita dal flyby terrestre sia fissato. Si ottiene un vettore di date e di Δv di flyby ad esse associate.

A partire da una data di flyby su Marte arbitrariamente fissata, la funzione esegue iterativamente, giorno dopo giorno, delle manovre di Lambert andando ad ottenere le velocità di arrivo sulla Terra v_2 . Per ogni valore ottenuto, viene valutata la differenza tra le velocità relative di ingresso e uscita dal flyby della Terra. Se la differenza risulta inferiore a una certa tolleranza `toll`, queste vengono impilate in un array, così come le date ad esse associate. Si tenga presente che, a livello teorico, un flyby vero e proprio dovrebbe avere differenza nulla tra le due velocità relative.

- earth_departure

```

66 %-----
67 function [t, dv, r, dv_esc_earth] = earth_departure(t1, t2)
68 %-----
69 % Questa funzione trova una data di partenza per Lambert pre
70 %flyby su Marte in modo da far coincidere le velocità '
71 %relative di entrata e uscita, minimizzando
72 %il deltav di uscita dalla Terra.
73 %
74 %  Dati in ingresso:
75 %      t1 - data del flyby (fissata) su Marte
76 %      t2 - data di arrivo del Lambert post flyby su Terra
77 %
78 %  Dati in uscita:
79 %      t                - tempo plausibile di partenza da
80 %                        planet_id0 per il Lambert pre flyby
81 %      dv               - deltav del flyby su planet_id1
82 %      r                - Raggio al periasse dell'iperbole di
83 %                        flyby
84 %      dv_esc_earth     - deltav di uscita dalla terra

```

Vengono date in ingresso le date di flyby su Marte e sulla Terra. Ciò implica che il vettore di velocità relativa in uscita dal flyby marziano sia fissato. Otteniamo un vettore di date e di Δv di flyby ad esse associate.

A partire da una data di partenza dalla Terra arbitrariamente fissata, la funzione esegue iterativamente, giorno dopo giorno, delle manovre di Lambert andando ad ottenere le velocità di arrivo su Marte v_2 . Per ogni valore ottenuto, viene valutata la differenza tra le velocità relative di ingresso e uscita dal flyby di Marte. Se la differenza risulta inferiore a una certa tolleranza `toll`, queste vengono impilate in un array, così come le date ad esse associate.

La funzione restituisce, per ogni data plausibile, tramite l'esecuzione di `escape_hyp`, i Δv necessari per entrare sull'iperbole di uscita dalla SOI terrestre.

4.2 Orbita di parcheggio e determinazione delle date di interesse

Le precedenti funzioni si integrano a vicenda all'interno dello script `mission_dates`, che riveste importanza fondamentale nella determinazione delle date. In esso è stata fissata la data di arrivo e l'eccentricità dell'orbita di parcheggio.

Quest'ultima è stata scelta pari a 0.6, in modo da non avere un Δv elevato. Facendo diverse prove avevamo infatti notato che più l'eccentricità tendeva a zero, maggiore era il dispendio energetico durante la fase di cattura.

Successivamente abbiamo fissato un valore di apoasse a 600.000 *km* in modo che non ci fossero problemi di intersezione tra l'orbita di parcheggio e l'orbita di Europa, la quale ha raggio medio di circa 670.000 *km*. Di conseguenza è stato calcolato il raggio al periasse con la formula

seguinte:

$$r_p = \left(\frac{1 - e_{park}}{1 + e_{park}} \right) r_a$$

Da r_p ricaviamo che la sonda deve orbitare ad un'altezza di 80.000 *km* (perigiovio) rispetto alla superficie gioviana.

Ricapitolando: applicando la funzione `earth_flyby`, viene trovata la data di flyby sulla Terra, che può a sua volta essere passata come input della funzione `mars_flyby`, ottenendo una data plausibile per il primo flyby su Marte.

Le date relative ai sorvoli ravvicinati vengono selezionate sfruttando come criterio quello del massimo Δv ottenibile da tale manovra.

Le due date vengono infine impiegate da `earth_departure` per definire un vettore di date di partenza plausibili da cui viene selezionata quella che richiede il minimo Δv per immettersi sull'iperbole di fuga. In conclusione otteniamo:

- **16/10/2024**: data di partenza dalla Terra;
- **03/03/2025**: data di flyby su Marte;
- **07/12/2026**: data di flyby sulla Terra;
- **11/04/2030**: data di arrivo su Giove.

Si noti che per la pianificazione della missione è stato dunque utilizzato un approccio a ritroso per il quale, una volta stabiliti determinati parametri di arrivo su Giove, sono stati ricavati i dati di interesse per le manovre precedenti all'arrivo, fino a determinare quella di partenza dalla Terra.

Capitolo 5

Svolgimento della missione

5.1 Fase iniziale: fuga dalla Terra

È richiesto che la missione si svolga interamente sul piano dell'eclittica. Pertanto la prima manovra da effettuare è un cambiamento del piano di 23.27° .

Ciò richiede una spesa di carburante non migliorabile, rappresentata dal valore del Δv seguente:

$$\Delta v = 2v \cdot \sin\left(\frac{\theta}{2}\right)$$

dove v è la velocità tangenziale della sonda sull'orbita circolare geostazionaria di altezza 200 km:

$$v = \sqrt{\left(\frac{\mu_{Earth}}{200 + R_{Earth}}\right)}$$

che risulta essere pari a 7.7884 km/s. Il Δv necessario risulta quindi essere **3.1587 km/s**. In Figura 5.1 è riportato il plot di tale manovra, ottenuto lanciando il file `change_orbital_plane` nella cartella `animation_functions`.

La fase di fuga è svolta nello script `escape_hyp`. L'idea che sta dietro questo script è quella di fornire un Δv alla sonda che le permetta di immettersi su una determinata iperbole di fuga. In particolare la velocità che la sonda dovrà avere in ingresso a tale iperbole è calcolata a partire dalla velocità v_∞ , ovvero la velocità relativa della sonda in uscita dalla SOI della Terra. In particolare, si noti che:

$$v_\infty = \|\mathbf{v}_p - \mathbf{v}_s\|$$

dove con \mathbf{v}_p si è indicata la velocità (assoluta) del pianeta, mentre con \mathbf{v}_s la velocità *assoluta* della sonda. Tale v_∞ dovrà dunque coincidere con quella necessaria per effettuare un trasferimento alla Lambert Terra-Marte. Il plot in Figura 5.2 è ottenuto con `escape_hyp_plot`.

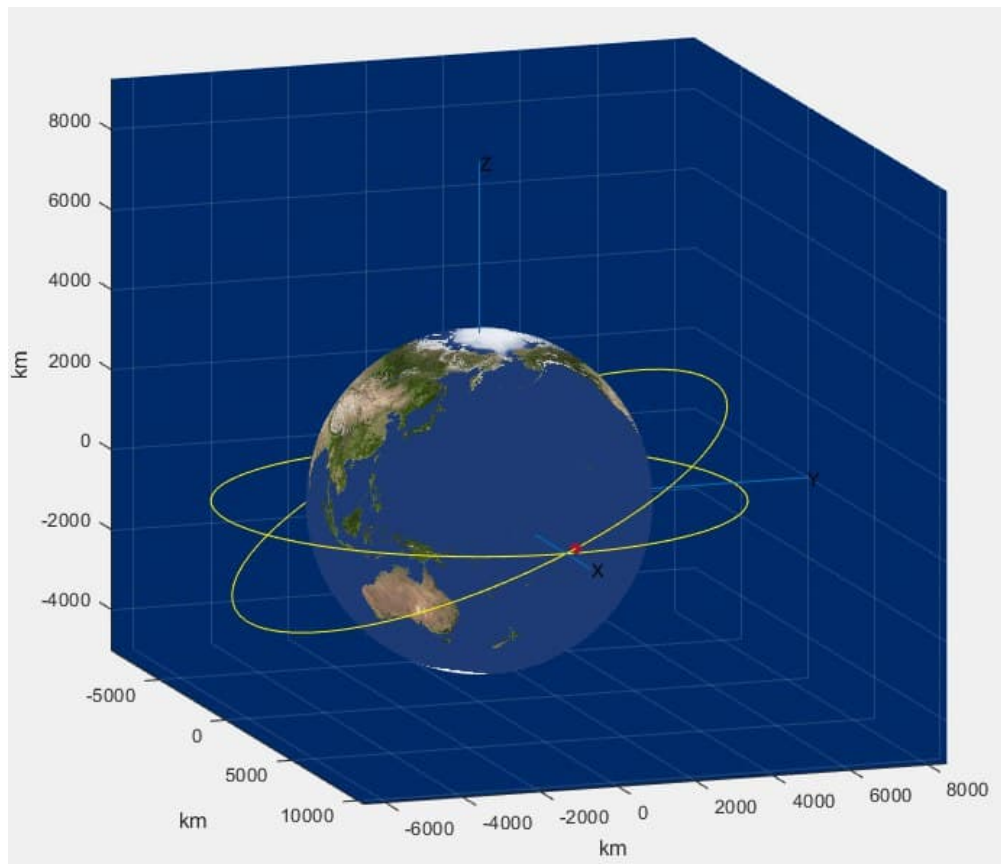


Figura 5.1: Cambiamento dell'orbita sul piano dell'eclittica

```

86 %-----
87 function [delta_v, coe] =
88     escape_hyp (body_id, V1, h_park, dep_time)
89 %-----
90 %Questa funzione calcola le caratteristiche dell'iperbole di
91 %uscita dalla SOI del pianeta body_id fissata un'orbita di
92 %parcheggio CIRCOLARE di raggio r_p attorno al pianeta body_id
93 %
94 %   Dati in ingresso:
95 %       V1      - velocita' eliocentrica della sonda all'uscita
96 %                della SOI      [km/s]
97 %       time    - data di inizio trasferimento interplanetario
98 %                  [datetime]
99 %       h_park  - altezza dell'orbita di parcheggio dal pianeta
100 %                body_id      [km]
101 %
102 %   Dati di uscita:
103 %       delta_v - deltaV necessario per entrare nell'iperbole
104 %                di uscita      [km/s]
105 %       e       - eccentricita'
106 %       a       - semiasse maggiore      [km]
107 %       theta   - angolo di partenza della sonda rispetto
108 %                all'asse x terrestre [deg]
109
110     v_inf = norm((V1-v),2);
111
112     %velocita' di parcheggio
113     v_park = sqrt(mu_p/r_p);
114
115     %velocita' di ingresso nell'iperbole
116     v_hyp = sqrt(2*(mu_p)/r_p + (v_inf)^2);

```

Questa prima fase risulta necessaria per uscire dalla sfera di influenza terrestre ed iniziare il trasferimento interplanetario. Terminata questa parte avremo a disposizione la velocità eliocentrica necessaria per raggiungere Marte alla data stabilita.

5.2 Prima parte del volo: fly-by

Con la funzione `lambert` si ricavano le velocità eliocentriche che la sonda ha alla partenza e all'arrivo della manovra, fissate le posizioni iniziale e finale della sonda e la durata del trasferimento. Queste velocità sono quelle che ci hanno vincolato nella caratterizzazione delle manovre.

```

116 %-----
117 function [V1, V2] = lambert(R1, R2, t, string)
118 %-----

```

Nel primo caso, R1 corrisponde alla posizione della Terra alla data di partenza e R2 corri-

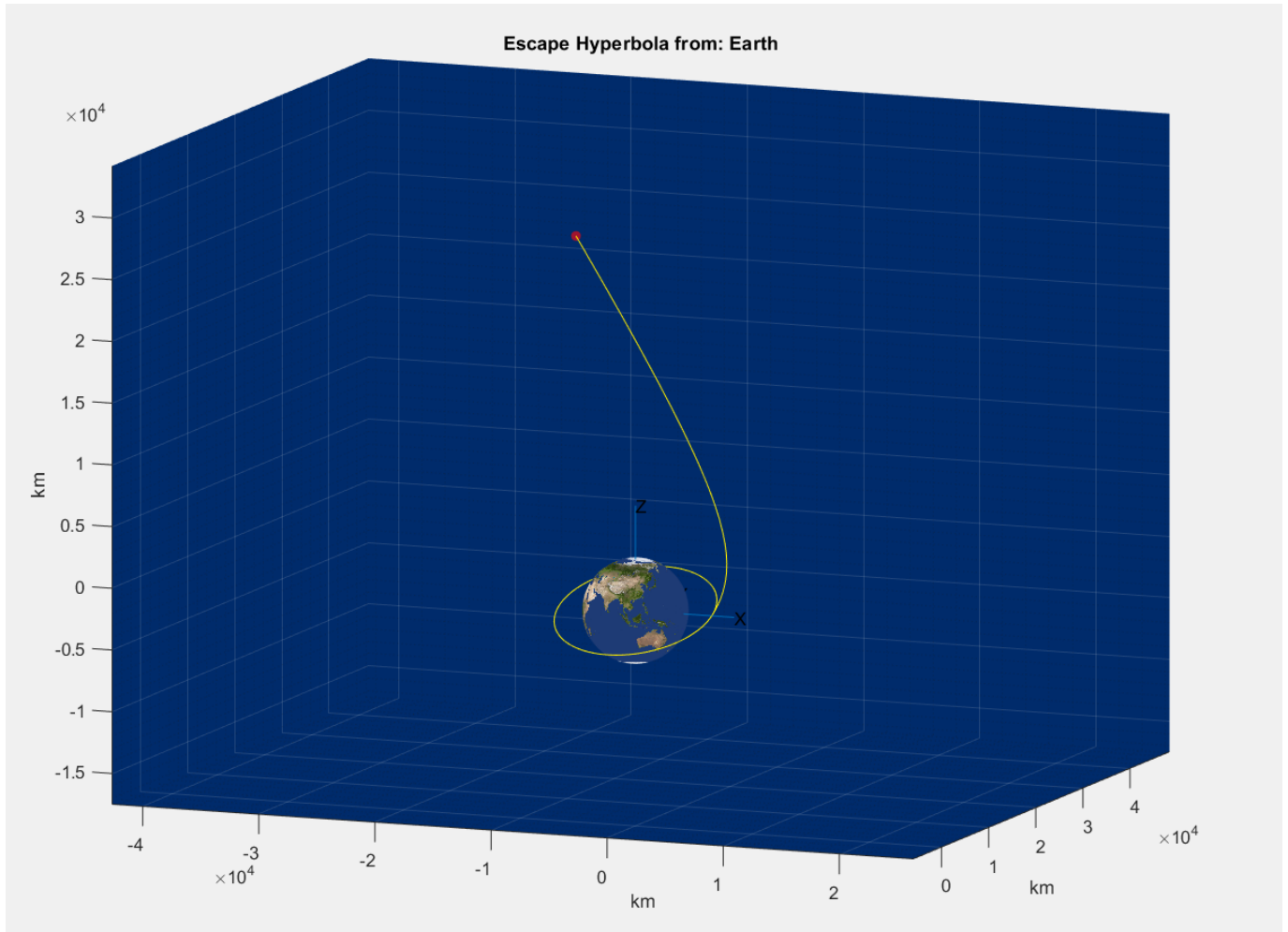


Figura 5.2: Iperbole di fuga - Terra

sponde alla posizione di Marte alla data del flyby.

La manovra di flyby sfrutta il campo gravitazionale del pianeta per eseguire un cambio di traiettoria a zero consumo. Durante questa fase la sonda entra con una certa velocità all'interno della SOI del pianeta, sufficiente da permetterle di compiere una traiettoria iperbolica attorno al corpo celeste con un raggio al periasse tale da evitare la collisione con quest'ultimo. Dalla differenza tra velocità della sonda e del pianeta otteniamo la v_∞ (velocità relativa della sonda), grandezza fondamentale per calcolare i parametri caratteristici dell'iperbole di flyby.

Tra questi citiamo:

- raggio al periasse

$$r_p = \left(\frac{\mu}{v_\infty^2} \right) \cdot \left(\frac{1}{\sin(\delta/2) - 1} \right)$$

- eccentricità

$$e = 1 + \left(\frac{r_p \cdot v_\infty^2}{\mu} \right)$$

ove δ corrisponde al *turn angle* del flyby, ovvero l'angolo compreso fra le due traiettorie,

di entrata e di uscita:

$$\delta = 2\sin^{-1}\left(\frac{1}{e}\right)$$

Queste grandezze sono calcolate all'interno dello script `flyby`.

5.2.1 Primo Flyby - Marte

È stata creata una funzione MATLAB che consentisse l'esecuzione del flyby.

```

119 %-----
120 function [deltav, deltav_inf, e, a, delta, r_p] =
121         flyby ( body_id, v1, V, v2, plot)
122 %-----
123 %Questa funzione calcola i parametri dell'iperbole di flyby
124 %e il deltav dovuto alla manovra.
125 % Dati in ingresso:
126 %   v1 - velocita' della sonda in entrata al SOI del pianeta
127 %       (da Lambert);
128 %   V  - velocita' del pianeta (da body_elements_and_sv);
129 %   v2 - velocita' in uscita da SOI del pianeta (da Lambert);
130 %   plot - flag che, se attivo (ovvero uguale a 1),
131 %          fa eseguire il plot della manovra
132 %
133 % Dati in uscita:
134 %   deltav - accelerazione dovuta al flyby;
135 %   e      - eccentricita' dell'iperbole di flyby;
136 %   a      - semiasse maggiore dell'iperbole di flyby;
137 %   delta  - angolo caratteristico di flyby;
138 %   r_p    - raggio del periasse dell'iperbole di flyby;

```

Avendo pianificato le manovre di Lambert successive al flyby, abbiamo potuto sfruttare la conoscenza delle velocità eliocentriche in ingresso e in uscita dalla SOI di Marte. Con esse siamo risaliti alla velocità v_∞ relativa della sonda di ingresso e uscita, e quindi successivamente, ai parametri dell'iperbole di flyby.

In particolare abbiamo calcolato eccentricità, raggio al periasse, semiasse maggiore e *turn angle*. Inoltre abbiamo anche aggiunto un check sul tipo di flyby effettuato, per discriminare un passaggio a monte o a valle del pianeta, e quindi un'accelerazione o decelerazione della sonda a seguito della manovra.

```

140     if(norm(v1) > norm(v2))
141         side = 'leading';
142     else
143         side = 'trailing';
144     end

```

Questo ci è risultato utile in fase di *debugging*.

```

145         %Il Flyby su Marte deve essere da davanti
146     if norm(v2) < norm(v_dep)
147         t0 = t0 + 1;
148         n = n + 1;
149         continue;
150     end

```

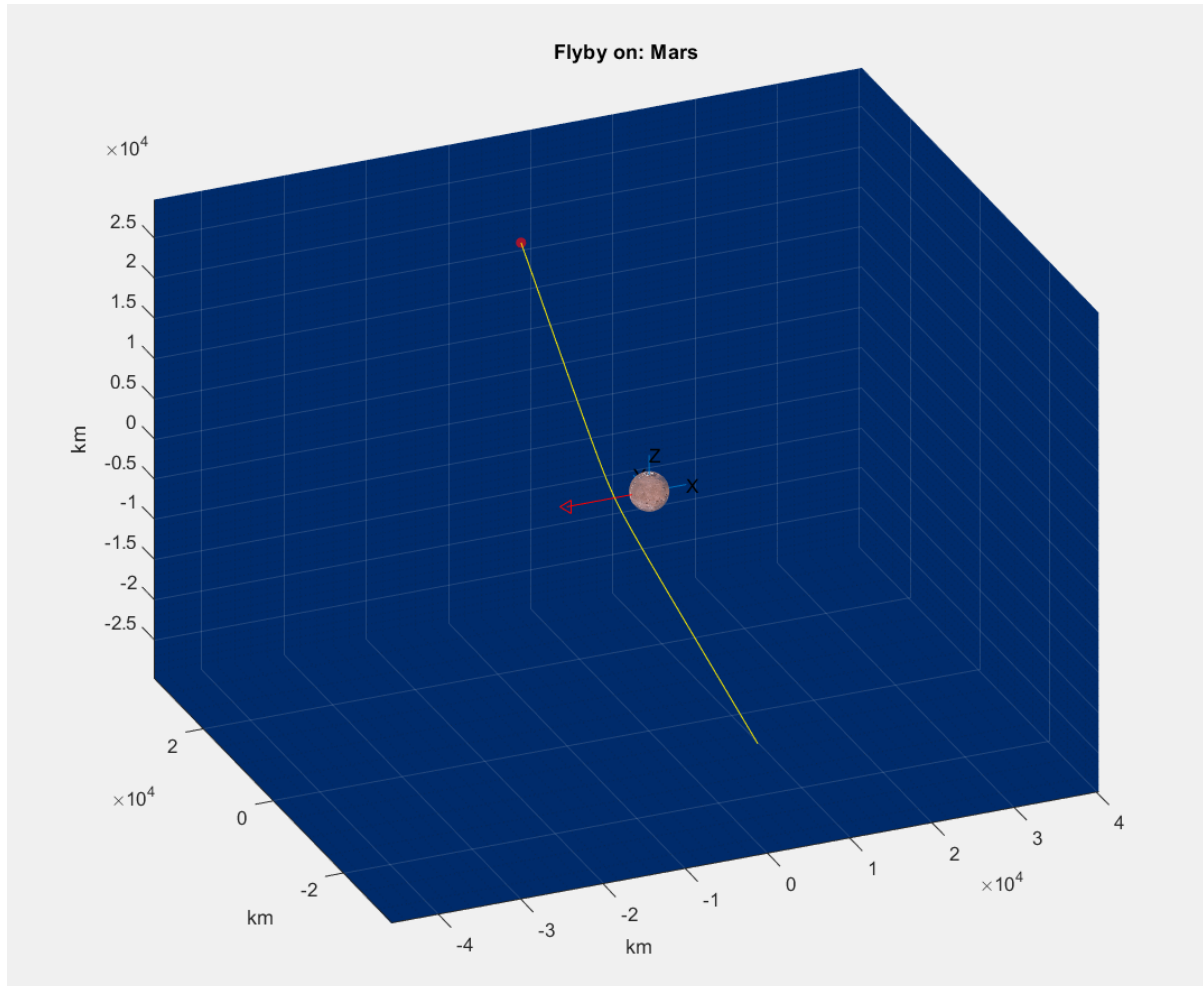


Figura 5.3: Flyby su Marte con passaggio a valle

5.2.2 Secondo Flyby - Terra

Analogamente, la procedura esposta nella sezione precedente è stata adottata anche per il flyby sulla Terra riadattando le date, con l'unica differenza che in questo caso non è stato necessario imporre un check sul tipo di passaggio di flyby.

In generale, la difficoltà riscontrata con il flyby, è stata la scelta di una data efficace che permettesse l'esecuzione di tale manovra. Inoltre, affinché la manovra di fionda gravitazionale avvenga correttamente è necessario che siano rispettate alcune caratteristiche geometriche.

In particolare, se decidiamo che la traiettoria sia *dietro* il pianeta, si deve necessariamente verificare che l'angolo ϕ tra la velocità relativa della sonda e la velocità del pianeta sia compreso

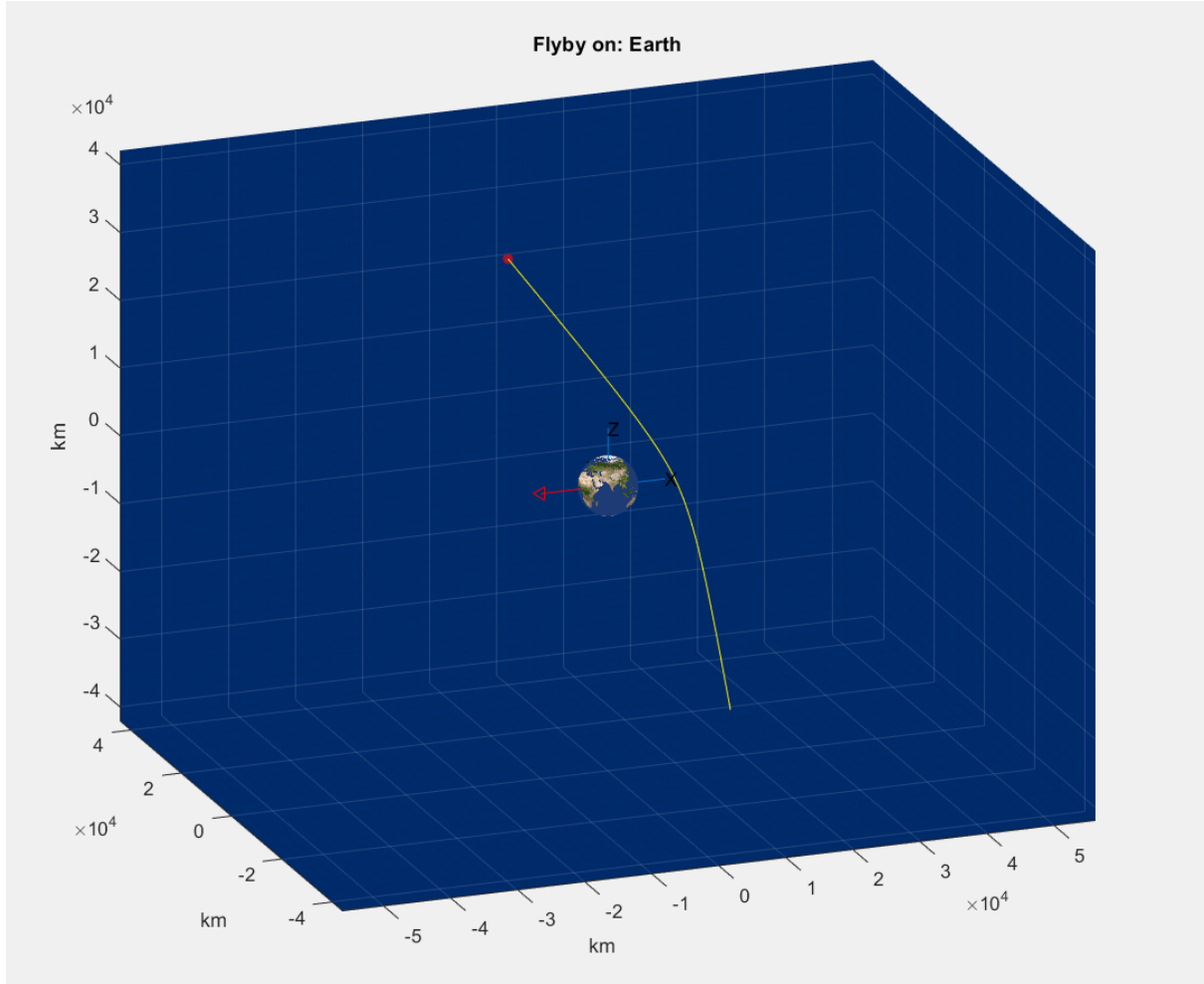


Figura 5.4: Flyby su Terra con passaggio a monte

tra $\frac{\pi}{2}$ e π . In questo modo si impone implicitamente una velocità relativa della sonda tale da poter effettuare un **passaggio a monte** del pianeta (vedi Figura 5.5).

$$\phi \in \left[\frac{\pi}{2}, \pi \right]$$

Nel caso di flyby *davanti* al pianeta la condizione sull'angolo ϕ è che sia compreso tra 0 e $\frac{\pi}{2}$, in questo modo la velocità relativa sarà orientata in modo da permettere un **passaggio a valle** del pianeta (vedi Figura 5.6).

$$\phi \in \left[0, \frac{\pi}{2} \right]$$

Altro vincolo da soddisfare, e imposto nel codice per il flyby, è che le velocità relative in ingresso e in uscita dalla SOI siano uguali.

Infatti questa tecnica permette di effettuare una variazione di traiettoria della sonda, lasciando inalterate le velocità relative di quest'ultima rispetto al pianeta. Quello che varia a seguito del flyby è la velocità eliocentrica che la sonda possiede. Il problema principale riscontrato nell'implementazione di questa manovra sta nel riuscire a rispettare la condizione sull'angolo tra velocità pianeta e velocità eliocentrica della sonda, che permetta un passaggio a monte del pianeta. Inoltre si deve verificare che i parametri della traiettoria iperbolica siano fisicamente realizzabili, in particolare dobbiamo ottenere un raggio al periasse che non sia inferiore all'altezza dell'atmosfera del pianeta e che allo stesso tempo non esca dalla sua sfera di influenza.


```

151 [~, r_jupiter1, ~] = body_elements_and_sv(5, y3, m3, d3, 0, 0, 0)
    ;
152 T_c = between (begin_date_3, end_date_3, 'Days');
    %[days]
153 t_c = (caldays(T_c)) * 24 * 3600;
    %[sec]
154 [V3_b, V4] = lambert(r_earth2, r_jupiter1, t_c);

```

Da notare che, avendo progettato tutta la missione a ritroso, **V3_b** coincide col vettore velocità post-fly by su Terra.

La traiettoria complessiva risultante è mostrata in Figura 5.7.

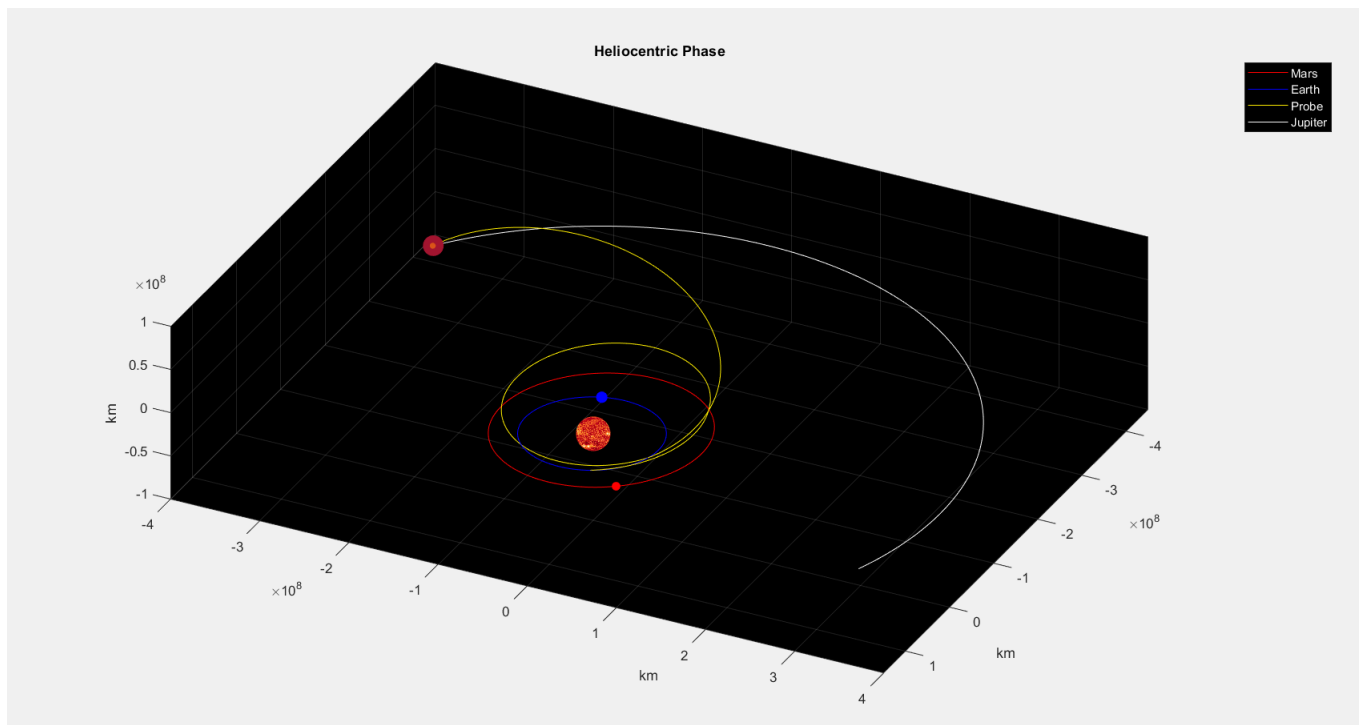


Figura 5.7: Traiettoria di trasferimento complessiva

5.4 Terza parte del volo: verso Europa

5.4.1 Ingresso SOI di Giove e parcheggio

Come anticipato, la data di arrivo su Giove è stata fissata all'11 Aprile 2030.

Una volta giunti alla SOI di Giove, inizia la manovra per portarsi sull'orbita di parcheggio, descritta nella sezione 4.2.

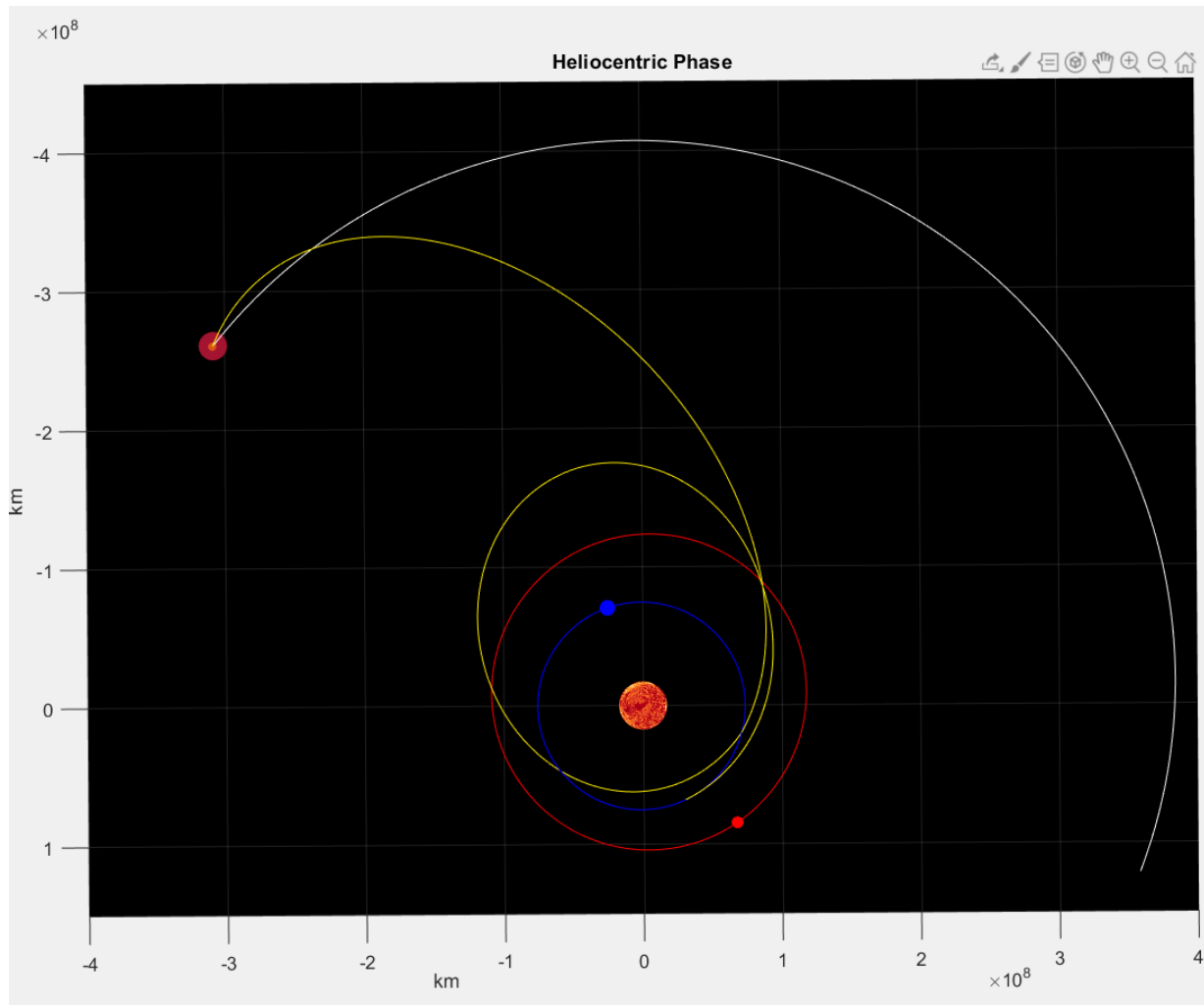


Figura 5.8: Traiettoria di trasferimento complessiva. Vista dall'alto

La funzione `capture_hyp` ci consente di effettuare la manovra di cattura e di parcheggio della sonda su un'orbita di raggio ed eccentricità arbitrarie intorno a Giove, note la data e la velocità di arrivo sul pianeta.

In particolare, la velocità di arrivo V_4 è nota dal trasferimento alla Lambert Terra-Giove.

```

1 %-----
2 function [delta_v, r_p] =
3     capture_hyp(body_id, Va, arr_time, varargin)
4 %-----
5 %Questa funzione calcola l'iperbole di avvicinamento della sonda
6 %in arrivo al pianeta identificato dal body_id e la fa rimanere
7 %in orbita attorno ad esso ad una distanza rp (distanza che
8 %ottimizza l'uso di carburante) - presa dal centro del pianeta.
9 %L'orbita scelta dovra' ovviamente essere chiusa.
10 %
11 %    Dati in ingresso:
12 %        Va        - velocita' eliocentrica della sonda, in arrivo
13 %                    alla SOI del pianeta target [km/s]
14 %        arr_time - data di arrivo alla SOI del body_id [datetime]
15 %        varargin - campo equivalente a due ingressi:
16 %        varargin{1} = h: input per definire l'altezza
17 %                    dell'orbita di cattura. L'utente puo' scegliere di
18 %                    utilizzare uno dei due casi:
19 %                    (a) varargin = 'opt': in tal caso questa funzione
20 %                        usa il raggio al periasse (preso dal centro
21 %                        del pianeta) che ottimizza il deltaV;
22 %                    (b) float varargin: altezza dell'orbita di
23 %                        parcheggio, scelta dall'utente.          [km]
24 %        varargin{2}: secondo input; in entrambi i casi, questo
25 %                    ingresso serve per specificare l'eccentricita'
26 %                    desiderata dell'orbita di parcheggio e_park
27 %
28 %    Dati di uscita:
29 %        delta_v - deltaV totale della manovra                [km/s]
30 %        r_p     - raggio al periasse dell'iperbole di ingresso
31 %                    e, quindi, dell'orbita di parcheggio [km]
32
33 [delta_v, rp_jupiter] = capture_hyp(5, V4, end_date_3, 8e4, 0.6);

```

Con essa vengono calcolati i parametri dell'iperbole di ingresso e dell'ellisse di parcheggio:

$$e_{hyp} = 1 + \frac{r_p \cdot v_{\infty}^2}{\mu_{jupiter}}$$

$$a_{park} = \frac{r_p}{(1 - e_{park})}$$

con r_p raggio al periasse, che coincide col punto in cui viene dato l'impulso per portarsi dall'iperbole all'ellisse. Per la prima, in corrispondenza di tale punto si ha per la sonda una velocità pari a

$$v_{hyp} = \sqrt{\left(\frac{v_{\infty}^2 + 2\mu_{jupiter}}{r_p} \right)}$$

Mentre per l'ellisse, nel medesimo punto si ha :

$$v_{park} = \sqrt{\left(\frac{\mu_{jupiter}(1 + e_{park})}{r_p} \right)}$$

Perciò in conclusione il Δv per tale fase risulta :

$$\Delta v = |v_{hyp} - v_{park}|$$

e la sua esecuzione è mostrata nella Figura 5.9.

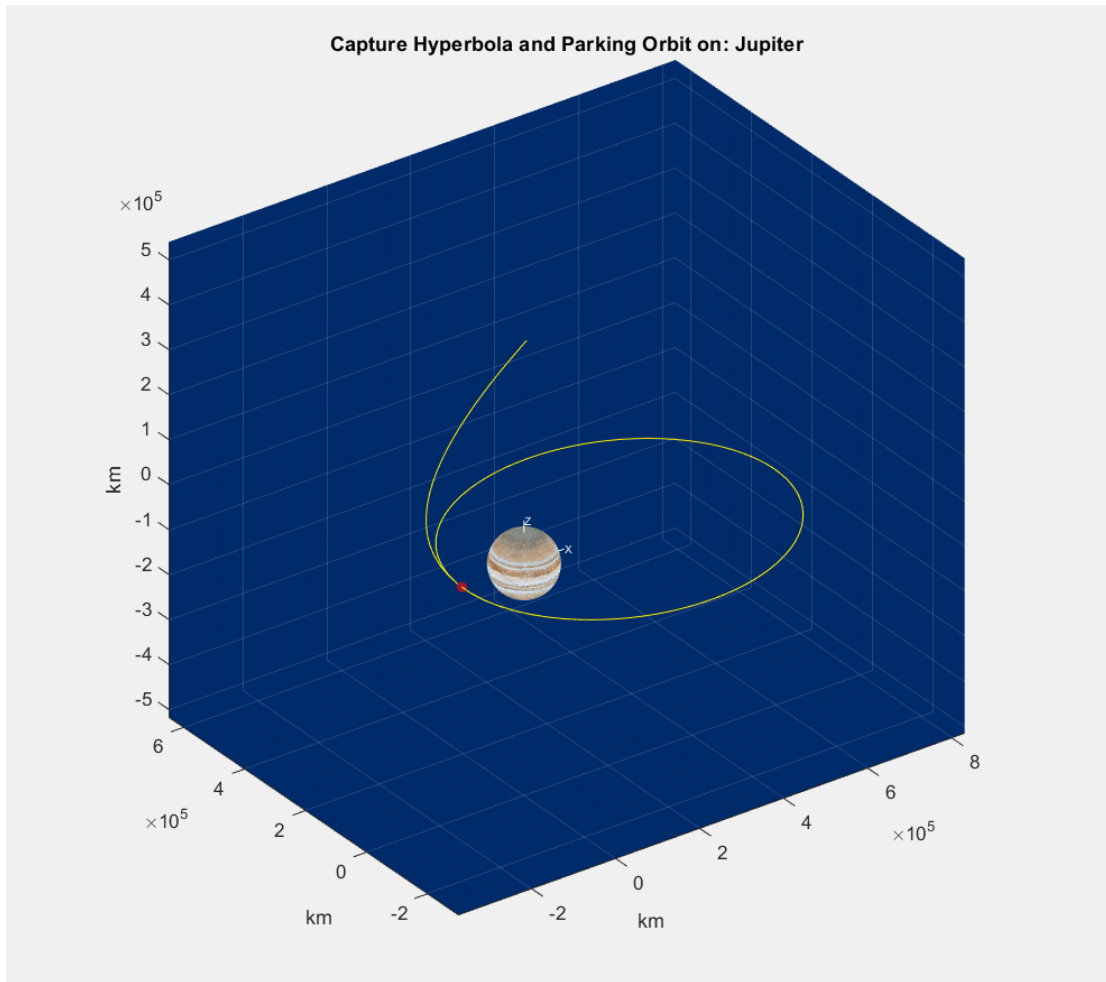


Figura 5.9: Ingresso e parcheggio su Giove

Inizialmente era stata effettuata una prova utilizzando l'opzione 'opt' in ingresso alla funzione `capture_hyp`, la quale calcola il raggio al periasse ottimo in termini di consumi energetici. I valori ottenuti con questa tecnica in termini di raggio al periasse, però, risultavano maggiori del raggio della sfera di influenza gioviana, calcolata dalla funzione `compute_soi`.

5.4.2 Trasferimento su Europa

Prima di trasferirci su Europa, attendiamo un certo intervallo di tempo, orbitando sull'ellisse di parcheggio.

L'idea iniziale era di effettuare il trasferimento con una traiettoria alla Hohmann. Per fare questo, saremmo dovuti partire al periasse dell'ellisse (in accordo ad una richiesta minore di Δv) in un certo istante che ci avrebbe permesso di giungere su Europa alla fine del trasferimento. Ciononostante abbiamo osservato che non esisteva una data accettabile che rispondesse ai requisiti di partenza al periasse e del phasing corretto. Per questo è stato deciso di optare per una manovra alla Lambert.

Per eseguire tale manovra abbiamo considerato diversi tempi di attesa e diverse durate della manovra da valutare all'interno di un ciclo `for` dello script `europa_phasing` come già accennato nella sezione 2.2.1.

La partenza della sonda è comunque sempre mantenuta al periasse dell'ellisse di parcheggio. Infatti il tempo di attesa viene preso come multiplo di un periodo.

Per l'ingresso e il parcheggio su Europa, è stata nuovamente utilizzata la funzione `capture_hyp` impostando a 100, l'altezza dell'orbita dalla superficie del satellite

```
34 [delta_v6, ~] = capture_hyp(10, v2_probe_eur, begin_date, 1, 100,
    0, v2_eur);
```

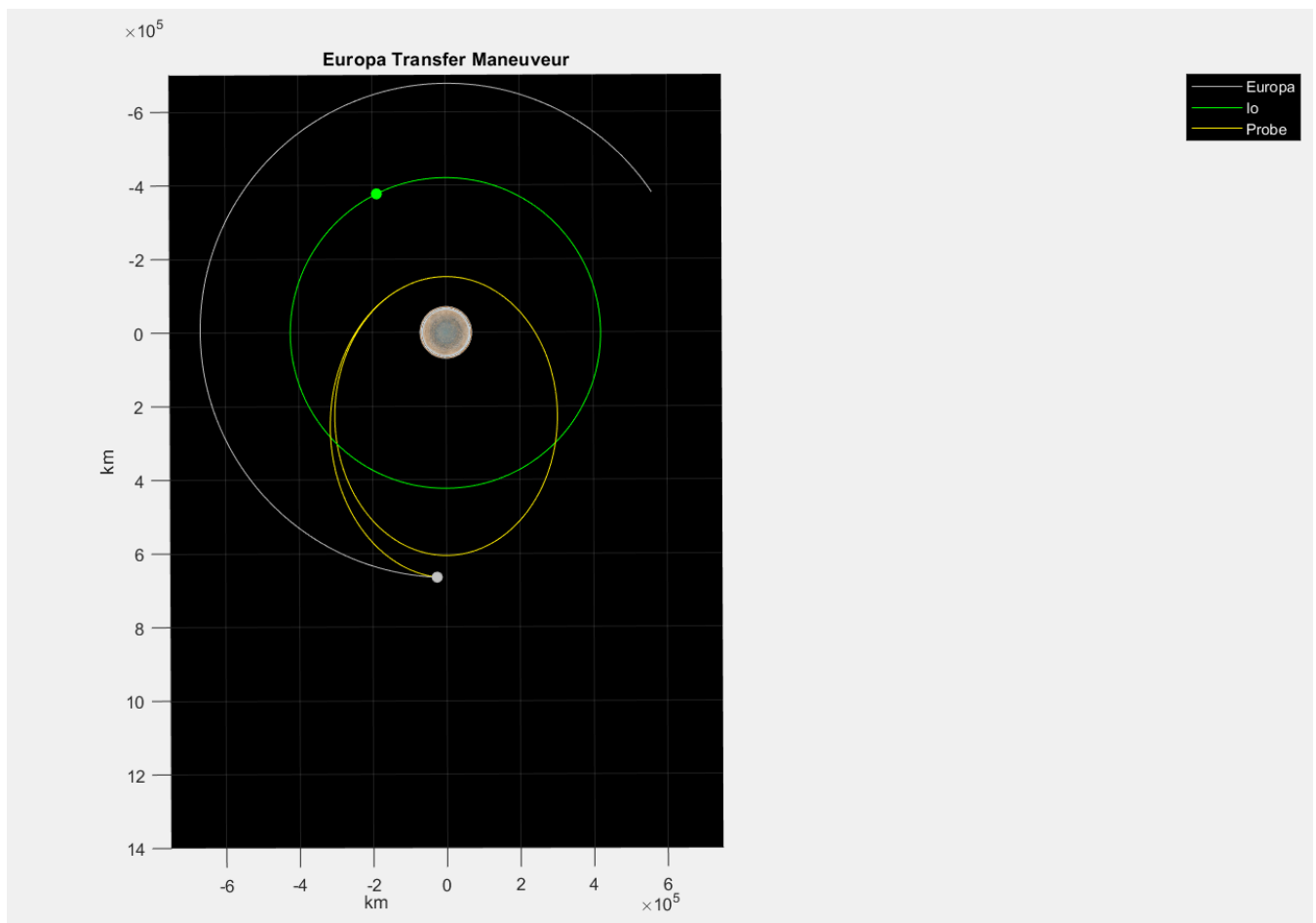


Figura 5.10: Trasferimento Giove - Europa

Una particolare attenzione è stata posta anche sulla parte grafica di questa fase della missione. Impiegando le stesse funzioni usate per i precedenti plot, le orbite dei satelliti risultavano dei poligoni. Ciò è dovuto al fatto che i moti di rivoluzione di Io ed Europa attorno a Giove sono di due ordini di grandezza minori rispetto a quelli della maggior parte dei pianeti attorno al Sole. Si parla di circa 1.7 giorni per Io e 3.5 giorni per Europa.

Quindi il passo di generazione della traiettoria usato in `rv_from_r0v0` non consentiva di ottenere una traiettoria sufficientemente accurata. Il problema è stato risolto integrando con un algoritmo a step fisso pari a un minuto.

Importante infine osservare che, per il calcolo del Δv necessario per l'iperbole di cattura, è stato passato come input aggiuntivo alla funzione `capture_hyp` la velocità di Europa, in modo da ottenere una maggiore precisione rispetto a quella giornaliera fornita dalle effemeridi (prelevate a loro volta tramite `body_elements_and_sv`).

Capitolo 6

Missione alternativa

6.1 Fase interplanetaria

Una via alternativa per raggiungere Giove è quella di utilizzare un'unica manovra a due impulsi di Hohmann.

La manovra di Hohman consiste in un trasferimento ellittico con partenza della sonda al periasse della traiettoria stessa (posizione che sarà occupata dalla Terra), e arrivo al suo apoasse (posizione che sarà occupata da Giove). Ciò implica il soddisfacimento di certi vincoli sulla posizione angolare nell'orbita e sulle velocità dei due corpi celesti in gioco.

Posto che la posizione angolare a cui la manovra inizia è indicata con θ_H , ma che al tempo iniziale considerato t_0 , la Terra si trova ad un certo θ_0 rispetto a Giove, allora:

$$\Delta\theta = \theta_0 - \theta_H$$

con:

$$\theta_H = \pi \left[1 - \sqrt{\left(\frac{1 - r_E/r_J}{2} \right)^3} \right]$$

e:

$$\theta_0 = \theta_{J0} - \theta_{E0}$$

distanza angolare da annullare nell'arco del tempo di attesa Δt_a , prima di poter fornire il primo impulso della manovra. Durante questo tempo di attesa infatti, la Terra percorrerà uno spostamento angolare $\theta_E = \omega_E \Delta t_a$ mentre Giove percorrerà un $\theta_J = \omega_J \Delta t_a$.

Da cui, posto T_E e T_J i periodi eliocentrici di Terra e Giove, definiti nel file `initializeEJE`, si ha :

$$\Delta\theta = \theta_E - \theta_J = (\omega_E - \omega_J) \Delta t_a = \left(\frac{2\pi}{T_E} - \frac{2\pi}{T_J} \right) \Delta t_a$$

Dall'equazione sopra si ricava il tempo di attesa:

$$\Delta t_a = \frac{\Delta\theta}{2\pi \left(\frac{1}{T_E} - \frac{1}{T_J} \right)}$$

Una volta partiti, la traiettoria è univocamente definita e il tempo di volo del trasferimento è :

$$\Delta t_H = \frac{\pi}{\sqrt{\mu}} \left(\frac{r_E + r_J}{2} \right)^{\frac{3}{2}}$$

La funzione `hohmann_phasing` calcola il tempo di attesa Δt_a e il tempo totale di trasferimento Δt_H , fissati il tempo iniziale t_0 e i due corpi intercettore e bersaglio (rispettivamente Terra e Giove).

```

35 %-----
36 function [delta_t_A, delta_t_H, delta_t_tot] =
37         hohmann_phasing(dep_body, arr_body, t0)
38 %-----

```

Con essa, si ricavano dapprima le posizioni iniziali dei due corpi al tempo iniziale, con semplici considerazioni geometriche tra le componenti del vettore posizione degli stessi.

```

39 %posizione iniziare di dep_body
40 TA_dep_body = atan2(r_dep(2), r_dep(1)); %[rad]
41 %posizione iniziale di arr_body [rad]
42 TA_arr_body = atan2(r_arr(2), r_arr(1)); %[rad]

```

Successivamente si applicano le formule sopra esposte.

Per una valutazione più generale del trasferimento, viene calcolato anche il dispendio energetico, utilizzando la funzione `hohmann_transfer`. Le ipotesi da mantenere per entrambi gli script sono quelle di complanarità delle orbite dei pianeti, e di circolarità di queste.

La prima ipotesi può essere considerata una buona approssimazione in quanto l'orbita gioviana giace su un piano di inclinazione 1.31° rispetto all'eclittica (piano di lavoro). Non a caso, nel graficare questa parte e mantenere coerente il plot, Giove è stato plottato anch'esso sul piano dell'eclittica.

Il Δv necessario a questo tipo di fase interplanetaria è dato dalla somma dei due Δv usati per i due impulsi dati rispettivamente al periasse e all'apoasse della semi-ellisse di trasferimento. :

$$\Delta v_{H_{first}} = \sqrt{\frac{2 \cdot (r_J/r_E)}{1 + (r_J/r_E)}} - 1$$

$$\Delta v_{H_{second}} = \sqrt{\frac{1}{r_J/r_E}} - \sqrt{\frac{2}{r_J/r_E \cdot (1 + r_J/r_E)}}$$

NB: I due, sono valori normalizzati alla velocità di percorrimento dell'orbita circolare di partenza v_{c1} . Per una analisi e successivo confronto di consumo del carburante infatti, in `hohmann_transfer_plot`, ci si riconduce a un Δv non normalizzato, semplicemente moltiplicando il valore ottenuto, con v_{c1} (dato che stiamo approssimando ad orbite circolari).

$$v_{c1} = \sqrt{\frac{\mu}{r_E}}$$

La traiettoria della sonda è mostrata in Figura 6.1.

6.2 Fase di Fuga - Terra

Date di partenza e di arrivo sono pertanto diverse da quelle trovate per la missione originale. Utilizzando la funzione `interplanetary`, ricaviamo quindi la velocità eliocentrica che la sonda deve avere in uscita alla SOI terrestre.

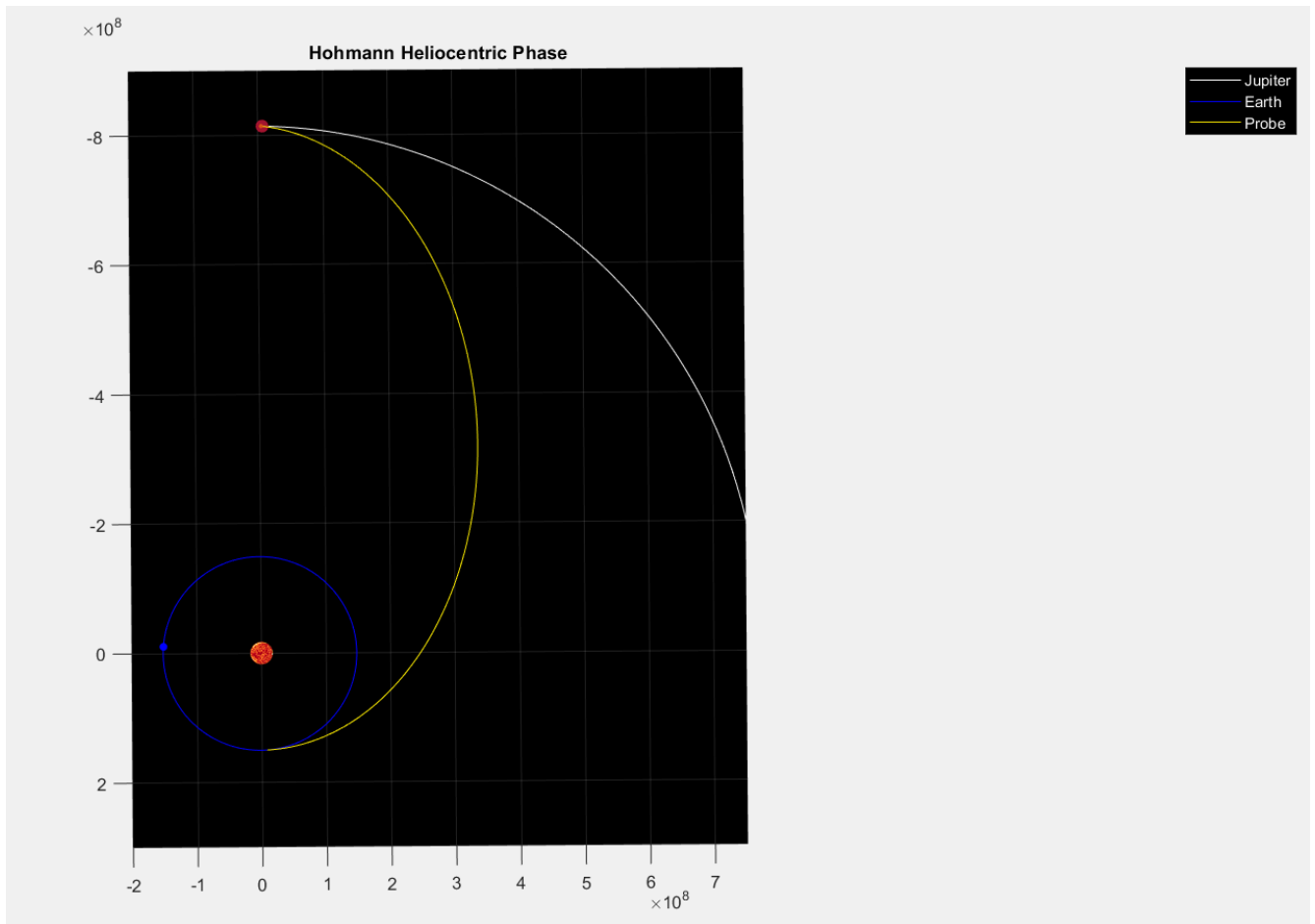


Figura 6.1: Traiettoria di trasferimento alla Hohmann, Terra - Giove

Questa informazione ci è servita per poter determinare a ritroso l'iperbole di fuga dalla Terra con `escape_hyp`, che quindi risulta diversa dalla precedente iperbole di fuga necessaria per la missione originale.

In Figura 6.2 è mostrato il plot della manovra.

Da notare che la velocità eliocentrica presa a riferimento per la fuga dalla SOI terrestre è di fatto calcolata come se il trasferimento interplanetario da fare fosse una manovra alla Lambert. Infatti, consideriamo il trasferimento alla Hohmann come caso particolare del trasferimento alla Lambert. Le motivazioni di questa scelta sono state descritte in dettaglio nel sottoparagrafo 2.2.1.

```
43 [dv1, ~] = escape_hyp(3, v0, 200, begin_date, 1);
```

6.3 Fase di cattura e parcheggio - Giove

La stessa procedura usata per la fase di fuga, è effettuata anche per la fase di cattura nella SOI di Giove, utilizzando stavolta la velocità eliocentrica della sonda in arrivo alla SOI gioviana (ovvero la velocità di fine trasferimento) e la funzione `capture_hyp`. I parametri dell'orbita di parcheggio rimangono invariati.

La manovra è plottata in Figura 6.3.

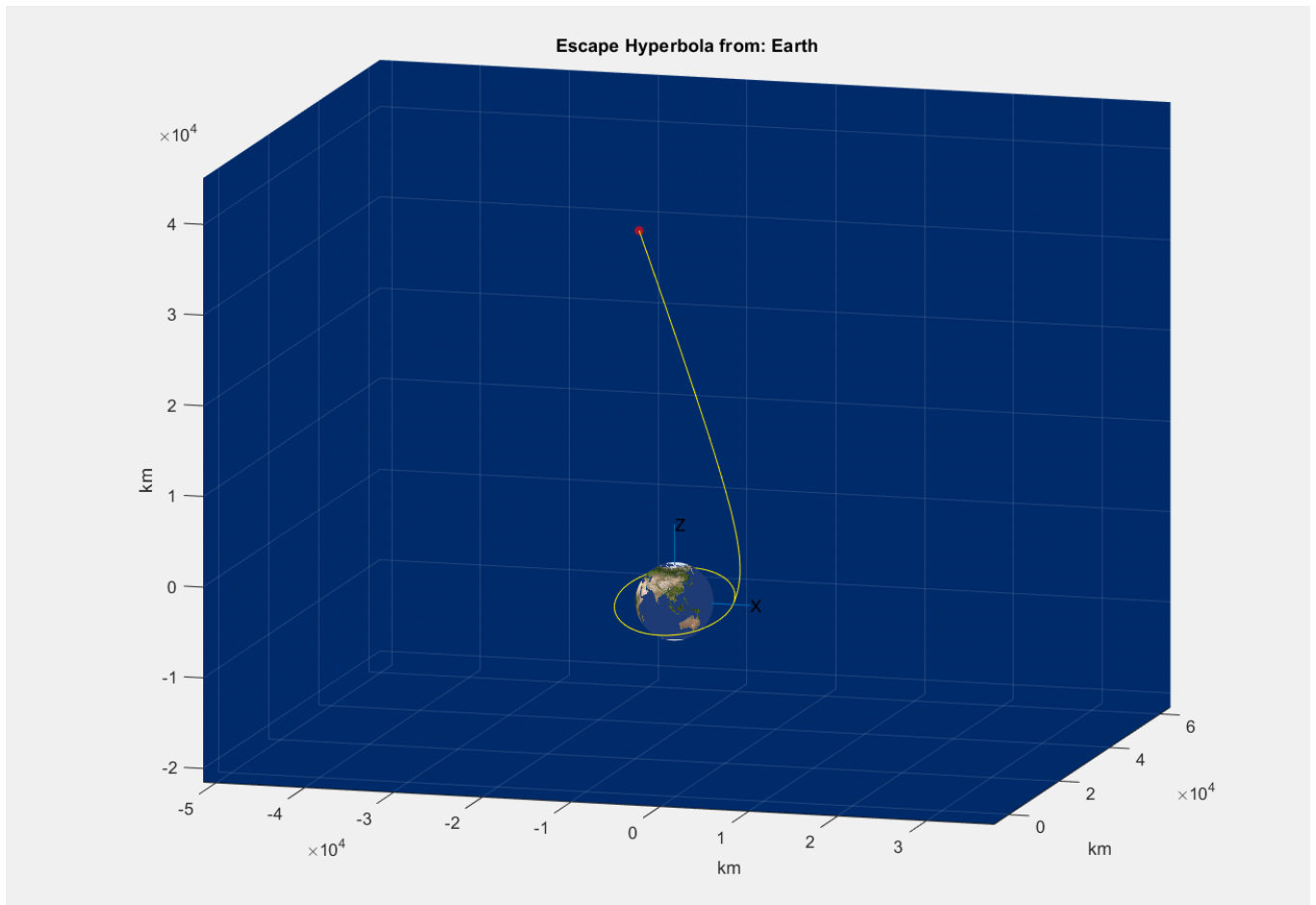


Figura 6.2: Iperbole di fuga dalla Terra - caso di trasferimento eliocentrico con Hohmann

```
44 [dv2, ~] = capture_hyp(5, vf, end_date, 1, 8e4, 0.6);
```

Si è scelto di mantenere la stessa orbita di parcheggio rispetto alla missione originale in modo da avere un termine di paragone più significativo. Per ognuna di queste fasi viene calcolato il dispendio energetico, che ci servirà per un successivo confronto.

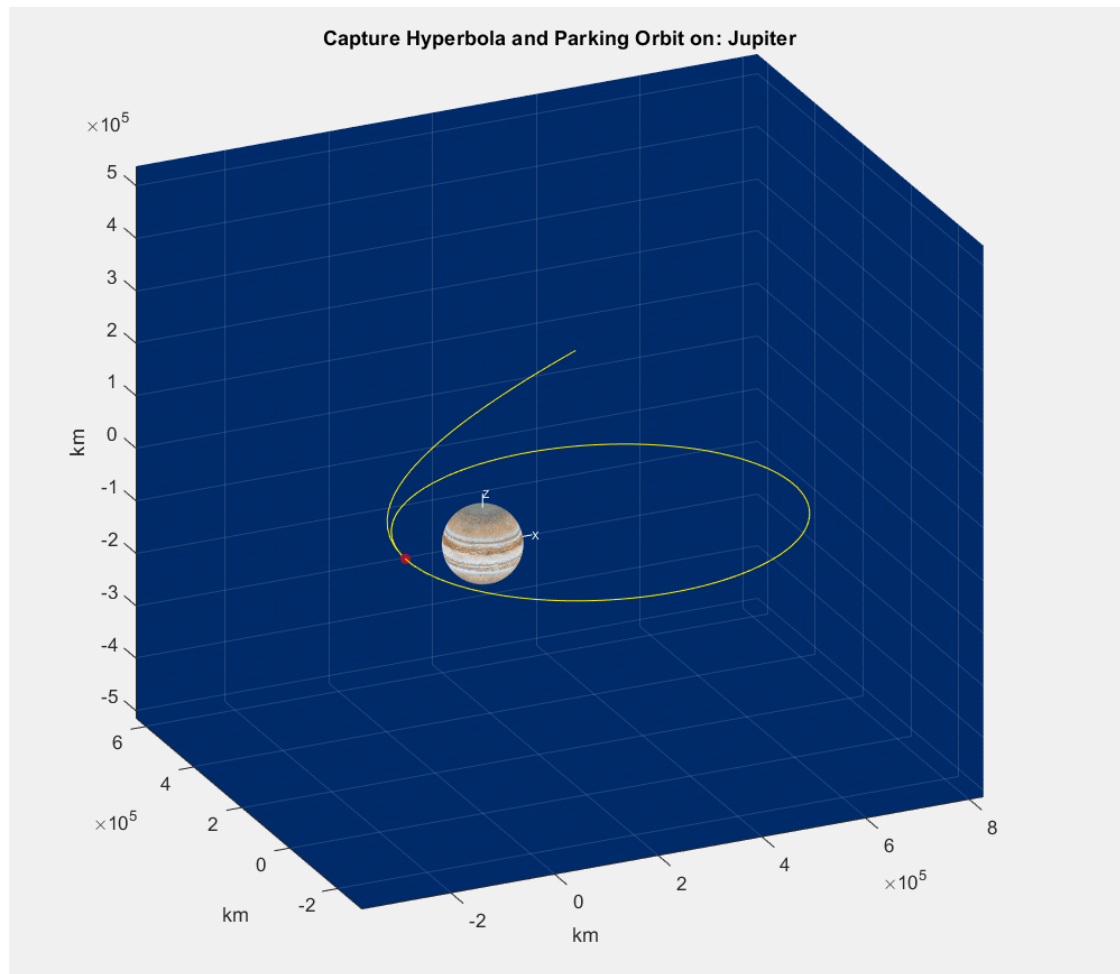


Figura 6.3: Ingresso e parcheggio su Giove - caso di trasferimento eliocentrico con Hohmann

Capitolo 7

Conclusioni

Alla luce dei calcoli fatti, si può notare come la missione alternativa sia più svantaggiosa in termini di consumi energetici ma più vantaggiosa in termini di tempi di volo.

I risultati sono in linea con quello che ci aspettavamo. Ciò risiede nel fatto che, nonostante Hohmann sia il trasferimento a due impulsi ottimo, nella missione originale abbiamo potuto contare anche sull'effetto fionda, con la quale abbiamo potuto ottenere una accelerazione senza utilizzo di carburante.

Le tempistiche invece sono notevolmente migliorate. In primis, vengono eliminati dalla traiettoria di missione i passaggi sui pianeti Marte e Terra, rendendo il percorso della sonda più breve; inoltre è stato considerato il tempo di volo effettivo, ritenendo il tempo di attesa Δt_a non influente.

Per quanto riguarda il Δv , il cui confronto con la missione originale può essere fatto tramite le figure 7.1 e 7.2, ad un'osservazione preliminare potrebbe sembrare che il vantaggio ottenuto con l'utilizzo delle manovre di flyby non sia tale da giustificare un tempo di volo praticamente raddoppiato. E' per questo fondamentale tenere presente che, nel caso di manovra di Hohmann, si è ricorsi all'approssimazione di manovra planare considerando la proiezione della posizione di Giove sul piano dell'eclittica. Utilizzando la funzione `body_elements_and_sv`, ciò ha significato annullare la sua componente z .

Per quanto riguarda la manovra originale, invece, si sono considerate le posizioni (e velocità) reali: di conseguenza, le orbite risultanti sono tridimensionali e questo potrebbe aver portato ad una diminuzione della differenza tra i due Δv necessari per la manovra che potrebbe risultare fuorviante.

Infine, la manovra alla Hohmann è stata sicuramente più facile da implementare in quanto l'unico vincolo a cui stare attenti è stato essenzialmente il rifasamento necessario alla partenza. Al contrario, una progettazione della missione con flyby, richiede uno studio preciso delle date e quindi delle posizioni relative fra i pianeti, oltre che a far coincidere velocità relativa della sonda in ingresso e in uscita al pianeta.

Quest'ultimo punto è solo in parte una complicità, in quanto, se così non fosse stato, avremmo dovuto fornire un Δv in uscita al flyby dal pianeta per uguagliare i moduli dei due vettori velocità, anche se ciò sarebbe significato un'ulteriore spesa energetica andando a limitare i benefici dell'eseguire un flyby.

Altra considerazione da fare è che abbiamo optato di ipotizzare trascurabile l'inclinazione di Europa rispetto all'equatore gioviano. Questa semplificazione ci è parsa ragionevole in quanto l'inclinazione ha un valore di $0,47^\circ$.

In ogni caso, abbiamo comunque valutato l'eventuale spesa energetica aggiuntiva per un cambio

di piano equatoriale durante la fase del trasferimento nel sistema di Giove. Come si nota infatti nella Figura 7.3, tale consumo risulta, in relazione agli altri Δv , irrisorio.

```

Command Window

/*-----*/
Departure Date from Earth: 16-Oct-2024

Delta_V required for escape hyperbola is: 4.8901 [km/s]

/*-----*/
First flyby => leading side:
Heliocentric velocity pre Flyby is: 24.2274 [km/s]
Heliocentric velocity post Flyby is: 22.7207 [km/s]

/*-----*/
Second flyby => trailing side:
Heliocentric velocity pre Flyby is: 35.2977 [km/s]
Heliocentric velocity post Flyby is: 39.0657 [km/s]

/*-----*/
Arrival Date on Jupiter: 11-Apr-2030

Delta_V required for parking orbit is: 4.74793 [km/s]

/*-----*/
Total Delta_V for Earth-Jupiter transfer is: 9.63803 [km/s]

/*-----*/
Time of Flight for Earth-Jupiter transfer is: 2003 [days]

/*-----*/
Jupiter parking orbit departure: 12-Apr-2030 12:10:00

Delta_V required for leaving Jupiter parking orbit is: 0.425358 [km/s]

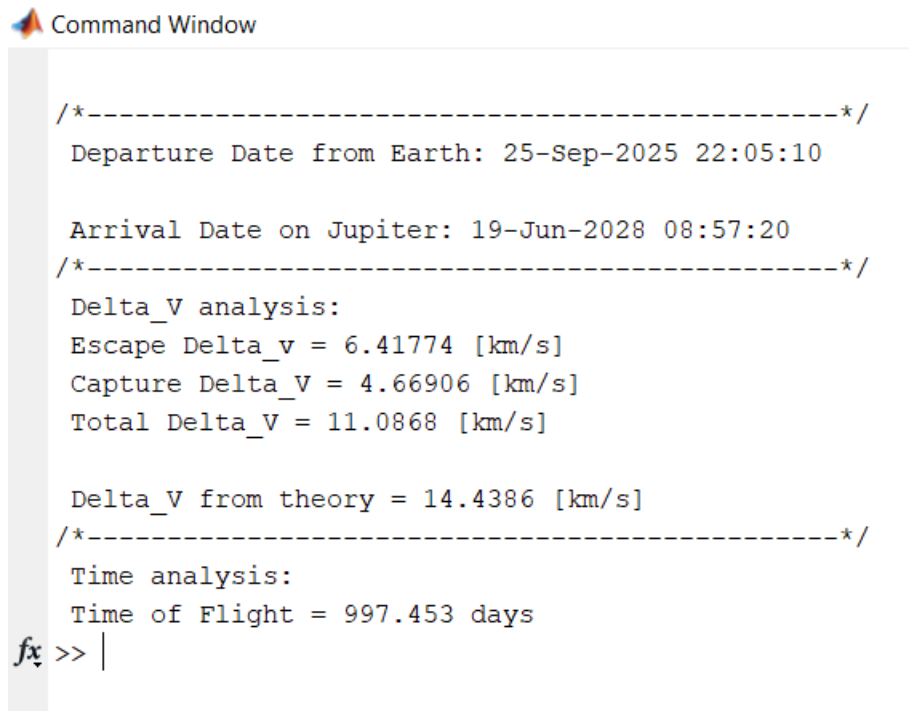
/*-----*/
Arrival Date on Europa parking orbit: 13-Apr-2030 07:22:00

Delta_V required for Europa parking orbit is: 4.43689 [km/s]

/*-----*/
Total mission Delta_v: 14.5003 [km/s]
/*-----*/

```

Figura 7.1: Dati di fine missione con pianificazione originale



```

Command Window

/*-----*/
Departure Date from Earth: 25-Sep-2025 22:05:10

Arrival Date on Jupiter: 19-Jun-2028 08:57:20
/*-----*/
Delta_V analysis:
Escape Delta_v = 6.41774 [km/s]
Capture Delta_V = 4.66906 [km/s]
Total Delta_V = 11.0868 [km/s]

Delta_V from theory = 14.4386 [km/s]
/*-----*/
Time analysis:
Time of Flight = 997.453 days
fx >> |

```

Figura 7.2: Dati di fine missione con pianificazione alternativa con Hohmann

```

>> 2*v*sin( deg2rad(0.47) / 2 )

ans =

    0.0114

```

Figura 7.3: Δv per l'eventuale cambio di piano su Europa: orbita finale di parcheggio equatoriale

Bibliografia

- [1] Howard D.Curtis. *Orbital Mechanics for Engineering Students, 3rd Edition*. Butterworth-Heinemann, Elsevier 2014.
- [2] Giovanni Mengali, Alessandro A.Quarta. *Fondamenti di Meccanica del Volo Spaziale*. Pisa, Pisa University Press, 2013.
- [3] NASA Mission Europa Clipper
<https://spaceflightnow.com/2021/02/11/nasa-decides-to-launch-europa-clipper-on-commen>
- [4] Appunti e slides del corso *Robotica Aerospaziale* A.A 2020/21, Università di Pisa.