Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Aerospaziale
A.A. 2021/2022
Sessione di Laurea Aprile 2022

Master Thesis

From Near-Rectilinear Halo Orbit to Low Lunar Orbit: a Propulsive Delta V Optimization Approach

Relatori:
Prof. Nicole Viola
Dr. Jasmine Rimani

Candidata:
Alessandra Rispoli s243903

Ringraziamenti

I miei più sentiti ringraziamenti vanno alla Professoressa Nicole Viola e alla mia correlatrice Jasmine Rimani, per avermi inserito in questo appassionante progetto, per il loro supporto, la loro attenta guida, la loro continua disponibilità. Grazie per avermi dato l'opportunità e i mezzi per poter crescere e concludere in modo più che soddisfacente questo percorso.

Semplicemente grazie alla mia famiglia, mia madre Rachele, mio padre Giuseppe e mia sorella Roberta. Sicuramente poche righe non saranno abbastanza per racchiudere l'immenso amore, sostegno, comprensione, pazienza che mi avete dimostrato per tutta la vita, non solo in questo percorso. Ci siete sempre stati e ci sarete sempre, siete la mia ancora e io sarò la vostra. Questo traguardo è vostro quanto mio perché non l'avrei mai raggiunto senza di voi, anzi, probabilmente sarei in un monastero thailandese a trovare me stessa dopo aver mollato gli studi! Un piccolo grazie va anche al membro più piccolino della famiglia, la nostra cagnolina Kira, per avermi fatto tanto ridere, avermi riempito d'affetto e avermi distrutto i calzini. Un grosso abbraccio e un grazie vanno anche ai nonni, per tutti i baci, l'affetto, i sorrisi e il cibo.

Nico, che dire, mi hai vista studiare fino alle 3 del mattino, mi hai visto gioire per un esame andato bene, ma anche entrare in crisi e sentirmi soffocare per l'ansia. Mi hai supportata, incoraggiata e spinta a credere in me. Ci sei stata in questo percorso quanto negli altri aspetti della mia vita, migliorandola sempre. Ti amo e sono felice di poter vivere con te questo traguardo e i prossimi che verranno.

Dedico un ringraziamento anche ai miei amici, perché ho condiviso con voi ogni fase e avvenimento di questo percorso universitario e avete sempre avuto la cura e l'attenzione giusta quando avevo bisogno di un confidente, la leggerezza giusta quando avevo voglia di divertirmi e la carica giusta quanto avevo bisogno di una spinta.

In ultimo, ma non per importanza, ringrazio me stessa, per l'impegno, il tempo, l'energia, i sacrifici, la forza nel tirarmi fuori dai momenti di sconforto e nel ritrovare la via quando mi sentivo un po' persa, la passione.

Abstract (English Version)

Recent trends in the international space sector see the Moon as a center of interest for the main Space Agencies, as well as for some private companies. Numerous studies, discoveries and technological advancements have made it possible for the permanent establishment of humans on the lunar surface to become one of the main objectives of space exploration, through the construction of special infrastructures and supports. The colonization of the surface of our satellite will in fact have important implications, both as a source of resources, and as an intermediate for the exploration of other celestial bodies, including Mars. Of particular interest is the lunar south pole, thanks above all to the considerable amount of ice water deposits discovered in this area.

This thesis is part of the iDREAM project, carried out by Politecnico di Torino and funded by the European Space Agency (ESA). The project aims to develop a rapid design and mission analysis tool for a Human Landing system. The tool will have to carry out the conceptual design of the main vehicle subsystems and the preliminary definition of the trajectories. The lander will include an Ascent & Descent module that is expected to operate between the lunar surface and the orbit of the Lunar Orbital Platform Gateway (LOP-G). The software used for mission analysis is Astos. The mission scenario involves the orbital transfer from the Near Rectilinear Halo Orbit (NRHO), in which the lander is in a docket configuration with the LOP-G, to the Low Lunar Orbit (LLO). From the LLO a Descent phase is planned towards the lunar surface. After a certain period on the lunar surface the Ascent phase is expected towards the LLO and, subsequently, the orbital transfer back to NRHO. But a problem arises: Halo Orbits are still under development in Astos. Now it is possible to use Halo Orbits as initial conditions for a scenario, but they would be too unstable, diverging very easily. It is in this context that this thesis is placed, aimed at modeling and optimizing the transfer trajectory between the NRHO and the LLO.

The first part of the thesis includes a literature review, aimed at understanding the dynamic behavior of the system, modeled as a Circular Restricted Three-Body Problem (CR3BP), and at understanding the optimization techniques applied to the trajectory. Then the problem and the corresponding mathematical model are defined and implemented using the Python programming language. The developed code provides for the optimization of the trajectory in terms of ΔV, through appropriate Initial Guess, the imposition of defined Equality and Inequality Constraints and appropriate Bounds within which to let the variables vary. Subsequently, the recalculation and plot of the optimized trajectory is envisaged, as well as the calculation of the quantities necessary for the simulation of the subsequent mission phases, including the consumption of propellant. These quantities, after having been verified by comparing them with the typical values found in the literature, will modify the file containing the data necessary to simulate the Descent trajectory through Astos. The simulation of the trajectory is then carried out in Astos with the data found through the Python code.

Abstract (Italian Version)

Le tendenze recenti del settore spaziale internazionale vedono la Luna al centro degli interessi delle principali Agenzie, nonché di alcune compagnie private. Diversi studi, scoperte e avanzamenti tecnologici hanno fatto sì che uno degli obiettivi principali dell'esplorazione spaziale diventasse proprio lo stabilimento permanente della presenza dell'uomo sulla superficie lunare, attraverso la costruzione di apposite infrastrutture e supporti. La colonizzazione della superficie del nostro satellite avrà infatti risvolti importanti, sia come punto di approvvigionamento di risorse, sia come intermezzo per l'esplorazione di altri corpi celesti, tra cui Marte. Di particolare interesse è il polo sud lunare, grazie soprattutto alla considerevole quantità di depositi di acqua ghiacciata scoperti in questa zona.

Di interesse per questa tesi è il progetto iDREAM, portato avanti dal Politecnico di Torino e finanziato dall'Agenzia Spaziale Europea (ESA). Il progetto si pone l'obiettivo di sviluppare un tool di progettazione rapida e mission analysis per un sistema Human Landing. Il tool dovrà prevedere la progettazione preliminare dei principali sottosistemi del velivolo e la definizione preliminare delle traiettorie. Il lander comprenderà un modulo di Ascent & Descent che ci si aspetta operi tra la superficie lunare e l'orbita della Lunar Orbital Platform Gateway (LOP-G). Il software utilizzato per l'analisi di missione è Astos. Lo scenario di missione prevede il trasferimento orbitale dalla Near Rectilinear Halo Orbit (NRHO), in cui il lander è in una configurazione docket con la LOP-G, verso la Low Lunar Orbit (LLO). Dalla LLO sussegue una fase di Descent verso la superficie lunare. Dopo un determinato periodo di tempo sulla superficie lunare segue la fase di Ascent verso la LLO e, successivamente, il trasferimento orbitale ancora in NRHO. Sorge però una problematica: le Halo Orbits sono ancora in fase di sviluppo in Astos. Al momento è possibile utilizzare le Halo Orbits come condizioni iniziali per uno scenario, ma risulterebbero troppo instabili, andando a divergere molto facilmente.

È in questo contesto che si colloca la tesi presentata, volta alla modellizzazione e ottimizzazione della traiettoria di trasferimento tra l'orbita NRHO e l'orbita LLO.

La prima parte della tesi prevede, dunque, un lavoro di literature review volto a comprendere il comportamento dinamico del sistema, modellizzato come un Circular Restricted Three-Body Problem (CR3BP), e a comprendere le tecniche di ottimizzazione applicate alla traiettoria. Segue una parte di definizione del problema e del corrispettivo modello matematico, accompagnata dall'implementazione dello stesso, utilizzando il linguaggio di programmazione Python. Il codice sviluppato prevede l'ottimizzazione della traiettoria in termini di ΔV, tramite opportune Initial Guess, l'imposizione di definiti

Vincoli di Equality e Inequality e di opportuni Bounds entro cui far variare le variabili. Si procede con il ricalcolo e il plot della traiettoria ottimizzata, oltre che con la determinazione delle grandezze necessarie alla definizione delle successive fasi di missione, tra cui il consumo di propellente. Tali quantità, dopo essere state verificate confrontandole con i valori tipici riscontrati in letteratura, andranno a modificare il file contenente i dati necessari alla simulazione della traiettoria di Descent tramite Astos. La simulazione della suddetta traiettoria viene quindi effettuata in Astos, con i dati trovati tramite il codice Python.

Indice

1. INTRODUZIONE

La Luna è diventata uno dei fulcri dell'esplorazione spaziale odierna. Si punta allo stabilimento di una presenza fissa e sostenibile dell'uomo sulla Luna, in prospettiva di espandersi poi dalla Luna verso Marte. Grazie all'esplorazione lunare sarà possibile avanzare infatti anche verso l'esplorazione marziana. Potremo dimostrare il successo delle nuove tecnologie, strategie e dei nuovi protocolli volti allo stabilimento nello spazio di una base abitabile, al trasporto, all'autonomia operativa, alla costruzione e alla manutenzione di nuove infrastrutture e veicoli, alla generazione di potenza. Decine di Agenzie Spaziali hanno già sottoscritto la nuova Global Exploration Roadmap riassunta in figura:

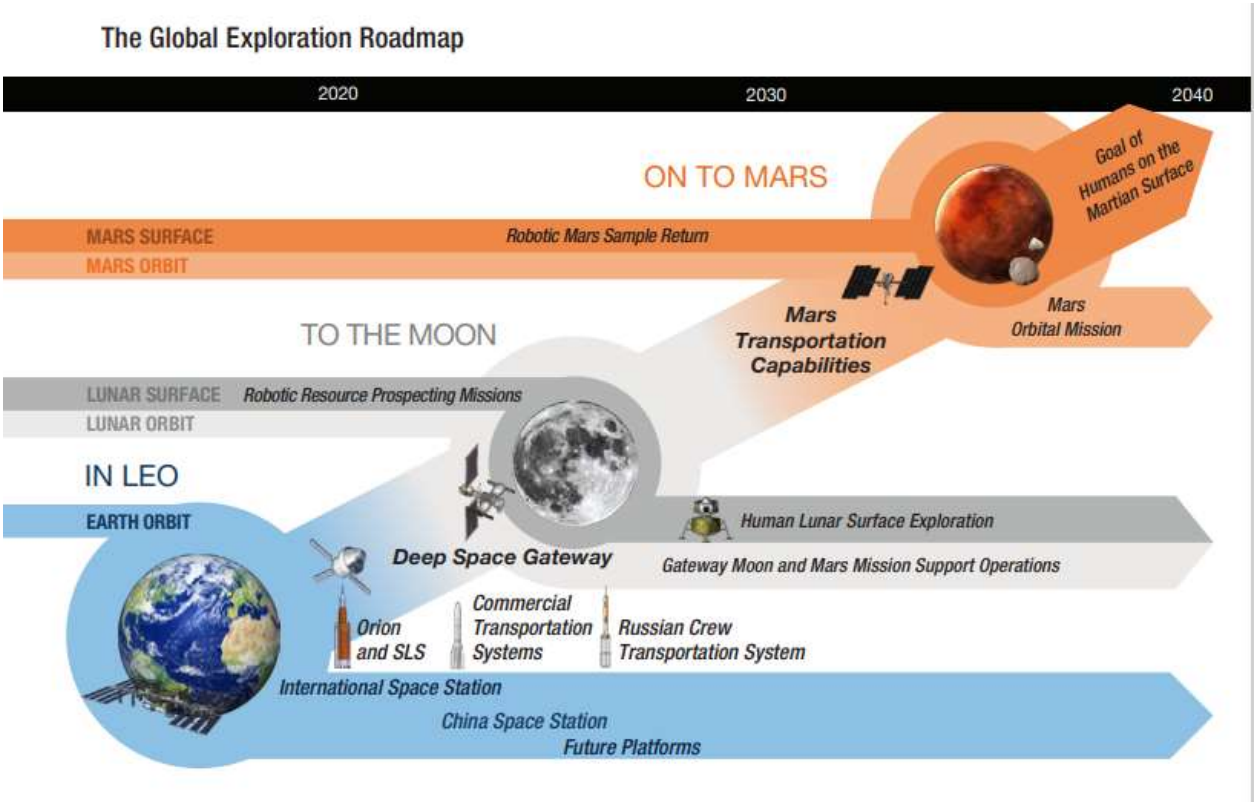

Figura 1 - Global Exploration Roadmap [credits: [11]]

Tante sono dunque le missioni, manned e unmanned, pianificate nel prossimo decennio verso il nostro satellite. Una recente scoperta pone basi ancora più solide alla permanenza dell'uomo su suolo lunare: la scoperta di acqua ghiacciata al polo sud lunare. Tali riserve di ghiaccio hanno il vantaggio di poter essere utilizzate per la produzione di acqua da bere e ossigeno per gli astronauti oltre che per produrre carburante a base di idrogeno.

Un ulteriore supporto operazionale sarà rappresentato dalla Lunar Gateway, una stazione spaziale orbitante in ambiente lunare, che permetterà ai velivoli di transito di ricevere la giusta manutenzione e il rifornimento di carburante, di testare nuove tecnologie e sarà inoltre accessibile a tutte le Agenzie o compagnie private che necessitano di un porto intermedio per raggiungere la Luna o Marte.

Prima di collocare in questo scenario il progetto protagonista di questa tesi facciamo un excursus più dettagliato su quello che è stata e che sarà l'esplorazione lunare.

[credits: [12],[13],[15],[16]]

1.1 PANORAMICA STORICA

La corsa verso la Luna si apre durante la Guerra Fredda tra Unione Sovietica e Stati Uniti. Il primo successo fu dell'Unione Sovietica nel 1959, anno in cui riuscì a far atterrare per la prima volta uno spacecraft sulla superficie lunare. Si trattava di Luna 2, che atterrò nella zona del Mare Imbrium, uno dei più grandi crateri del Sistema Solare. Un ulteriore conquista dell'Unione Sovietica furono, nel 1966, le prime immagini mai scattate del suolo lunare. A catturarle fu il Luna 9, atterrato nell'Oceanus Procellarum, il più vasto mare lunare. Il Luna 9 fu anche il primo spacecraft ad atterrare in sicurezza sul suolo lunare. Poco tempo dopo il Luna 10 divenne il primo velivolo ad orbitare intorno al satellite. Negli stessi anni però iniziano a fare progressi anche gli Stati Uniti. Il programma Ranger, attivo dal 1961 al 1965, fu il primo a permettere la cattura di immagini ad alta quota della Luna. In particolare, dopo 6 missioni fallimentari, le missioni Ranger 7, Ranger 8 e Ranger 9 permisero di comprendere meglio il paesaggio lunare e di compiere avanzamenti con la tecnologia necessaria per le missioni future. Un altro importante passo per la NASA fu il programma Surveyor, che ebbe luogo dal 1966 al 1968. Il programma, che si compose di 7 landers equipaggiati di fotocamere e tecnologie di analisi del suolo lunare, riuscì a dimostrare la fattibilità dell'atterraggio dell'uomo sulla superficie del satellite. Fu la volta quindi del programma Lunar Orbiter, ideato per permettere la mappatura del satellite e di conseguenza selezionare il luogo di atterraggio per le missioni che porteranno l'uomo sulla Luna.

È il 20 luglio 1969, missione Apollo 11. Gli astronauti Neil Armstrong e Edwin "Buzz" Aldrin atterrano nel Mare Tranquillitatis all'interno del modulo lunare Eagle (LM-5), mentre l'astronauta Michael Collins resta in orbita intorno alla Luna nel modulo di comando Columbia (CM-107). Il primo a toccare il suolo lunare fu

Neil Armstrong, pronunciando la famosa frase: “That’s one small step for a man, one giant leap for mankind”.

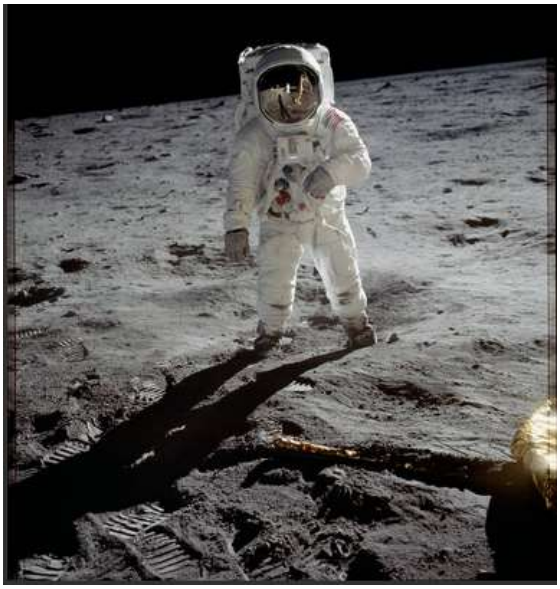

Figura 2 - Buzz Aldrin on the Moon [credits: [27]]

Seguirono diverse missioni sulla superficie lunare. L’Apollo 12 riuscì a migliorare l’atterraggio. L’Apollo 13 invece dovette abortire la missione a causa di un’esplosione dei serbatoi di ossigeno. La missione è considerata comunque un successo in quanto si riuscì a riportare sani e salvi gli astronauti sulla Terra. Il comandante dell’Apollo 14, Alan Shepard superò il record per la distanza più lunga percorsa sul suolo lunare percorrendo 3 Km. L’Apollo 15 trascorse circa 3 giorni sul satellite, portando a termine diversi passi in avanti. Si riuscirono infatti a collezionare alcuni Kg di campioni e a percorrere più di 27 Km all’interno del rover lunare soprannominato Moon Buggy, un velivolo elettrico pensato per espandere il range di esplorazione degli astronauti.

L’Apollo 16 e l’Apollo 17 furono le ultime missioni umane sulla Luna, nel 1972. L’unmanned lander Luna 24 fu l’ultimo ad atterrare su suolo lunare nello scorso secolo, nel 1976. Da tali missioni ci portiamo dietro un’enorme quantità di nuovi dati e conoscenze acquisite sulla storia e le caratteristiche del nostro satellite. In seguito a questi ultimi atterraggi l’interesse verso la Luna si spense per alcuni anni.

Figura 3 - Lunar Roving Vehicle, also called Moon Buggy [credits: [23]]

Nel 1994 la Luna tornò ad essere obiettivo dei programmi spaziali, grazie ad una missione congiunta fra la NASA e la Strategic Defense Initiative Organization. Lo spacecraft Clementine mappò la superficie attraverso delle lunghezze d'onda dall'ultravioletto all'infrarosso, allo scopo di collezionare dati sulle proprietà gravitazionali e sulla mineralogia del suolo. Grazie ad alcuni risultati della sonda Clementine fu possibile suggerire la presenza di acqua ghiacciata ai poli. Nel 1999 la missione Lunar Prospector confermò la possibilità di acqua ghiacciata ai poli, ma lo spacecraft si schiantò sulla superficie lunare prima di trovare prove a riguardo. Nel 2009 il satellite NASA LCROSS (Lunar Crater Observation and Sensing Satellite) riuscì a trovare prove della presenza di acqua nelle vicinanze del polo sud lunare. Sempre dal 2009 il Lunar Reconnaissance Orbiter (LRO) sta raccogliendo mappe 3D ad alta risoluzione della superficie lunare, con una copertura del 98,2%. Lo scopo è quello di determinare possibili siti d'atterraggio, identificare risorse sulla superficie, valutare i livelli di radiazione e testare le nuove tecnologie. Tra il 2011 e il 2012 la missione GRAIL (Gravity Recovery and Interior Laboratory), parte del programma Discovery della NASA, fu lanciata. Essa era supportata da due satelliti spacecraft gemelli: GRAIL A e GRAIL B. I due veicoli riuscirono a mappare la struttura del campo gravitazionale lunare, per poi schiantarsi intenzionalmente nell'area del polo nord lunare.

[credits: [14],[17],[18],[19],[20],[21],[24],[25],[26]]

1.2 MISSIONI RECENTI E PROSPETTIVE FUTURE

Oggigiorno la Luna è entrata a far parte dei principali interessi di molte agenzie pubbliche, quelle già citate e protagoniste della storia dell'esplorazione lunare, ma anche quelle che fin'ora non avevano preso parte alla corsa alla Luna, come l'Agenzia Spaziale Giapponese (JAXA), l'Agenzia Spaziale Europea (ESA) o l'Agenzia Spaziale Canadese (CSA). Inoltre, con l'ascesa del settore privato dei viaggi spaziali, anche alcune compagnie hanno espresso il loro interesse verso il nostro satellite naturale, come la compagnia americana Space X.

L'Agenzia Spaziale Europea è il motore del settore spaziale in Europa. Attualmente ha 4 progetti attivi sul versante lunare. Il primo è chiamato Luna 27 o Lunar Resource Lander, in collaborazione con l'Agenzia Spaziale Russa Roscosmos. Il lander avrà l'obiettivo di esplorare e raccogliere informazioni sulle regioni del polo sud lunare. L'interesse verso questa area è dato soprattutto dal fatto che vi sono le condizioni favorevoli alla conservazione di importanti quantità di acqua ghiacciata, utile per ricavarne acqua potabile e carburante, e di altre sostanze volatili. Il compito dell'ESA sarà quello di provvedere al sistema GNC di precisione, che permetterà l'atterraggio nel punto designato. Il sistema è stato chiamato PILOT (Precise Intelligent Landing using On-board Technology). Una volta effettuato l'atterraggio, il sistema PROSPECT inizierà ad operare. Il sistema si compone di un trapano a percussione che raccoglierà campioni di regolite e ghiaccio per portarli all'interno del laboratorio dello spacecraft. All'interno di esso troviamo una strumentazione composta da quindici dispositivi atti non solo ad analizzare i campioni lunari, ma anche il plasma nell'esosfera, la polvere e l'attività sismica. Il secondo progetto che coinvolgerà la Luna è Orion, frutto di una collaborazione fra l'ESA e la NASA. Orion è un velivolo spaziale pensato per il trasporto di un equipaggio per diverse tipologie di missione, come Artemis 1 o per un collegamento con la futura stazione spaziale lunare LOP-G. Il veicolo è parzialmente riutilizzabile ed è composto da un Crew Module progettato da Lockheed Martin e un Service Mudule progettato da ESA e da Airbus Defence & Space. Quest'ultimo avrà lo scopo di fornire potenza, spinta propulsiva, acqua, ossigeno e altre necessità agli astronauti. Il terzo progetto è ISRU (In-Site Resource Utilization). Ha l'obiettivo di raccogliere e riprocessare materiale estratto direttamente dalla superficie lunare (o potenzialmente da un altro corpo celeste) per convertirlo in prodotti o servizi. Tale strumento potrebbe fornire materiale per combustibili, costruzioni, supporto alla vita, energia, riducendo il carico di partenza dalla Terra. L'ultimo progetto analizzato è la missione Heracles, in collaborazione

con le agenzie JAXA e CSA. Consiste nell'invio sulla superficie del polo sud lunare di un lander. Un rover esplorerà quindi zone inesplorate e sensibili dal punto di vista scientifico e raccoglierà campioni che verranno portati al lander. Il lander successivamente effettuerà una traiettoria di ascesa verso la LOP-G. Dalla LOP-G i campioni potranno essere inviati sulla Terra. La missione rappresenta un terreno di prova per l'interazione uomo-robot. Sarà infatti interamente operata dagli astronauti a bordo della Lurar Gateway.

L'ESA non è l'unica delle principali agenzie, diverse dalla NASA e dalla Roscosmos, ad aver puntato i riflettori verso la Luna. Nel 2007 la JAXA lancia il suo primo orbiter lunare, SELENE. Nello stesso anno la Cina lancia la sua prima missione lunare, la Chang'e-1, che ha realizzato una mappatura 3D della superficie. Nel 2013, con la missione Chang'e-3 la Cina diventa la terza nazione ad aver effettuato un soft landing sulla superficie lunare. Nel 2019 il rover cinese Yutu-2 diventa il primo ad essere atterrato nella regione più lontana della Luna. Un ruolo fondamentale iniziano ad occuparlo anche le compagnie private. Fin dagli anni '60 la NASA stipula contratti con privati. Fra le principali compagnie spaziali collaboranti con la NASA abbiamo adesso Space X, Blue Origine e Astrobotic. Tali compagnie però non si limitano più alla collaborazione con le agenzie, ma hanno impostato una serie di ambiziosi obiettivi aziendali. La Blue Origine di Jeff Bezoz vorrebbe costruire una base lunare in cui le persone potranno vivere e lavorare. Sta pianificando inoltre il trasporto di turisti in orbita lunare, così come la Space X e la Virgin Galactic. La Space X di Elon Musk progetta un veicolo spaziale, lo Starship, in grado di trasportare esseri umani sulla Luna e su Marte. È stata inoltre la prima compagnia ad aver progettato rockets riutilizzabili e attualmente si occupa di trasportare rifornimenti alla Stazione Spaziale Internazionale. Nel 2020 lo spacecraft della Space X Crew Dragon trasportò in orbita, dagli Stati Uniti per la prima volta dopo quasi dieci anni, gli astronauti della NASA Doug Hurley e Bob Behnken.

D'altra parte anche la NASA pianifica il suo ritorno sulla Luna, con il programma Artemis. L'obiettivo è quello di portare sulla superficie lunare la prima donna e il prossimo uomo, utilizzando lo spacecraft Orion sopra citato e il sistema di lancio SLS (Space Launch System). Il prossimo step sarebbe quindi la costruzione in orbita lunare della Lunar Orbital Platform-Gateway, in collaborazione con le altre agenzie. La Lunar Gateway rappresenterà un punto intermedio fondamentale per lo stabilimento di una base umana permanente sulla Luna e per l'esplorazione di Marte.

È in questo contesto che nasce il progetto iDREAM e conseguentemente la tesi qui esposta.

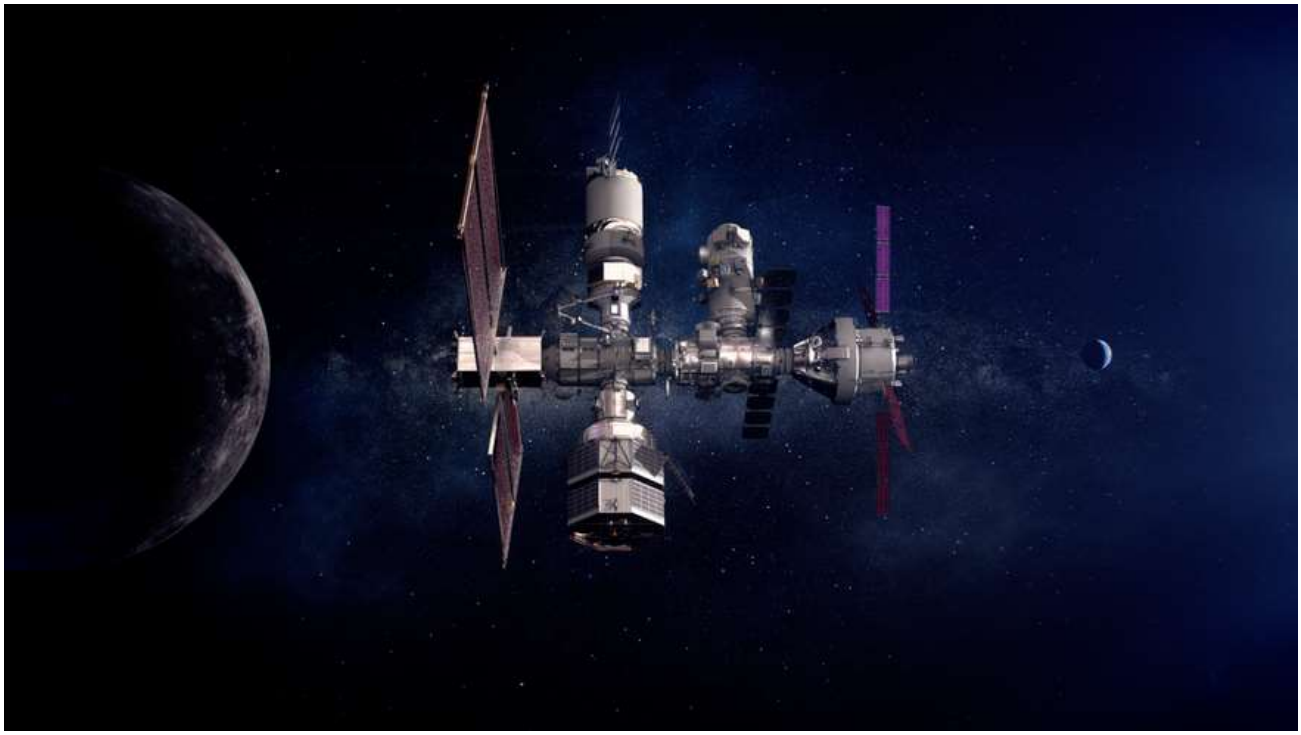

Figura 4 - Lunar Orbital Platform-Gateway (LOP-G) [credits: [29]]

[credits: [14],[16],[22]]

1.3 PROGETTO iDREAM

La seguente tesi si inserisce all'interno del progetto iDREAM. iDREAM è frutto di un contratto di studio fra l'Agenzia Spaziale Europea (ESA) e il Politecnico di Torino e ha come principale scopo la realizzazione di un tool che permetta di svolgere analisi preliminari di differenti concept di missione sulla superficie lunare. Le attività svolte dal Politecnico di Torino hanno dunque lo scopo di supportare il design rapido e l'analisi di missione di un sistema Human Landing (HLS) per la futura esplorazione lunare. Il sistema dovrà essere composto da un unico veicolo contenente un modulo di ascesa e discesa e dovrà operare tra la Lunar Orbital Platform-Gateway (LOP-G) e il polo sud lunare. Di seguito è riportata una tabella contenente i requisiti forniti dall'Agenzia Spaziale Europea che permetteranno di comprendere meglio le funzionalità da includere nel processo di progettazione:

	Requirements
	functional requirements
R.1	*The HLS shall be commanded and controlled using humans in the loop.*
R.2	*The HLS shall perform the tasks of docking with LOP-G and landing at the Moon having humans in the loop.*
R.3	*The HLS shall be able to recover from contingencies at any moment of the flight.*
R.4	*The HLS crew shall allow to monitor, command, control, and recover the propulsion subsystem ot the HLS flight vehicle*
R.5	*The HLS shall be able to withstand a travel time from the LOP-G of at least TBC hours.*
R.6	*The HLS mission shall be able to withstand a travel distance of at least TBC km.*
R.7	*The HLS shall be able to communicate with the LOP-G at any given time.*
R.8	*The HLS shall be able to used and re-used multiple times (up to 5 times)*
R.9	*The HLS shall be able to ascend and manoeuvre back to docking with the LOP-G.*
	Interface requirements
R.10	*The HLS shall be able to arrive to and depart from the LOP-G and land of the Moon using humans in the loop.*
	Enviromental requirements
R.11	*The HLS shall be able to sustain the environment (day/night) of the Moon South pole and the arrival to and departure from the LOP-G.*
	Operational requirements
R.12	*The HLS shall be able to accommodate* ***4 astronauts****.*
R.13	*The HLS shall allow to fly astronauts by means of a pre-programmed flight sequence.*
R.14	*The HLS shall allow stay on the surface of the Moon for up to 3 lunar nights.*
	Implementation requirements
R.15	*The HLS module shall include the following subsystems: Propulsion, avionics (GNC, OBC etc.), ECLSS, TCS, Comms, Structure, Power for all elements of HLS (ascender and descender).*
R.16	*The HLS module shall consist of the following inputs (with the option to insert only minimal set of inputs, and not necessarily all):* ***R16.1.1.*** *Vehicle Parts and configuration:* ***16.1.1.1.*** *Propulsion System: engine type, engine properties, Propellant tank, propellant type and properties* ***16.1.1.2.*** *Subsystems properties (mass, power etc.)* ***16.1.1.3.*** *Payload/Number of astronauts* ***16.1.1.4.*** *Configuration of the HLS and its different elements* ***R16.1.2.*** *Mission related:* ***16.1.2.1.*** *Desired landing site* ***16.1.2.2.*** *Mission phases* ***16.1.2.3.*** *Note that the initial point would be NRHO* ***R16.1.2.*** *Mission related:* ***16.1.2.1.*** *Desired landing site* ***16.1.2.2.*** *Mission phases* ***16.1.2.3.*** *Note that the initial point would be NRHO*

R.17	*The HLS module shall deliver the following outputs:* ***R17.1.*** *Mission and Vehicle Performance (propellant mass, ΔV, trajectories etc)* ***R17.2.*** *System budgets; mass, power, volume*

Tabella 1 - Requisiti del Sistema di Human Landing [credits: [1]]

I requisiti sopra elencati, rendono più chiaro ciò che ci si aspetta dallo spacecraft. Si vuole ad esempio che il velivolo stia sulla superficie lunare per tre notti lunari, che trasporti 4 astronauti, che possa essere riutilizzato 5 volte.

Anche se la progettazione dovrà attenersi a tali requisiti, lo scenario di missione dovrà mantenersi il più generale possibile in quanto dovrà permettere lo studio di diversi casi di missione.

Il processo inizia con una routine di definizione del veicolo e dei diversi sottosistemi che lo compongono, utilizzando come input iniziali il valore di ΔV e i dati riguardanti il payload. Dividiamo i sottosistemi in due categorie principali:

- Sottosistemi correlati al sistema propulsivo e alla massa di propellente
- Sottosistemi indipendenti dal sistema propulsivo

Una volta definiti i risultati della routine di progettazione del veicolo, in termini di budget volumetrico, di massa e di potenza, i parametri di interesse verranno condivisi, andando ad aggiornare il file *"homotopy.xml"* contenuto nella cartella *"bach"* nella directory dello scenario, per eseguire l'analisi di missione. L'analisi di missione verrà eseguita, utilizzando sia i dati forniti dalla routine precedente sia gli input assegnati dall'user. Una volta ottenuti anche i risultati dell'analisi di missione, il valore di ΔV ottenuto verrà confrontato con il ΔV iniziale e le attività riprenderanno dall'inizio per tutti i sottosistemi dipendenti dal sistema propulsivo e dalla massa di carburante. Il processo è iterativo e continua fin quando non si trova corrispondenza fra i valori calcolati durante la routine di progettazione dei sottosistemi e i risultati dell'analisi di missione, ottenendo quindi i risultati definitivi. Di seguito lo schema delle attività iterative di progettazione del tool completo:

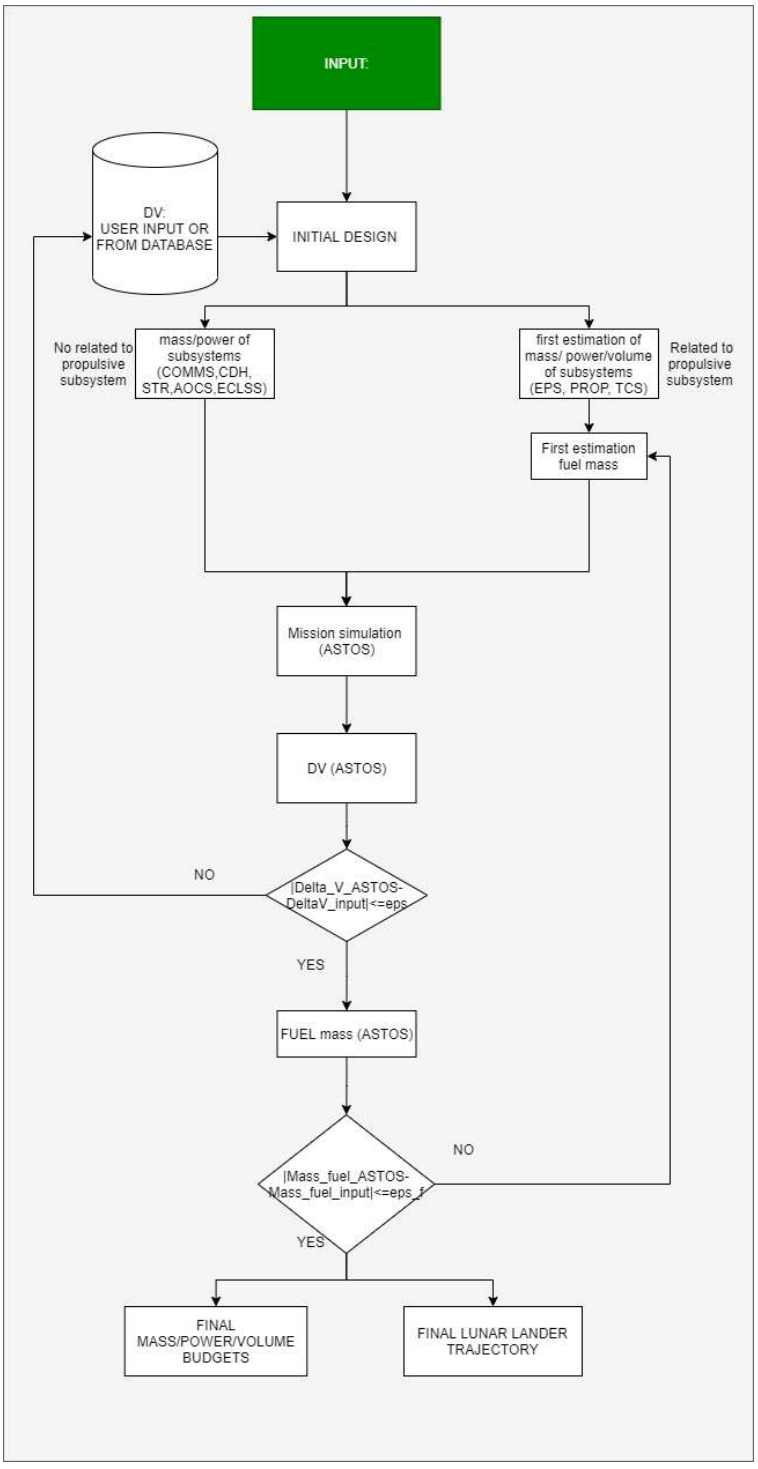

Figura 5 - Attività di Progettazione del Tool [credits: [1]]

E delle attività iterative di progettazione dei sottosistemi dipendenti dal sistema propulsivo:

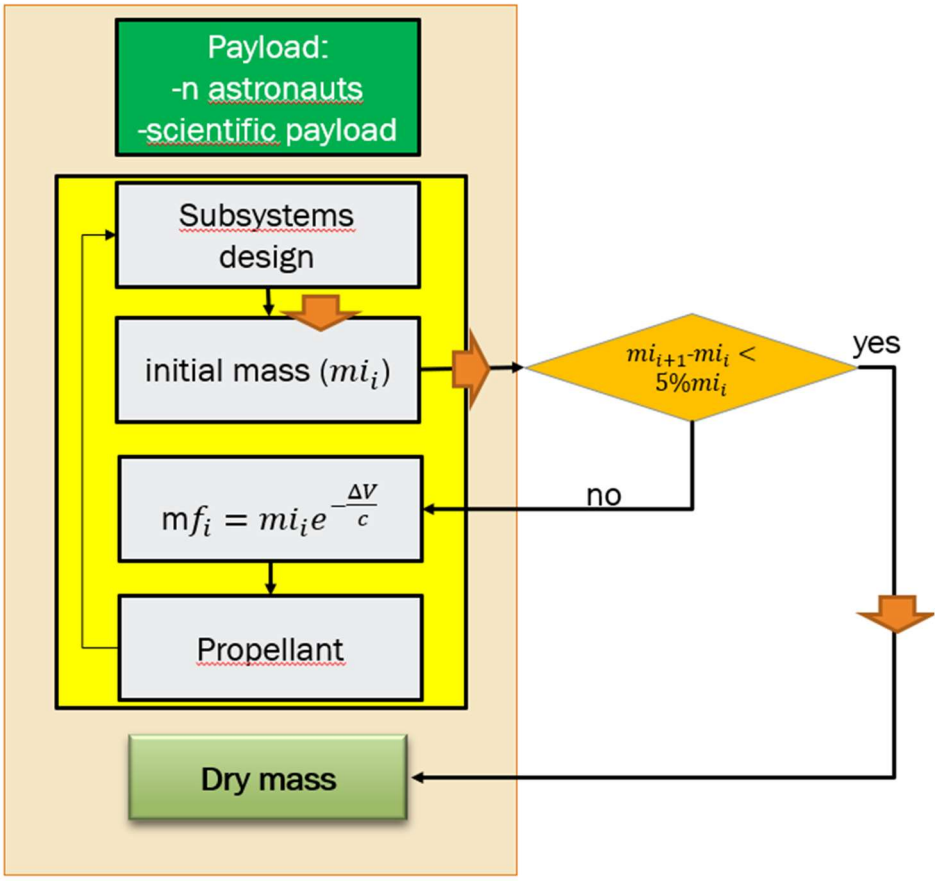

Figura 6 - Attività di Progettazione dei Sottosistemi dipendenti dal Sistema Propulsivo [credits: [1]]

Ad essere progettati utilizzando questo schema sono essenzialmente:

- Il sottosistema propulsivo
- L'AOCS
- Le strutture e altri meccanismi

Ad essere progettati senza tale schema iterativo, poiché non dipendenti dal sistema propulsivo, sono:

- L'ECLSS
- Il Communication Subsystem
- Il C&DH Subsystem
- L'EPS
- Il TCS

Una volta raccolti tutti i risultati si passa alla fase di analisi di missione.

Il profilo di missione considerato si compone di cinque fasi principali:

1. Orbita iniziale: NRHO con l'HLS in una configurazione docked con la LOP-G

2. Trasferimento orbitale da NRHO a LLO
3. Fase di Descent da LLO a superficie lunare
4. Dopo un determinato periodo temporale, fase di Ascent da superficie lunare a LLO
5. Trasferimento orbitale da LLO a NRHO (con l'HLS in configurazione docked con la LOP-G)

Tale profilo di missione dovrà essere implementato in Astos. Sorgono però due problematiche importanti:

- Le orbite halo sono ancora in via di sviluppo in Astos. In caso di utilizzo come condizioni iniziali porterebbero lo scenario a divergere rapidamente.
- Non è possibile automatizzare il processo di ottimizzazione per uno generico scenario in Astos, come invece si auspicherebbe

Per quanto riguarda il secondo punto si è deciso di disabilitare l'ottimizzazione e di lasciare la scelta all'user per molti inputs, in modo da permettere l'automatizzazione del processo. È possibile, infatti, modificare il valore di alcuni parametri andando a modificare il file *"homotopy.xml"* contenuto nella cartella *"bach"* della directory dello scenario. Astos andrà a leggere tali valori e li assocerà alle variabili al suo interno. Un esempio del file *"homotopy.xml"* è riportato di seguito:

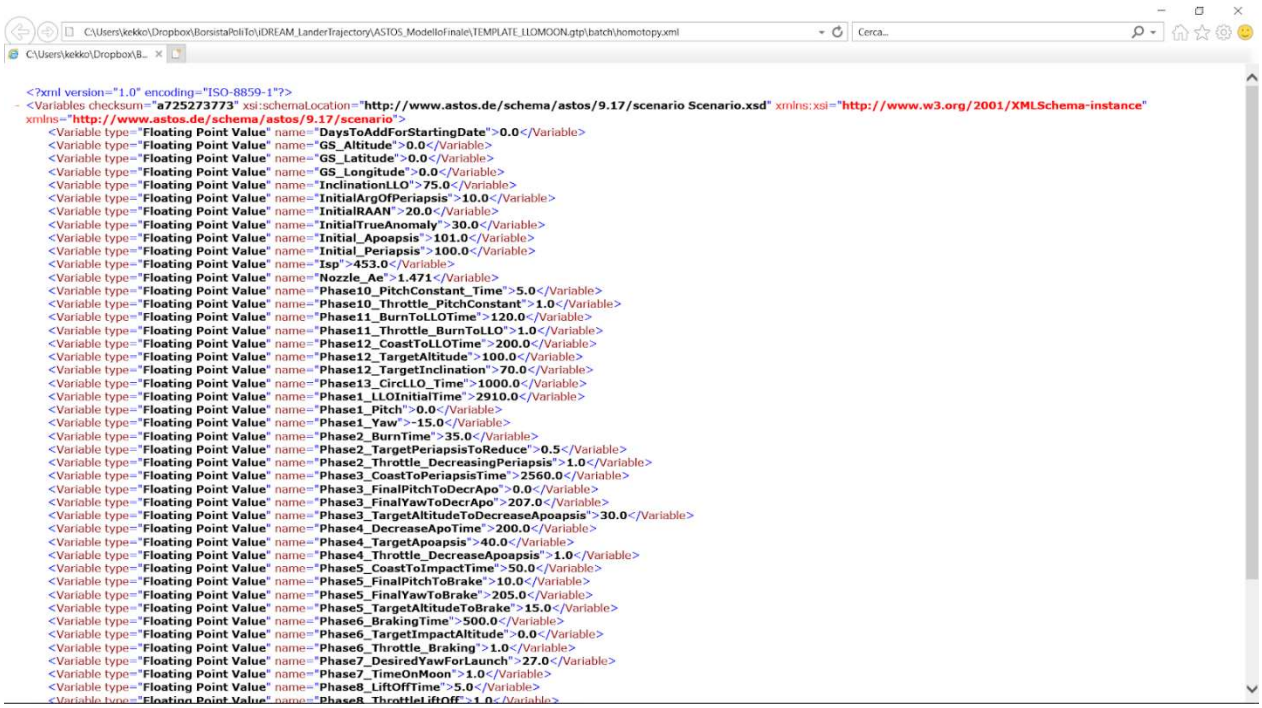

Figura 7 - Esempio File "homotopy.xml" [credits: [1]]

È importante che i parametri definiti dall'user abbiano un elevato grado di accuratezza in quanto non è effettuata alcuna ottimizzazione. Alcuni parametri del file vengono invece modificati con i risultati della routine di definizione dei sottosistemi e con la routine di definizione e ottimizzazione della traiettoria da orbita Halo a orbita lunare. Per ovviare al problema indicato nel primo punto, infatti, tale tratto di traiettoria viene ottimizzato utilizzando un codice Python. È proprio intorno a questo punto che si sviluppa la tesi in questione. I risultati di tale codice di ottimizzazione vengono utilizzati per modificare il file *"homotopy.xml"* e far partire dunque la simulazione delle successive fasi di missione tramite Astos. I risultati della simulazione completa verranno quindi salvati in due reports, uno per le variabili, uno per i grafici.

[credits: [1]]

2. MECCANICA ORBITALE

2.1 PARAMETRI ORBITALI

Per identificare in maniera univoca l'ubicazione di uno spacecraft si necessita di sei parametri orbitali di seguito elencati:

- Semiasse maggiore *a*: considerando un'orbita ellittica, rappresenta la metà del suo asse maggiore in lunghezza. Chiamiamo perigeo il punto dell'orbita meno distante dal corpo celeste attorno cui è definita e apogeo, al contrario, il punto maggiormente distante.

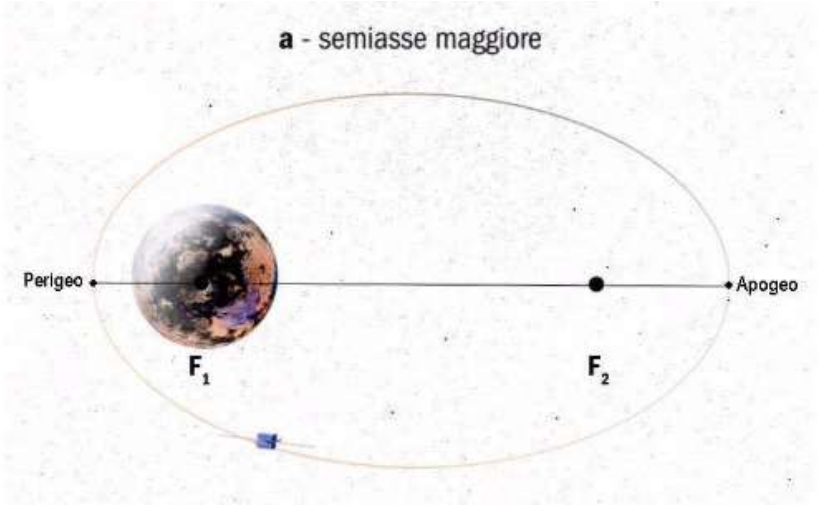

Figura 8 - Semiasse Maggiore [credits: [9]]

- Eccentricità *e*: un parametro costante che va a definire la forma dell'orbita.

e=0	*Cerchio*
0<e<1	*Ellissi*
e=1	*Parabola*
e>1	*Iperbole*

Tabella 2 - Possibili Valori dell'Eccentricità e [credits: [4]]

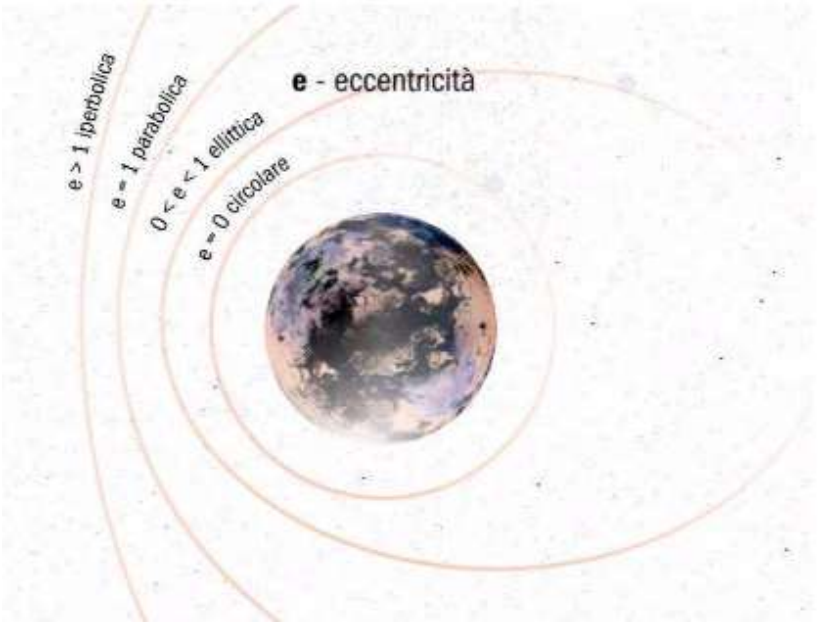

Figura 9 – Eccentricità [credits: [9]]

- Inclinazione *i*: indica l'angolo tra il piano orbitale e il piano equatoriale. È compreso tra *0°* e *180°*.

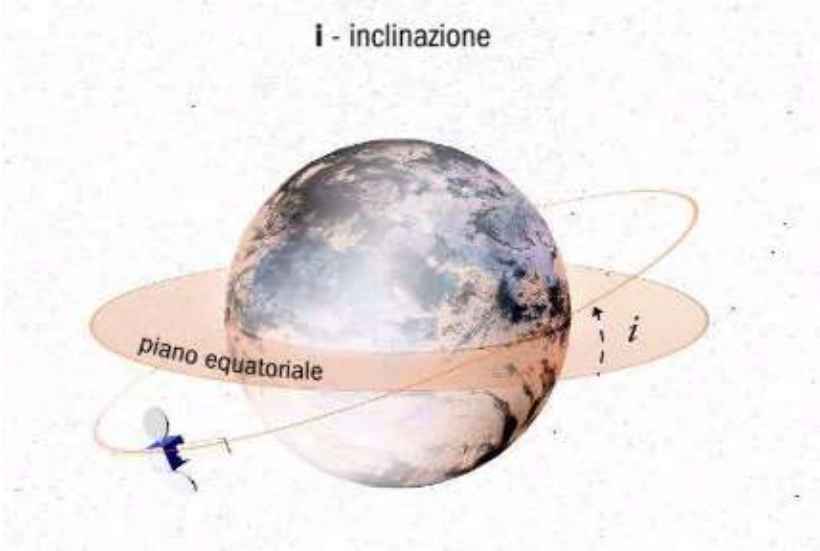

Figura 10 – Inclinazione [credits: [9]]

Chiamiamo nodi i punti in cui il piano orbitale e in piano equatoriale si intersecano. Nodo ascendente sarà quello in corrispondenza del quale il velivolo passa dall'emisfero australe all'emisfero boreale, viceversa per il nodo discendente.

Figura 11 - Nodo Ascendente e Discendente [credits: [9]]

Parliamo di orbita diretta se il veicolo percorre l'orbita con la stessa direzione di rotazione del corpo celeste attorno al quale è posizionata. L'angolo varierà tra *0°* e *90°*.

Figura 12 - Orbita Diretta [credits: [9]]

Parliamo invece di orbita retrograda nel caso in cui il veicolo viaggi in senso opposto a quello definito appena sopra. L'angolo in tal caso sarà compreso tra *90°* e *180°*.

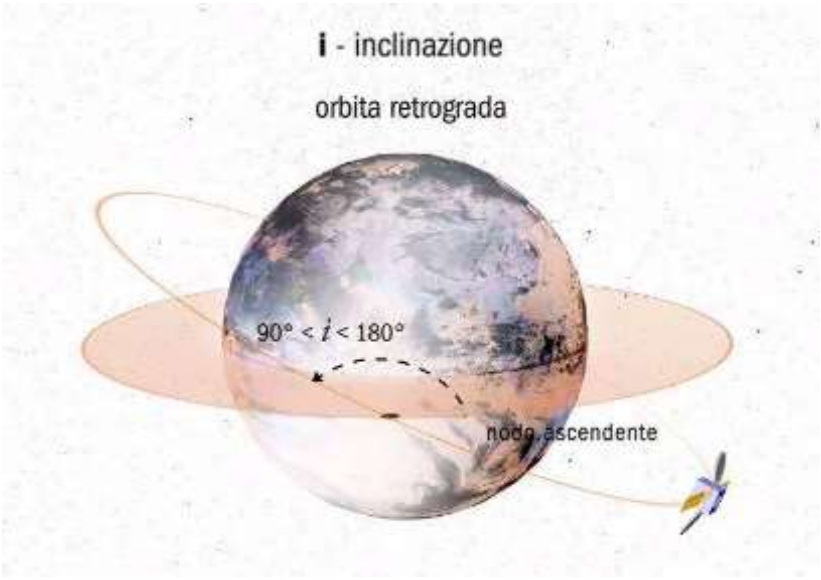

Figura 13 - Orbita Retrograda [credits: [9]]

- Ascensione retta del nodo ascendente (RAAN) Ω: definisce l'angolo compreso tra il primo punto d'ariete e il nodo ascendente. Può variare tra *0°* e *360°*.

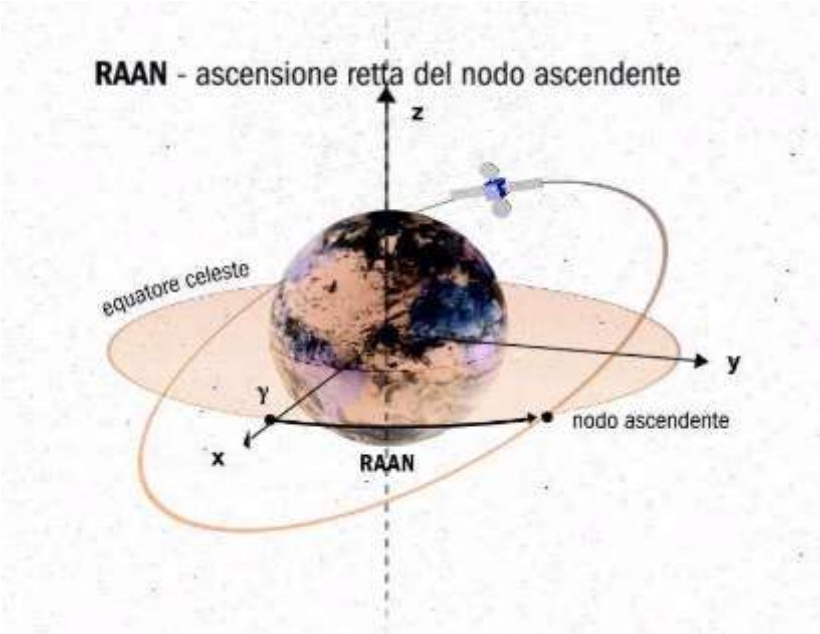

Figura 14 - Ascensione Retta del Nodo Ascendente [credits: [9]]

- Argomento del perigeo *ω*: rappresenta l'angolo compreso tra il perigeo e il nodo ascendente. I valori assunti posso variare tra *0°* e *360°*.

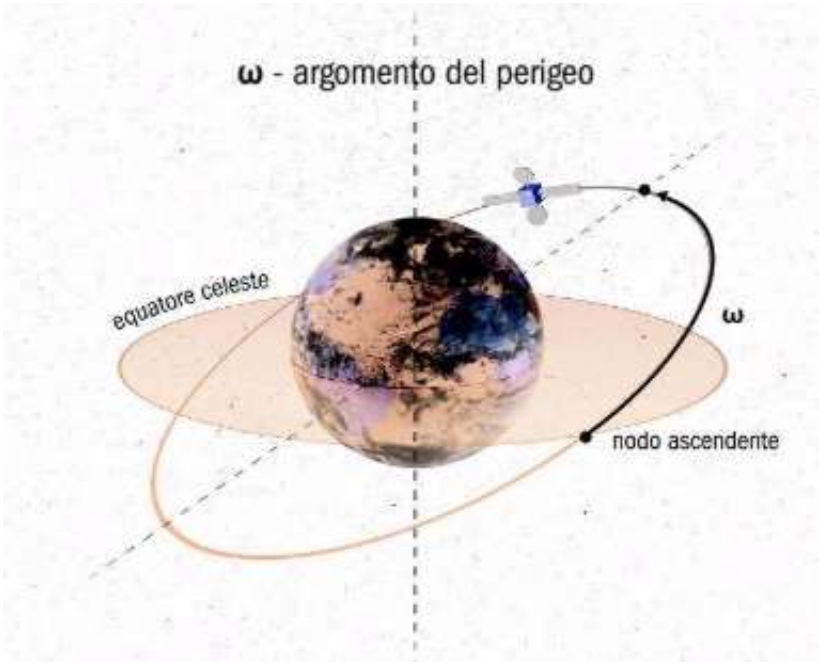

Figura 15 - Argomento del Perigeo [credits: [9]]

- Anomalia vera v: varia in funzione del tempo. Definisce l'angolo tra il perigeo e la posizione dello spacecraft in un determinato istante di tempo.

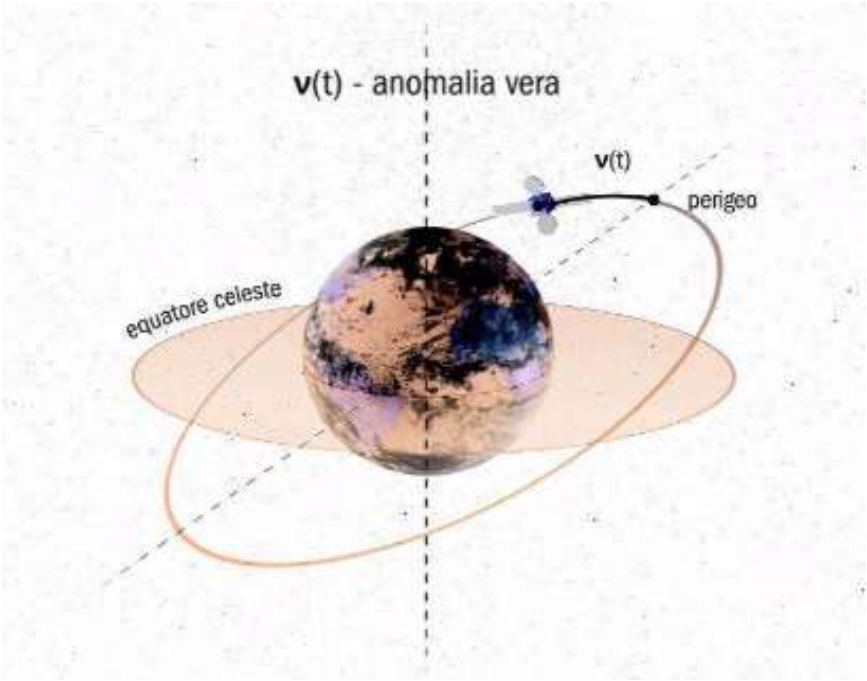

Figura 16 - Anomalia Vera [credits: [9]]

[credits: [9],[4]]

2.2 CIRCULAR RESTRICTED THREE BODY PROBLEM

Andiamo a considerare il moto di due corpi, m_1 e m_2. Essi si muovono, sotto l'azione del mutuo campo gravitazionale, con un'orbita circolare di raggio r_{12}. Il sistema di riferimento è un sistema xyz non inerziale, con origine nel centro di massa del sistema a due corpi, asse x diretto verso il corpo m_2, asse y giacente nel piano orbitale e asse z perpendicolare a completare la terna di assi. Assumiamo $m_1 > m_2$. Nel nostro caso andremo a identificare m_1 come la Terra e m_2 come la Luna. Al sistema viene aggiunta una terza massa m molto minore rispetto alle due precedenti. La massa m è considerata così piccola da non avere alcun effetto sul moto delle due masse primarie. Per il problema preso da noi in esame, andremo a identificare la massa m come il lander di cui vogliamo calcolare la traiettoria. Il problema così descritto è chiamato "Restricted 3 body problem". Non esiste una soluzione in forma chiusa per questo tipo di moto, ma possiamo comunque ricavarne le equazioni del moto.

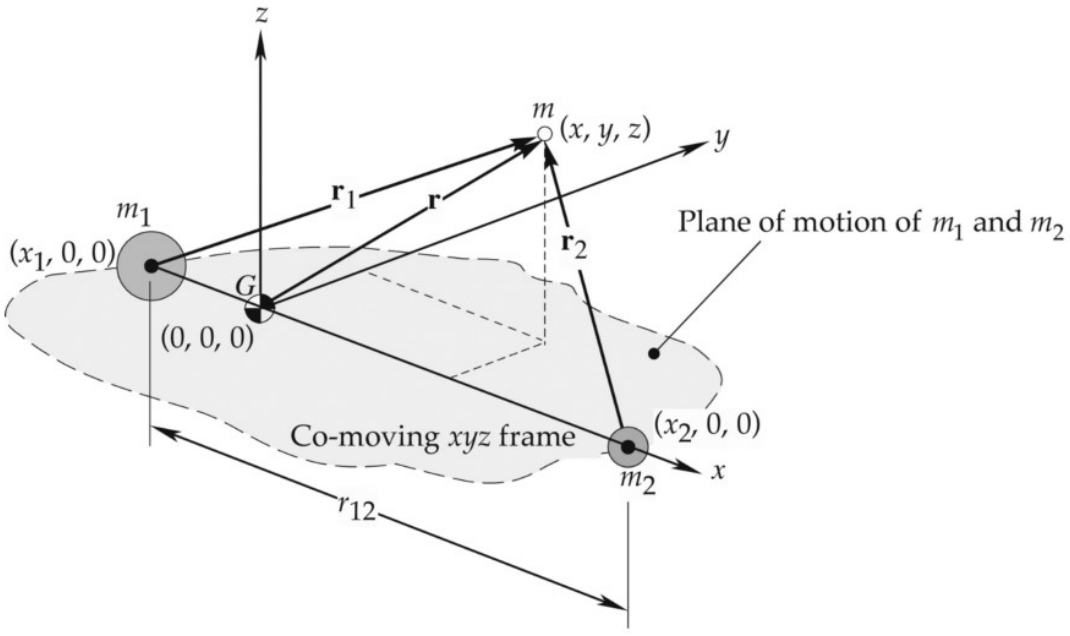

Figura 17 - Circular Restricted Three Body Problem [credits: [2]]

La massa totale del sistema dei due corpi primari m_1 e m_2 è:

$$M = m_1 + m_2$$

Definiamo dunque lo standard gravitational parameter μ come il prodotto fra la massa M e la gravitational constant G:

$$\mu = GM$$

Possiamo ora definire il periodo orbitale *T:*

$$T = \frac{2\pi}{\sqrt{\mu}} r_{12}^{3/2}$$

E dunque la velocità angolare inerziale $\bar{\omega}$:

$$\bar{\omega} = \omega \hat{k}$$

In cui:

$$\omega = \frac{2\pi}{T} = \sqrt{\frac{\mu}{r_{12}^3}}$$

Essendo m_1 e m_2 sul piano orbitale, possiamo indicare come nulle le componenti della loro posizione sugli assi *y* e *z*. La posizione lungo *x* può essere calcolata utilizzando la definizione di centro di massa:

$$m_1 x_1 + m_2 x_2 = 0$$

Considerando che m_2 si trova a distanza r_{12} rispetto m_1 nella direzione positiva dell'asse *x,* possiamo allora scrivere:

$$x_2 = x_1 + r_{12}$$

Otteniamo dunque:

$$x_1 = -\frac{m_2}{m_1 + m_2} r_{12}$$

$$x_2 = \frac{m_1}{m_1 + m_2} r_{12}$$

Definiamo i mass ratio π_1 e π_2:

$$\pi_1 = \frac{m_1}{m_1 + m_2}$$

$$\pi_2 = \frac{m_2}{m_1 + m_2}$$

Riscriviamo quindi le posizioni x_1 e x_2 delle masse m_1 e m_2 lungo *x*:

$$x_1 = -\pi_2 r_{12}$$

$$x_2 = \pi_1 r_{12}$$

Possiamo ora ricavare il vettore posizione della massa secondaria *m* rispetto a m_1:

$$\bar{r}_1 = (x - x_1)\hat{\imath} + y\hat{\jmath} + z\hat{k} = (x + \pi_2 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}$$

Rispetto a m_2:

$$\bar{r}_2 = (x - \pi_1 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}$$

E rispetto al centro di massa:

$$\bar{r} = x\hat{\imath} + y\hat{\jmath} + z\hat{k}$$

La velocità assoluta di *m* sarà calcolata come:

$$\dot{\bar{r}} = \bar{v}_G + \bar{\omega} \times \bar{r} + \bar{v}_{rel}$$

In cui $\bar{v}_G$ è la velocità inerziale del centro di massa e $\bar{v}_{rel}$ è la velocità di *m* nel sistema di riferimento non inerziale *xyz*, cioè:

$$\bar{v}_{rel} = \dot{x}\hat{\imath} + \dot{y}\hat{\jmath} + \dot{z}\hat{k}$$

L'accelerazione assoluta di *m* è calcolata usando la "five-term" relative acceleration formula:

$$\ddot{\bar{r}} = \bar{a}_G + \dot{\bar{\omega}} \times \bar{r} + \bar{\omega} \times (\bar{\omega} \times \bar{r}) + 2\bar{\omega} \times \bar{v}_{rel} + \bar{a}_{rel}$$

La velocità del centro di massa è costante, perciò possiamo fare la seguente semplificazione:

$$\bar{a}_G = \bar{0}$$

La velocità angolare dell'orbita circolare è anch'essa costante, possiamo quindi semplificare ulteriormente:

$$\dot{\bar{\omega}} = \bar{0}$$

L'accelerazione assoluta diventa:

$$\ddot{\bar{r}} = \bar{\omega} \times (\bar{\omega} \times \bar{r}) + 2\bar{\omega} \times \bar{v}_{rel} + \bar{a}_{rel}$$

In cui:

$$\bar{a}_{rel} = \ddot{x}\hat{\imath} + \ddot{y}\hat{\jmath} + \ddot{z}\hat{k}$$

Possiamo esplicitare l'equazione dell'accelerazione assoluta tenendo in considerazione le definizioni di $\bar{r}$, $\bar{v}_{rel}$ e $\bar{a}_{rel}$, nonché quella di $\bar{\omega}$. Essa diventa:

$$\ddot{\bar{r}} = \left(\omega\hat{k}\right) \times \left[\left(\omega\hat{k}\right) \times \left(x\hat{\imath} + y\hat{\jmath} + z\hat{k}\right)\right] + 2\left(\omega\hat{k}\right) \times \left(\dot{x}\hat{\imath} + \dot{y}\hat{\jmath} + \dot{z}\hat{k}\right) + \ddot{x}\hat{\imath} + \ddot{y}\hat{\jmath} + \ddot{z}\hat{k}$$
$$= -\omega^2(x\hat{\imath} + y\hat{\jmath}) + 2\omega\dot{x}\hat{\jmath} - 2\omega\dot{y}\hat{\imath} + \ddot{x}\hat{\imath} + \ddot{y}\hat{\jmath} + \ddot{z}\hat{k}$$

Mettendo in evidenza otteniamo:

$$\ddot{\bar{r}} = (\ddot{x} - 2\omega\dot{y} - \omega^2 x)\hat{\imath} + (\ddot{y} + 2\omega\dot{x} - \omega^2 y)\hat{\jmath} + \ddot{z}\hat{k}$$

Ora che abbiamo definito l'accelerazione inerziale della massa *m*, la seconda legge di Newton per tale massa è:

$$m\ddot{\bar{r}} = \bar{F}_1 + \bar{F}_2$$

Dove $\bar{F}_1$ e $\bar{F}_2$ sono le forze gravitazionali esercitate rispettivamente dalle masse m_1 e m_2 sulla massa *m*. Esse sono calcolate come:

$$\bar{F}_1 = -\frac{Gm_1 m}{r_1^2}\hat{u}_{r_1} = -\frac{\mu_1 m}{r_1^3}\bar{r}_1$$

$$\bar{F}_2 = -\frac{Gm_2 m}{r_2^2}\hat{u}_{r_2} = -\frac{\mu_2 m}{r_2^3}\bar{r}_2$$

In cui:

$$\mu_1 = Gm_1$$

$$\mu_2 = Gm_2$$

Inseriamo $\bar{F}_1$ e $\bar{F}_2$ appena definite all'interno della seconda legge di Newton per la massa *m*:

$$m\ddot{\bar{r}} = -\frac{\mu_1 m}{r_1^3}\bar{r}_1 - \frac{\mu_2 m}{r_2^3}\bar{r}_2$$

Semplificando la massa *m* otteniamo:

$$\ddot{\bar{r}} = -\frac{\mu_1}{r_1^3}\bar{r}_1 - \frac{\mu_2}{r_2^3}\bar{r}_2$$

Esplicitiamo tale relazione utilizzando la definizione data precedentemente di $\bar{r}_1$ e $\bar{r}_2$:

$$\ddot{\bar{r}} = -\frac{\mu_1}{r_1^3}[(x + \pi_2 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}] - \frac{\mu_2}{r_2^3}[(x - \pi_1 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}]$$

Eguagliamo questa relazione a quella trovata precedentemente di $\ddot{\bar{r}}$:

$$(\ddot{x} - 2\omega\dot{y} - \omega^2 x)\hat{\imath} + (\ddot{y} + 2\omega\dot{x} - \omega^2 y)\hat{\jmath} + \ddot{z}\hat{k} = -\frac{\mu_1}{r_1^3}[(x + \pi_2 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}] - \frac{\mu_2}{r_2^3}[(x - \pi_1 r_{12})\hat{\imath} + y\hat{\jmath} + z\hat{k}]$$

Andando ad eguagliare membro a membro le tre componenti del vettore, otteniamo le tre equazioni del moto scalari del three body problem:

$$\ddot{x} = -\frac{\mu_1}{r_1^3}(x + \pi_2 r_{12}) - \frac{\mu_2}{r_2^3}(x - \pi_1 r_{12}) + \omega^2 x + 2\omega\dot{y}$$

$$\ddot{y} = -\frac{\mu_1}{r_1^3}y - \frac{\mu_2}{r_2^3}y + \omega^2 y - 2\omega\dot{x}$$

$$\ddot{z} = -\frac{\mu_1}{r_1^3}z - \frac{\mu_2}{r_2^3}z$$

In forma vettoriale:

$$\bar{g}(\bar{r}, \bar{v}) = -\frac{\mu_1}{|\bar{r} - \bar{r}_1|^3}(\bar{r} - \bar{r}_1) - \frac{\mu_2}{|\bar{r} - \bar{r}_2|^3}(\bar{r} - \bar{r}_2) - \bar{\omega} \times (\bar{\omega} \times \bar{r}) - 2\bar{\omega} \times \bar{v}_{rel}$$

Possiamo riscrivere quest'ultima equazione vettoriale come:

$$\bar{g}(\bar{r}, \bar{v}) = \bar{g}_1(\bar{r}) + \bar{g}_2(\bar{r}) + \bar{g}_{cen}(\bar{r}) + \bar{g}_{cor}(\bar{v})$$

In cui:

$$\bar{g}_1(\bar{r}) = -\frac{\mu_1}{|\bar{r} - \bar{r}_1|^3}(\bar{r} - \bar{r}_1)$$

$$\bar{g}_2(\bar{r}) = -\frac{\mu_2}{|\bar{r} - \bar{r}_2|^3}(\bar{r} - \bar{r}_2)$$

$$\bar{g}_{cen}(\bar{r}) = -\bar{\omega} \times (\bar{\omega} \times \bar{r})$$

$$\bar{g}_{cor}(\bar{v}) = -2\bar{\omega} \times \bar{v}_{rel}$$

Sono rispettivamente: il campo gravitazionale terrestre ($\bar{g}_1(\bar{r})$), il campo gravitazionale lunare ($\bar{g}_2(\bar{r})$), l'accelerazione centrifuga ($\bar{g}_{cen}(\bar{r})$), l'accelerazione di Coriolis ($\bar{g}_{cor}(\bar{v})$).

[credits: [2],[10]]

2.3 PUNTI DI LAGRANGE

Riprendiamo le equazioni scalari del CR3BP ricavate al paragrafo precedente:

$$\ddot{x} = -\frac{\mu_1}{r_1^3}(x + \pi_2 r_{12}) - \frac{\mu_2}{r_2^3}(x - \pi_1 r_{12}) + \omega^2 x + 2\omega\dot{y}$$

$$\ddot{y} = -\frac{\mu_1}{r_1^3}y - \frac{\mu_2}{r_2^3}y + \omega^2 y - 2\omega\dot{x}$$

$$\ddot{z} = -\frac{\mu_1}{r_1^3}z - \frac{\mu_2}{r_2^3}z$$

Possiamo utilizzarle per la ricerca dei punti di equilibrio. I punti di equilibrio (o punti di Lagrange) rappresentano dei luoghi nello spazio in cui la massa secondaria *m* avrebbe velocità e accelerazione nulla. Una volta posizionatasi in uno di questi punti, la massa *m* rimarrebbe presumibilmente in questa posizione. I punti di Lagrange sono dunque definiti dalle condizioni:

$$\dot{x} = \dot{y} = \dot{z} = 0$$
$$\ddot{x} = \ddot{y} = \ddot{z} = 0$$

Andando a sostituire queste condizioni nelle equazioni sopra:

$$-\omega^2 x = -\frac{\mu_1}{r_1^3}(x + \pi_2 r_{12}) - \frac{\mu_2}{r_2^3}(x - \pi_1 r_{12})$$
$$-\omega^2 y = -\frac{\mu_1}{r_1^3} y - \frac{\mu_2}{r_2^3} y$$
$$0 = -\frac{\mu_1}{r_1^3} z - \frac{\mu_2}{r_2^3} z$$

Da questa ultima relazione:

$$\left(\frac{\mu_1}{r_1^3} + \frac{\mu_2}{r_2^3}\right) z = 0$$

Dato che: $\frac{\mu_1}{r_1^3} > 0$ e $\frac{\mu_2}{r_2^3} > 0$ sarà:

$$z = 0$$

I punti di equilibrio giacciono quindi nel piano orbitale.

Sapendo invece che:

$$\pi_1 = 1 - \pi_2$$

E:

$$\omega = \sqrt{\frac{\mu}{r_{12}^3}}$$

Sfruttando:

$$\pi_1 = \frac{\mu_1}{\mu}\,, \qquad \pi_2 = \frac{\mu_2}{\mu}$$

E assumendo $y \neq 0$, possiamo riscrivere anche le altre due relazioni:

$$(1 - \pi_2)(x + \pi_2 r_{12})\frac{1}{r_1^3} + \pi_2(x + \pi_2 r_{12} - r_{12})\frac{1}{r_2^3} = \frac{x}{r_{12}^3}$$
$$(1 - \pi_2)\frac{1}{r_1^3} + \pi_2 \frac{1}{r_2^3} = \frac{1}{r_{12}^3}$$

Trattandole come due equazioni lineari in $\frac{1}{r_1^3}$ e $\frac{1}{r_2^3}$ possiamo risolverle simultaneamente ottenendo che:

$$\frac{1}{r_1^3} = \frac{1}{r_2^3} = \frac{1}{r_{12}^3}$$

Quindi:

$$r_1 = r_2 = r_{12}$$

Sfruttando questo risultato, insieme a $z = 0$ e a $\pi_1 = 1 - \pi_2$ possiamo ricavare dalle equazioni di $\bar{r}_1$ e $\bar{r}_2$:

$$r_{12}^2 = (x + \pi_2 r_{12})^2 + y^2$$
$$r_{12}^2 = (x + \pi_2 r_{12} - r_{12})^2 + y^2$$

Andando ad eguagliare queste ultime relazioni giungiamo a:

$$x = \frac{r_{12}}{2} - \pi_2 r_{12}$$

Sostituendolo in una delle due relazioni di partenza avremo:

$$y = \pm \frac{\sqrt{3}}{2} r_{12}$$

Abbiamo trovato le coordinate di due punti di Lagrange, L_4 e L_5:

$$x = \frac{r_{12}}{2} - \pi_2 r_{12}, \qquad y = \pm \frac{\sqrt{3}}{2} r_{12}, \qquad z = 0$$

Questi due punti distano da entrambi i corpi primari, m_1 e m_2 la stessa distanza r_{12} che i due corpi distano l'uno dall'altro. Ne deduciamo che i due punti di Lagrange in esame e i due corpi primari giacciono ai vertici di due triangoli equilateri.
I tre punti di Lagrange rimanenti vengono calcolati ponendo $y = 0$ oltre che $z = 0$. Le equazioni di $\bar{r}_1$ e $\bar{r}_2$ allora diventano:

$$\bar{r}_1 = (x + \pi_2 r_{12})\hat{\imath}$$
$$\bar{r}_2 = (x - \pi_1 r_{12})\hat{\imath} = (x + \pi_2 r_{12} - r_{12})\hat{\imath}$$

Da cui:

$$r_1 = |x + \pi_2 r_{12}|$$
$$r_2 = |x + \pi_2 r_{12} - r_{12}|$$

Sostituendo queste relazioni, insieme a $\omega = \sqrt{\frac{\mu}{r_{12}^3}}$, $\pi_1 = \frac{\mu_1}{\mu}$, $\pi_2 = \frac{\mu_2}{\mu}$ e $\pi_1 = 1 - \pi_2$ all'interno di:

$$-\omega^2 x = -\frac{\mu_1}{r_1^3}(x + \pi_2 r_{12}) - \frac{\mu_2}{r_2^3}(x - \pi_1 r_{12})$$

Otteniamo:

$$(1 - \pi_2)\frac{x + \pi_2 r_{12}}{|x + \pi_2 r_{12}|^3} + \pi_2 \frac{x + \pi_2 r_{12} - r_{12}}{|x + \pi_2 r_{12} - r_{12}|^3} - \frac{1}{r_{12}^3} x = 0$$

Un'ulteriore semplificazione si ottiene adimensionalizzando x:

$$\xi = \frac{x}{r_{12}}$$

L'equazione diventa:

$$f(\pi_2, \xi) = 0$$

Con:

$$f(\pi_2, \xi) = (1 - \pi_2)\frac{\xi + \pi_2}{|\xi + \pi_2|^3} + \pi_2 \frac{\xi + \pi_2 - 1}{|\xi + \pi_2 - 1|^3} - \xi$$

Riportiamo quindi la curva rappresentante il luogo dei punti per cui $f(\pi_2, \xi) = 0$:

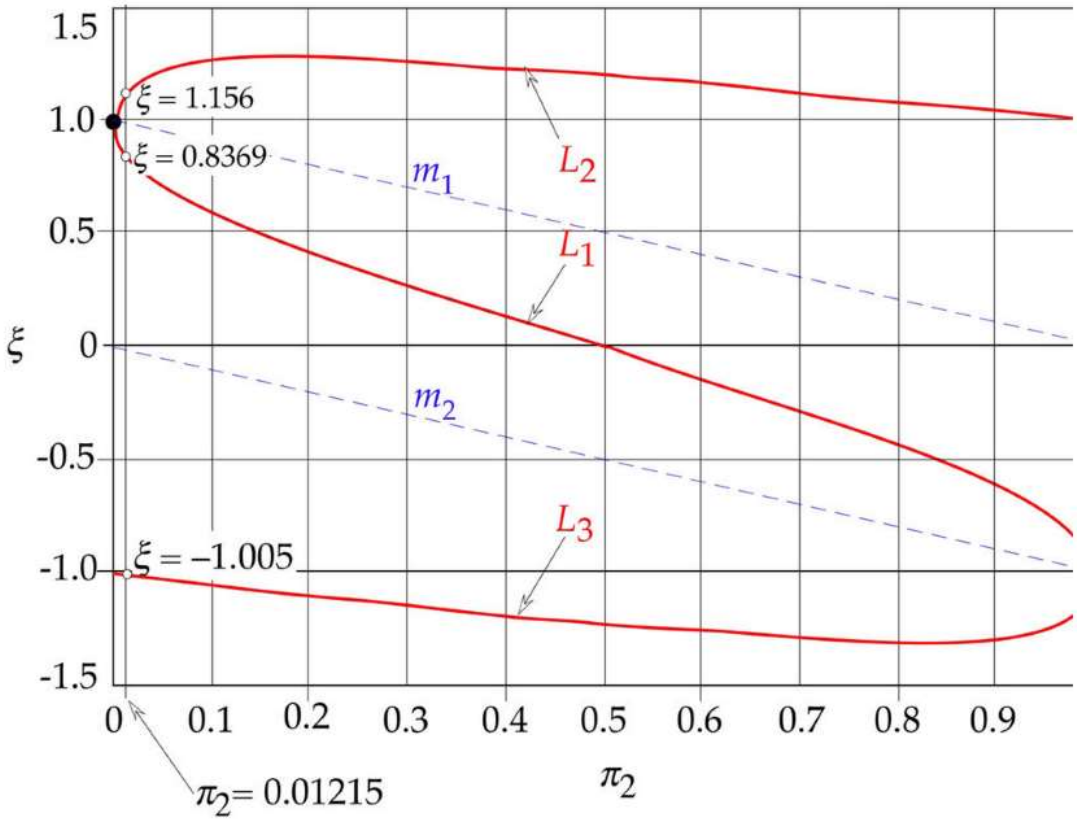

Figura 18 - Curva di f(π_2,ξ)=0 [credits: [2]]

Le linee orizzontali $\xi = 1$, $\xi = -1$ dividono la curva in tre curve distinte: L_1, L_2, L_3. Per ogni valore di $0 < \pi_2 < 1$ otterremo quindi tre valori di ξ. Le curve m_1 e m_2 rappresentano la forma adimensionalizzata delle equazioni di x_1 e x_2:

$$m_1: \ \xi + \pi_2 = 0$$
$$m_2: \ \xi + \pi_2 - 1 = 0$$

Assumendo che la massa m_2 sia posizionata a destra della massa m_1, i punti di Lagrange collineari saranno posizionati come segue: L_3 a sinistra di m_1, L_2 a destra di m_2 e L_1 tra le due masse m_1 e m_2.

Per ottenere la coordinata x dei punti di Lagrange L_1, L_2 e L_3 basterà moltiplicare i valori di ξ_1, ξ_2 e ξ_3 per la distanza r_{12}:

$$x_1 = \xi_1 r_{12}$$
$$x_2 = \xi_2 r_{12}$$
$$x_3 = \xi_3 r_{12}$$

In figura vengono inoltre mostrati il valore di mass ratio π_2 del sistema Terra-Luna e i corrispondenti valori di ξ_1, ξ_2 e ξ_2.

La disposizione dei cinque punti di Lagrange del sistema Terra-Luna è dunque la seguente:

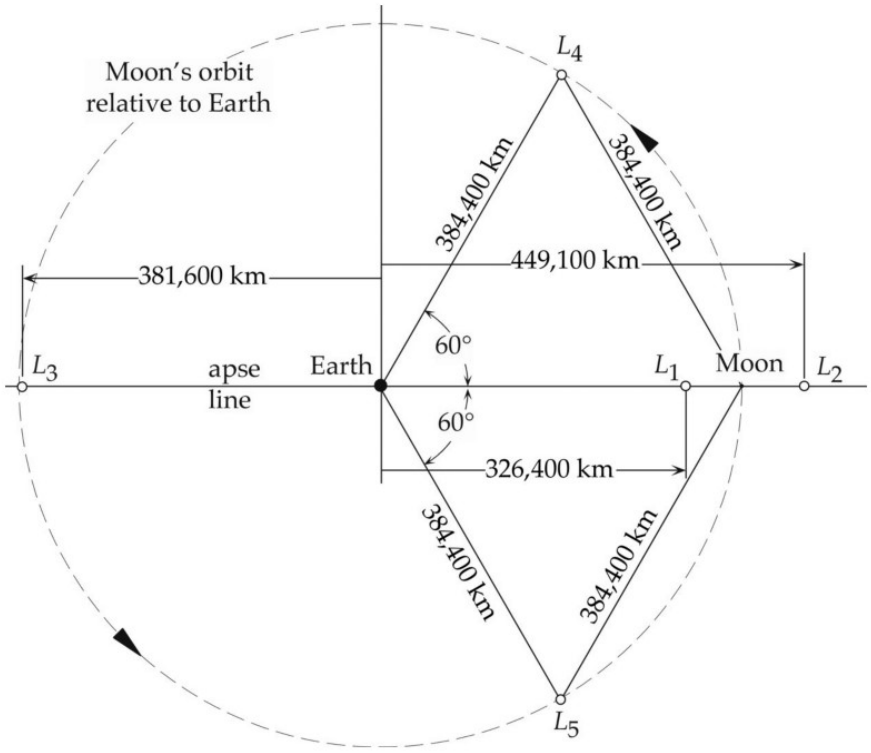

Figura 19 - Punti di Lagrange nel Sistema Terra-Luna [credits: [2]]

Questi punti orbitano attorno alla Terra con lo stesso periodo del suo satellite naturale.

Dovremo a questo punto analizzare la stabilità di questi punti. Scriviamo le equazioni del moto come:

$$\dot{\bar{p}} = \bar{F}(\bar{p})$$

E consideriamo un punto fisso $\bar{p}_0$ per cui:

$$\bar{F}(\bar{p}_0) = 0$$

Allora la dinamica della variazione lineare è data da:

$$\dot{\bar{p}}_0 + \delta\dot{\bar{p}} = \bar{F}(\bar{p}_0 + \delta\bar{p}) \approx \bar{F}(\bar{p}_0) + [\partial_p \bar{F}(\bar{p}_0)]\delta\bar{p}$$

$$\delta\dot{\bar{p}} = [\partial_p \bar{F}(\bar{p}_0)]\delta\bar{p}$$

Possiamo studiare la stabilità lineare del sistema in quel punto andando a valutare gli autovalori del Jacobiano $J = \partial_p \bar{F}(\bar{p}_0)$. Questo perché, se λ è un autovalore della matrice J e $\delta\bar{p}_\lambda$ l'autovettore corrispondente, la perturbazione in quella direzione sarà descritta da:

$$\delta\bar{p}(t) = e^{\lambda t}\delta\bar{p}_\lambda$$

Se la parte reale dell'autovalore è positiva la perturbazione verrà amplificata e tenderà ad incrementare nel tempo. Se al contrario la parte reale dell'autovalore è negativa, la perturbazione verrà smorzata e tenderà verso il valore nullo. Il caso con parte reale dell'autovalore nulla e parte immaginaria diversa da zero, porta invece ad un'oscillazione costante attorno al punto di equilibrio. Per il nostro ambito di applicazione la dinamica completa del sistema è data da:

$$\frac{d}{dt}\begin{bmatrix}\bar{r}\\ \bar{v}\end{bmatrix} = \begin{bmatrix}F_r\\ F_v\end{bmatrix} = \begin{bmatrix}\bar{v}\\ \bar{a}_{rel}(\bar{r},\bar{v})\end{bmatrix}$$

E il Jacobiano è dato da:

$$J = \begin{bmatrix}\partial_r F_r & \partial_v F_r\\ \partial_r F_v & \partial_v F_v\end{bmatrix} = \begin{bmatrix}0 & I_3\\ \partial_r \bar{a}_{rel} & \partial_v \bar{a}_{rel}\end{bmatrix}$$

Dove I_3 rappresenta la matrice identità di dimensione *3x3*, $\partial_r \bar{a}_{rel}$ è calcolato come:

$$\partial_r \bar{a}_{rel} = 3\frac{\mu_1}{|\bar{r}-\bar{r}_1|^5}(\bar{r}-\bar{r}_1)(\bar{r}-\bar{r}_1)^T - \frac{\mu_1}{|\bar{r}-\bar{r}_1|^3}I_3 + 3\frac{\mu_2}{|\bar{r}-\bar{r}_2|^5}(\bar{r}-\bar{r}_2)(\bar{r}-\bar{r}_2)^T - \frac{\mu_1}{|\bar{r}-\bar{r}_2|^3}I_3 + \omega^2 I_3 - \bar{\omega}\bar{\omega}^T$$

E $\partial_v \bar{a}_{rel}$ è calcolato come:

$$\partial_v \bar{a}_{rel} = -2[\bar{\omega}]_x$$

Possiamo ricercare gli autovalori della matrice Jacobiana appena definita andando a risolvere la seguente equazione:

$$\det(J - \lambda I_6) = 0$$

$$\det\left(\begin{bmatrix}-\lambda I_3 & I_3\\ \partial_r \bar{a}_{rel} & \partial_v \bar{a}_{rel} - \lambda I_3\end{bmatrix}\right) = 0$$

Possiamo riscrivere questa matrice in modo da computare più facilmente il determinante:

$$\begin{bmatrix}-\lambda I_3 & I_3\\ \partial_r \bar{a}_{rel} & \partial_v \bar{a}_{rel} - \lambda I_3\end{bmatrix} = \underbrace{\begin{bmatrix}I_3 & 0\\ -\frac{1}{\lambda}\partial_r \bar{a}_{rel} & \lambda^2 I_3 - \lambda\partial_v \bar{a}_{rel} - \partial_r \bar{a}_{rel}\end{bmatrix}}_{\det(\lambda^2 I_3 - \lambda\partial_v \bar{a}_{rel} - \partial_r \bar{a}_{rel})} \underbrace{\begin{bmatrix}-\lambda I_3 & I_3\\ 0 & -\frac{1}{\lambda}I_3\end{bmatrix}}_{det = 1}$$

Quindi, in maniera equivalente, possiamo porre:

$$\det(\lambda^2 I_3 - \lambda\partial_v \bar{a}_{rel} - \partial_r \bar{a}_{rel}) = 0$$

Per quanto riguarda i punti di equilibrio L_1, L_2, L_3 abbiamo *y=z=0*. Possiamo quindi scrivere:

$$\lambda^2 I_3 - \lambda\partial_v \bar{a}_{rel} - \partial_r \bar{a}_{rel} = \begin{bmatrix}\lambda^2 - \omega^2 - 2k & -2\omega\lambda & 0\\ 2\omega\lambda & \lambda^2 - \omega^2 + k & 0\\ 0 & 0 & \lambda^2 - k\end{bmatrix}$$

In cui:

$$k = \frac{\mu_1}{|\bar{r}-\bar{r}_1|^3} + \frac{\mu_2}{|\bar{r}-\bar{r}_2|^3}$$

Ponendo quindi il determinante uguale a zero otteniamo:

$$(\lambda^2 + k)(\lambda^4 + \lambda^2(2\omega^2 - k) + \omega^4 + \omega^2 k - 2k^2) = 0$$

Ne risultano tre paia di autovalori:

$$\lambda = \pm\sqrt{k}i$$

$$\lambda = \pm\sqrt{\frac{k}{2} - \omega^2 + \sqrt{\frac{9k^2}{4} - 2\omega^2 k}}$$

$$\lambda = \pm \sqrt{\frac{k}{2} - \omega^2 - \sqrt{\frac{9k^2}{4} - 2\omega^2 k}}$$

Può essere dimostrato che c'è sempre un paio di autovalori reali. Ne risulta che i punti L_1, L_2, L_3 sono sempre instabili.

Ora occupiamoci dei punti di Lagrange L_4, L_5. Avremo:

$$\lambda^2 I_3 - \lambda \partial_v \bar{a}_{rel} - \partial_r \bar{a}_{rel} = \begin{bmatrix} \lambda^2 - \frac{3}{4}\omega^2 & \pm\frac{3\sqrt{3}}{4}\omega^2\gamma - 2\omega\lambda & 0 \\ \pm\frac{3\sqrt{3}}{4}\omega^2\gamma + 2\omega\lambda & \lambda^2 - \frac{9}{4}\omega^2 & 0 \\ 0 & 0 & \lambda^2 + \omega^2 \end{bmatrix}$$

In cui:

$$\gamma = \frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}$$

Annullando quindi il determinante si ottiene:

$$(\lambda^2 + \omega^2)\left(\lambda^4 + \omega^2\lambda^2 + \frac{27}{16}\omega^4(1 - \gamma^2)\right) = 0$$

Ne ricaviamo tre paia di autovalori:

$$\lambda = \pm\omega i$$

$$\lambda = \pm\omega \sqrt{-\frac{1}{2} + \frac{1}{4}\sqrt{27\gamma^2 - 23}}$$

$$\lambda = \pm\omega \sqrt{-\frac{1}{2} - \frac{1}{4}\sqrt{27\gamma^2 - 23}}$$

Per far sì che i punti di Lagrange in esame siano stabili, gli autovalori devono essere tutti immaginari. Si può dimostrare che questo è vero se il mass ratio è circa:

$$\frac{\mu_1}{\mu_2} \geq 25$$

Dato che il mass ratio del sistema Terra-Luna è di circa *81*, L_4 e L_5 risultano stabili. L'orbita di partenza, su cui si muove la LOP-G, è un'orbita quasi stabile.

[credits: [2],[10]]

2.4 COSTANTE DI JACOBI

Riprendiamo le equazioni scalari del CR3BP e moltiplichiamole rispettivamente per $\dot{x}$, $\dot{y}$, $\dot{z}$:

$$\ddot{x}\dot{x} = -\frac{\mu_1}{r_1^3}(x\dot{x} + \pi_2 r_{12}\dot{x}) - \frac{\mu_2}{r_2^3}(x\dot{x} - \pi_1 r_{12}\dot{x}) + \omega^2 x\dot{x} + 2\omega\dot{x}\dot{y}$$

$$\ddot{y}\dot{y} = -\frac{\mu_1}{r_1^3}y\dot{y} - \frac{\mu_2}{r_2^3}y\dot{y} + \omega^2 y\dot{y} - 2\omega\dot{x}\dot{y}$$

$$\ddot{z}\dot{z} = -\frac{\mu_1}{r_1^3} z\dot{z} - \frac{\mu_2}{r_2^3} z\dot{z}$$

Sommando il lato destro e sinistro delle equazioni e con qualche riaggiustamento otteniamo:

$$\ddot{x}\dot{x} + \ddot{y}\dot{y} + \ddot{z}\dot{z} - \omega^2(x\dot{x} + y\dot{y}) = -\frac{\mu_1}{r_1^3}(x\dot{x} + y\dot{y} + z\dot{z} + \pi_2 r_{12}\dot{x}) - \frac{\mu_2}{r_2^3}(x\dot{x} + y\dot{y} + z\dot{z} - \pi_1 r_{12}\dot{x})$$

Da notare che:

$$\ddot{x}\dot{x} + \ddot{y}\dot{y} + \ddot{z}\dot{z} = \frac{1}{2}\frac{d}{dt}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) = \frac{1}{2}v^2$$

Dove v è la velocità della massa secondaria relativa al sistema di riferimento in rotazione.

Similmente:

$$\ddot{x}\dot{x} + \ddot{y}\dot{y} = \frac{1}{2}\frac{d}{dt}(x^2 + y^2)$$

Dall'equazione di $\bar{r}_1$ possiamo ricavare:

$$r_1^2 = (x + \pi_2 r_{12})^2 + y^2 + z^2$$

Perciò:

$$\frac{dr_1}{dt} = \frac{1}{r_1}(\pi_2 r_{12}\dot{x} + x\dot{x} + y\dot{y} + z\dot{z})$$

Segue che:

$$\frac{d}{dt}\frac{1}{r_1} = -\frac{1}{r_1^2}\frac{dr_1}{dt} = -\frac{1}{r_1^3}(x\dot{x} + y\dot{y} + z\dot{z} + \pi_2 r_{12}\dot{x})$$

Allo stesso modo, dall'equazione di $\bar{r}_2$ otteniamo:

$$\frac{d}{dt}\frac{1}{r_2} = -\frac{1}{r_2^3}(x\dot{x} + y\dot{y} + z\dot{z} + \pi_1 r_{12}\dot{x})$$

Sostituendo queste ultime equazioni nell'equazione della somma lato destro e sinistro e riarrangiando i termini otteniamo:

$$\frac{d}{dt}\left[\frac{1}{2}v^2 - \frac{1}{2}\omega^2(x^2 + y^2) - \frac{\mu_1}{r_1} - \frac{\mu_2}{r_2}\right] = 0$$

Il che significa che l'espressione all'interno della parentesi è costante:

$$\frac{1}{2}v^2 - \frac{1}{2}\omega^2(x^2 + y^2) - \frac{\mu_1}{r_1} - \frac{\mu_2}{r_2} = C$$

$\frac{1}{2}v^2$ rappresenta l'energia cinetica per unità di massa relativa al sistema di riferimento rotante, $-\frac{\mu_1}{r_1}$ e $-\frac{\mu_2}{r_2}$ sono invece le energie potenziali gravitazionali delle due masse primarie, $-\frac{1}{2}\omega^2(x^2 + y^2)$ potrebbe essere interpretato come l'energia potenziale della forza centrifuga per unità di massa indotta dalla rotazione del sistema di riferimento. La costante C è detta costante di Jacobi. Essa rappresenta l'energia totale della massa secondaria relativa al sistema di riferimento in rotazione. Risolvendo l'ultima equazione per v^2 avremo:

$$v^2 = \omega^2(x^2 + y^2) + 2\frac{\mu_1}{r_1} + 2\frac{\mu_2}{r_2} + 2C$$

Dato che v^2 non può assumere valori negativi, dovrà essere vero che:

$$\omega^2(x^2 + y^2) + 2\frac{\mu_1}{r_1} + 2\frac{\mu_2}{r_2} + 2C \geq 0$$

Non sono consentite per la massa secondaria traiettorie in regioni dove questa diseguaglianza è violata. Le condizioni di confine tra zone permesse e zone non consentite sono definite ponendo la seguente uguaglianza: $v^2 = 0$. Questo si traduce in:

$$\omega^2(x^2 + y^2) + 2\frac{\mu_1}{r_1} + 2\frac{\mu_2}{r_2} + 2C = 0$$

Sfruttando questa equazione, per un valore dato della costante di Jacobi, possiamo tracciare le curve a zero velocità. Dato che i primi tre termini dell'equazione sono positivi, tali curve corrispondono a valori negativi della costante C. Grossi valori di C corrispondono ad una posizione del corpo secondario molto lontana dal centro di massa del sistema o anche vicina ad una delle due masse primarie. Considerando sempre il sistema Terra-Luna, possiamo tracciare le curve a zero velocità per diversi valori di C. Osserviamo la figura:

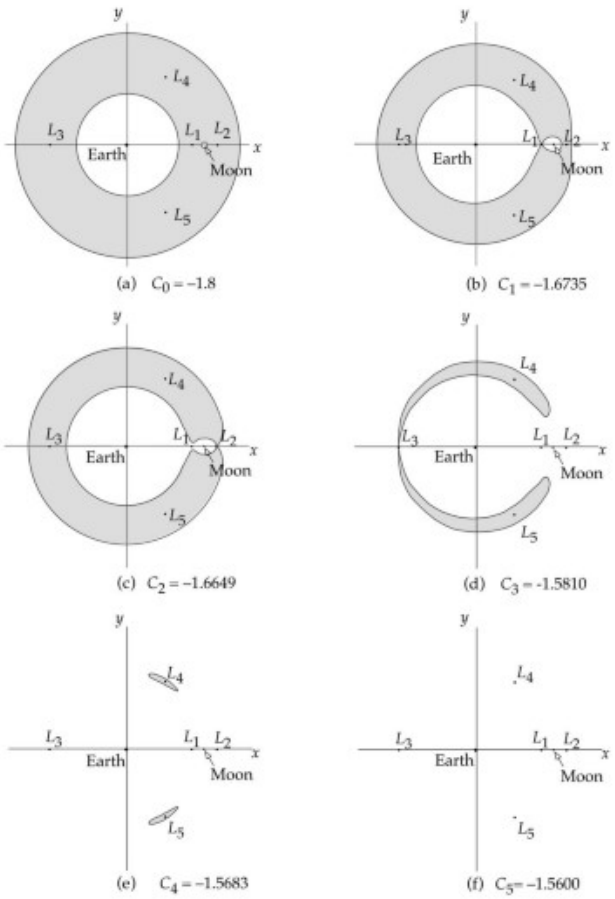

Figura 20 - Curve a Velocità Zero per Diversi Valori della Costante di Jacobi [credits: [2]]

Per $C_0 = -1.8\,{}^{Km^2}/_{s^2}$ le zone consentite sono solo due cerchi intorno alla Terra e alla Luna. Un veicolo lanciato con tale valore di C non potrebbe raggiungere la Luna, tantomeno uscire dal sistema Terra-Luna. Parliamo ora in termini di modulo delle costanti di Jacobi trovate. Sostituendo all'interno dell'equazione i punti di Lagrange L_1, L_2, L_3 troviamo valori più elevati della costante di Jacobi, C_1, C_2, C_3 che ci permettono di raggiungere i corrispondenti punti di Lagrange a velocità zero. C_2 rappresenta anche il valore minimo per cui si riesce a scappare dal sistema Terra-Luna, attraverso uno stretto corridoio attorno al satellite. Alzando ulteriormente il valore di C questo corridoio si amplia. Arrivati al valore di C_3 riusciamo a scappare dal sistema anche in direzione opposta alla Luna. Innalzando ulteriormente la costante di Jacobi, arrivando ad esempio a C_4 riusciamo a navigare all'interno del sistema Terra-Luna, fatta eccezione per delle zone proibite intorno ai punti di Lagrange L_4 e L_5 e a scappare da esso in qualsiasi direzione. Arrivando a valori ancora più elevati, rappresentati da C_5, riusciamo a raggiungere ogni punto del sistema e ad uscire da esso in ogni direzione.
[credits: [2]]

2.5 ANGOLI DI ASSETTO

Per specificare l'orientazione di un corpo in un sistema di riferimento inerziale, abbiamo bisogno di tre angoli. Possiamo utilizzare a tal scopo gli angoli di Eulero o gli angoli di assetto Yaw, Pitch e Roll. Ci focalizzeremo, per i nostri scopi, su questi ultimi. Tali angoli permetteranno di ruotare il sistema di riferimento inerziale XYZ nel sistema di riferimento di assi body fissi xyz, grazie a una sequenza di tre rotazioni elementari.

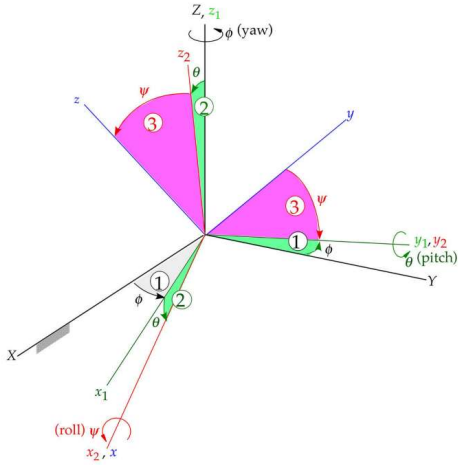

Figura 21 - Rotazioni elementari di Pitch, Yaw e Roll [credits: [2]]

La prima rotazione avverrà attorno all'asse Z, che coinciderà con l'asse z_1, con un angolo pari all'angolo di Yaw φ. Gli assi X e Y si trasformano rispettivamente negli assi x_1 e y_1. La seconda rotazione è attorno a y_1, che coinciderà con y_2, di un angolo di Pitch ϑ. Questo porta x_1 e z_1 a diventare rispettivamente x_2 e z_2. L'ultima rotazione sarà attorno all'asse x_2, che coinciderà con l'asse x, con un angolo di Roll ψ. y_2 e z_2 verranno trasformati rispettivamente in y e z.

La matrice di trasformazione da sistema di riferimento inerziale a sistema di rifermento assi corpo sarà data da:

$$[Q]_{Xx} = [R_1(\psi)][R_2(\vartheta)][R_3(\varphi)]$$

In cui le singole matrici di rotazione elementari sono date da:

$$R_1(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & sin\psi \\ 0 & -sin\psi & cos\psi \end{bmatrix}$$

$$R_2(\vartheta) = \begin{bmatrix} cos\vartheta & 0 & -sin\vartheta \\ 0 & 1 & 0 \\ sin\vartheta & 0 & cos\vartheta \end{bmatrix}$$

$$R_3(\varphi) = \begin{bmatrix} cos\varphi & sin\varphi & 0 \\ -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice di trasformazione risultante da questa moltiplicazione matriciale è:

$$[Q]_{Xx} = \begin{bmatrix} cos\varphi cos\vartheta & sin\varphi cos\vartheta & -sin\vartheta \\ cos\varphi sin\vartheta sin\psi - sin\varphi cos\psi & sin\varphi sin\vartheta sin\psi + cos\varphi cos\psi & cos\vartheta sin\psi \\ cos\varphi sin\vartheta cos\psi + sin\varphi sin\psi & sin\varphi sin\vartheta cos\psi - cos\varphi sin\psi & cos\vartheta cos\psi \end{bmatrix}$$

Per effettuare la trasformazione inversa dal sistema di riferimento xyz al sistema di riferimento XYZ, basta fare la trasposta della matrice sopra definita:

$$[Q]_{xX} = \begin{bmatrix} cos\varphi cos\vartheta & cos\varphi sin\vartheta sin\psi - sin\varphi cos\psi & cos\varphi sin\vartheta cos\psi + sin\varphi sin\psi \\ sin\varphi cos\vartheta & sin\varphi sin\vartheta sin\psi + cos\varphi cos\psi & sin\varphi sin\vartheta cos\psi - cos\varphi sin\psi \\ -sin\vartheta & cos\vartheta sin\psi & cos\vartheta cos\psi \end{bmatrix}$$

Gli angoli di assetto sono quindi definiti da:

$$tan\varphi = \frac{Q_{12}}{Q_{11}}, \qquad sin\vartheta = -Q_{13}, \qquad tan\psi = \frac{Q_{23}}{Q_{33}}$$

[credits: [2]]

2.6 EQUAZIONE DEL RAZZO

Le forze che agiscono su un razzo cambiano drasticamente durante un volo tipico. Il combustibile viene consumato causando una variazione di peso e massa durante il volo. A causa di questa variazione non è possibile utilizzare la seconda legge del moto di Newton per valutare l'accelerazione e la velocità del razzo. Lo scopo di questa sezione sarà quella di valutare la variazione di velocità considerando anche la variazione in massa del velivolo. Gli effetti di portanza e resistenza non saranno considerati.

Definiamo quindi M la massa istantanea del razzo, u la sua velocità, v la velocità di fuoriuscita dei gas esausti e p la loro pressione, p_0 è la pressione atmosferica, A l'area di fuoriuscita dei gas dall'ugello. Durante una infinitesima variazione di tempo dt, il veicolo subisce una variazione della sua massa dm. Possiamo quindi valutare la variazione di momento del razzo:

$$change\ in\ rocket\ momentum = M(u + du) - Mu = Mdu$$

Possiamo anche valutare la variazione di momento della massa infinitesima espulsa a velocità v:

$$change\ in\ exhaust\ momentum = dm(u - v) - dm\ u = -dm\ v$$

Quindi la variazione totale del momento del sistema è:

$$change\ in\ system\ momentum = Mdu - dm\ v$$

Andiamo ora a considerare le forze agenti sul razzo, fatta eccezione per la resistenza. La forza totale viene calcolata come somma della forza gravitazionale e della forza di pressione nella direzione del moto:

$$F_{TOT} = (p - p_0)A - Mgcos(a)$$

Dove g è la costante gravitazionale e a è il flight path angle.

La variazione di momento del sistema è pari all'impulso su di esso, che a sua volta è pari alla forza totale sul sistema per il tempo dt:

$$Mdu - dm\ v = [(p - p_0)A - Mgcos(a)]dt$$

Ignorando il contributo della forza peso possiamo ricavare:

$$Mdu = [(p - p_0)A]dt + dm\ v$$

La massa espulsa dm è però pari a:

$$dm = \dot{m}dt$$

Dove $\dot{m}$ è il mass flow rate. Possiamo riscrivere l'equazione come:

$$Mdu = [(p - p_0)A + \dot{m}v]dt$$

Introduciamo ora la velocità equivalente di uscita dal propulsore:

$$V_{eq} = \frac{(p - p_0)A}{\dot{m}} + v$$

E sostituiamola nell'equazione:

$$Mdu = V_{eq}\dot{m}dt$$

Il termine $\dot{m}dt$ rappresenta la variazione della massa istantanea del razzo, negativa poiché il razzo sta perdendo massa consumando il combustibile:

$$\dot{m}dt = -dM$$

Sostituiamo ancora nell'equazione e otteniamo:

$$Mdu = -V_{eq}dM$$

Da cui:

$$du = -V_{eq}\frac{dM}{M}$$

Integrando:

$$\Delta u = -V_{eq}\ln(M)$$

Gli estremi di integrazione sono la massa iniziale e la massa finale del razzo. La massa istantanea si compone della massa a vuoto del velivolo, costante nel tempo, e della massa di propellente, che diminuirà nel tempo.

$$M(t) = m_e + m_p(t)$$

Inizialmente la massa del velivolo è data da:

$$M_{initial} = m_f = m_e + m_p$$

Terminato il volo sarà:

$$M_{final} = m_e$$

Sostituendo questi valori otterremo:

$$\Delta u = V_{eq} \ln \left(\frac{m_f}{m_e}\right)$$

Questa è denominata equazione ideale del razzo. Possiamo riscriverla in diverse forme. Definiamo come mass ratio il seguente rapporto:

$$MR = \frac{m_f}{m_e}$$

E sostituiamo nell'equazione:

$$\Delta u = V_{eq} \ln (MR)$$

V_{eq} è correlato all'impulso specifico I_{sp}:

$$V_{eq} = I_{sp} g_0$$

Dove g_0 è l'accelerazione gravitazionale. Sostituendo:

$$\Delta u = I_{sp} g_0 \ln (MR)$$

Se abbiamo un Δu desiderato, possiamo invertire l'equazione per ottenere la massa di propellente richiesta:

$$MR = e^{\Delta u / I_{sp} g_0}$$

[credits: [33]]

3. TEORIA DELL'OTTIMIZZAZIONE

3.1 INTRODUZIONE

L'ottimizzazione per definizione è il raggiungimento del risultato più vantaggioso possibile nei termini dati o in relazione a un determinato fine. L'ottimizzazione viene ormai utilizzata in moltissimi campi e prevede l'utilizzo di algoritmi programmati per la ricerca del minimo (o del massimo) di una funzione, detta funzione obiettivo, collocata nel quadro di un problema adeguatamente modellizzato. Tali tecniche possono essere utilizzate per svariate tipologie di problema, ad esempio l'ottimizzazione del personale in un'azienda, l'ottimizzazione delle prestazioni di un macchinario, l'ottimizzazione del tempo nello svolgimento di una determinata attività e così via. Nel nostro caso andiamo ad utilizzare tali metodologie per la ricerca di una traiettoria ottimale.

Definiamo in maniera generica un problema di ottimizzazione. Definiamo il vettore delle variabili decisionali:

$$\bar{x} \in \mathbb{R}^n$$

$$\bar{x} = \{x_j\}, \qquad con\ j = 1,2, \dots, n$$

La funzione obiettivo:

$$f: \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$$

L'insieme base, logico o fisico:

$$X \subseteq \mathbb{R}^n$$

Le funzioni constraint, che definiscono i vincoli su x:

$$g_i: \mathbb{R}^n \to \mathbb{R}$$

$$g_i(\bar{x}) \geq b_i\,, \qquad con\ i \in I$$

$$g_i(\bar{x}) = d_i\,, \qquad con\ i \in E$$

In cui $b_i \in \mathbb{R}$ e $d_i \in \mathbb{R}$.

Il problema generico di ottimizzazione è quindi:

$$\underset{\bar{x}}{minimize}\ f(\bar{x}),$$

$$\begin{gathered} g_i(\bar{x}) \geq b_i\,, \ \ i \in I, \\ g_i(\bar{x}) = d_i\,, \ \ i \in E, \\ \bar{x} \in X \end{gathered}$$

Possiamo classificare la tipologia di problema di ottimizzazione in funzione della tipologia di f, g_i e X. Avremo:

Linear programming (LP): Linearità delle funzioni f e g_i, $i \in I \cup E$.

Non linear programming (NLP): Non linearità delle funzioni f e g_i, $i \in I \cup E$.

Continuous optimization: f e g_i, $i \in I \cup E$ sono continue su un insieme aperto contenente X; X è chiuso e convesso.

Integer programming (IP): $X \subseteq \{0,1\}^n$ (binario) o $X \subseteq \mathbb{Z}^n$ (intero)

Unconstrained optimization: $I \cup E \coloneqq \emptyset; X \coloneqq \mathbb{R}^n$

Constrained optimization: $I \cup E \neq \emptyset$ and/or $X \subset \mathbb{R}^n$

Differentiable optimization: f e g_i, $i \in I \cup E$ sono funzioni differenziabili su X

Non-differentiable optimization: almeno una fra f e g_i, $i \in I \cup E$ non è una funzione differenziabile

Convex programming (CP): f è convessa; g_i, $i \in I$ sono concave; g_i, $i \in E$ sono concave; X è chiuso e convesso

Non-convex programming: il complemento di quanto sopra

Optimal Control: la variabile x è una funzione, tipicamente continua, che dipende da un insieme finito di variabili di controllo. Il problema verrà discretizzato con un numero finito di variabili.

Un'altra grande classificazione che faremo è quella fra metodi diretti e metodi indiretti.

Metodi diretti: Vanno a costruire una sequenza di punti $x_1, x_2, \dots, x^*$ tale per cui viene minimizzata la funzione obiettivo e tale per cui si ha generalmente $F(x_1) > F(x_2) > \cdots > F(x^*)$. La sequenza viene costruita usando solo valori di $F(x)$. Questi metodi hanno il vantaggio di risultare computazionalmente poco costosi e allo stesso tempo accurati. La loro applicazione può però risultare complessa per alcuni vincoli o problemi.

Metodi indiretti: Vanno a calcolare il gradiente della funzione obiettivo e tramite un algoritmo di ricerca delle radici, calcolano le variabili in modo da ottenere un gradiente nullo. Tramite tali metodologie è possibile calcolare la

soluzione del problema di ottimizzazione senza mai computare la funzione obiettivo, ma il gradiente della stessa. Si costruisce la sequenza di punti tali per cui si hanno valori decrescenti della norma del gradiente. I vantaggi dell'utilizzo di questi metodi sono l'elevata robustezza dell'algoritmo, la semplificazione della trattazione dei vincoli, l'idoneità nel trattare problemi complessi e l'esistenza di molti risultati teorici che assicurano la convergenza dei metodi. Hanno però anche degli svantaggi, come la poca accuratezza, molto influenzata dalla scelta dei valori di partenza, e l'elevato costo computazionale.

In generale, tutti i metodi di ottimizzazione producono una successione di approssimazioni $x_0, x_1, x_2, \dots, x_f$ che sotto opportune ipotesi convergono alla soluzione ottima del problema. Possono però far ottenere soluzioni non ottimali se la scelta dei valori iniziali non è adeguatamente accurata.

Si va solo a modificare iterativamente la soluzione con lo scopo di ridurre gli errori sulle condizioni di ottimo.

[credits: [5],[6],[7],[8]]

3.2 METODI PER L'OTTIMIZZAZIONE DI TRAIETTORIE

3.2.1 FORMULAZIONE MATEMATICA GENERALE

La dinamica del sistema a cui vogliamo applicare la teoria dell'ottimizzazione viene generalmente espressa come:

$$\dot{\bar{y}} = f(\bar{y}(t), \bar{u}(t), \bar{p}, t)$$

$\bar{y}$ rappresenta il vettore di stato a n_y componenti, $\bar{u}$ il vettore dei controlli a n_u componenti, t rappresenta il tempo e infine $\bar{p}$ i possibili parametri non dipendenti dal tempo.

Le condizioni a contorno al tempo iniziale t_0 sono:

$$\psi_{0l} \leq \psi(\bar{y}(t_0), \bar{u}(t_0), \bar{p}, t_0) \leq \psi_{0u}$$

Le condizioni a contorno al tempo finale t_f sono:

$$\psi_{fl} \leq \psi(\bar{y}(t_f), \bar{u}(t_f), \bar{p}, t_f) \leq \psi_{fu}$$

Eventualmente, indichiamo con ψ_0 il valore iniziale della funzione ψ e con ψ_f il suo valore finale.

Inoltre devono anche essere rispettati i bounds sulle variabili di stato:

$$\bar{y}_l \leq \bar{y}(t) \leq \bar{y}_u$$

E sulle variabili di controllo:

$$\bar{u}_l \leq \bar{u}(t) \leq \bar{u}_u$$

Oltre che le equazioni algebriche legate ai vincoli:

$$\bar{g}_l \leq \bar{g}(\bar{y}(t), \bar{u}(t), \bar{p}, t) \leq \bar{g}_u$$

In cui $\bar{g}$, di dimensioni n_g, racchiude tutti i vincoli.

Lo scopo della trattazione sarà di trovare le funzioni a valori vettoriali $\bar{y}^*$, $\bar{u}^*$ e il vettore $\bar{p}$, che trovino il minimo (o il massimo se viene richiesto un problema di massimo) della funzione obiettivo così definita, secondo la formulazione di Bolza:

$$J(\bar{y}) = \varphi\big(\bar{y}(t_0), \bar{y}(t_f), t_0, t_f\big) + \int_{t_0}^{t_f} \Phi(\bar{y}, \dot{\bar{y}}, t)dt$$

Secondo la formulazione di Meyer, con $\Phi = 0$:

$$J(\bar{y}) = \varphi\big(\bar{y}(t_0), \bar{y}(t_f), t_0, t_f\big)$$

Secondo la formulazione di Lagrange, con $\varphi = 0$:

$$J(\bar{y}) = \int_{t_0}^{t_f} \Phi(\bar{y}, \dot{\bar{y}}, t)dt$$

[credits: [5],[6],[7],[8]]

3.2.2 NON LINEAR PROGRAMMING (NLP)

Tra i metodi numerici più utilizzati, annoveriamo i metodi di Non Linear Programming (NLP). I metodi evidenziati permettono la ricerca delle variabili finite, tali che consentano di trovare il punto di ottimo di una determinata funzione, sotto determinati vincoli imposti. Esempi della suddetta famiglia di metodi sono il Linear Programming (LP), il Quadrating Programming (QP) e il metodo dei minimi quadrati.

Il fondamento degli schemi di risoluzione che fanno parte di questo sottoinsieme di metodi, viene dagli studi di Newton per la risoluzione delle equazioni non lineari. Supponiamo di avere un'equazione non lineare del tipo:

$$c(x) = 0$$

Si approssima la funzione $c(x)$ con un'espansione di Taylor troncata al secondo ordine rispetto ad una soluzione iniziale ipotizzata x_k:

$$c(x) = c(x_k) + c'(x_k)(x - x_k)$$

Geometricamente questo equivale a dire che la funzione è stata approssimata localmente con una retta passante per il punto $(x_k, c(x_k))$.

È facile dunque trovare il punto x_{k+1}, tale che $c(x_k + 1) \approx 0$, trovando la radice di tale retta:

$$x_{k+1} = x_k - \frac{c(x_k)}{c'(x_k)}$$

Ripetendo il procedimento iterativamente si giunge alla radice x^* della funzione di partenza.

Ipotizziamo ora di ricercare la soluzione di un problema di minimizzazione della funzione obiettivo *J(x)*.

Andiamo a sviluppare la funzione con una serie di Taylor al terzo ordine:

$$J(x) = J(x_k) + J'(x_k)(x - x_k) + \frac{1}{2}J''(x_k)(x - x_k)^2 \qquad (*)$$

Per trovare il punto di minimo occorre che poniamo pari a zero la derivata della funzione $J(x)$:

$$J'(x) = \frac{dJ(x)}{dx} = J'(x_k) + J''(x_k)(x - x_k) = 0$$

Utilizzando quindi il metodo di Newton per risolvere tale equazione non lineare, possiamo trovare il punto x_{k+1} come:

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$

Definiamo con p l'incremento da apportare alla radice per ottenere l'iterazione successiva:

$$p = -\frac{J'(x_k)}{J''(x_k)} \qquad (**)$$

Dunque:

$$x_{k+1} = x_k + p$$

Il limite della successione che sto costruendo è un punto x^* tale che $J'(x^*) = 0$.

La soluzione trovata potrebbe rappresentare un punto di minimo o di massimo. Per capire se ci troviamo in presenza di un minimo o di un massimo della funzione è necessario studiare il segno della derivata seconda della funzione.

Avremo trovato un minimo se:

$$J'(x^*) = 0$$

$$J''(x^*) \geq 0$$

Torniamo ora ad un sistema multidimensionale e ipotizziamo un'analisi di ottimo non vincolata.

Avremo:

$$\bar{x} = \{x_1, \dots, x_k, \dots, x_n\}^T$$

$$J(\bar{x}) = J(x_1, \dots, x_k, \dots, x_n)$$

Indichiamo con $\tilde{\bar{x}}$ l'approssimazione iniziale da cui avranno inizio le iterazioni. Indichiamo con $\bar{p}$ il vettore che denota il passo:

$$\bar{p} = \bar{x} - \tilde{\bar{x}}$$

Indichiamo con $\bar{g}$ il gradiente della funzione obiettivo:

$$\bar{g}(\bar{x}) = \nabla J(\bar{x}) = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \vdots \\ \frac{\partial J}{\partial x_n} \end{bmatrix}$$

Indichiamo con *H* la matrice Hessiana della funzione obiettivo, così definita:

$$H = \nabla^2 J(\bar{x}) = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_n \partial x_1} & \frac{\partial^2 J}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_n^2} \end{bmatrix}$$

Possiamo scrivere:

$$J(\bar{x}) = J(\tilde{\bar{x}}) + \bar{g}^T \bar{p} + \frac{1}{2} \bar{p}^T H \bar{p}$$

Analogo multidimensionale della funzione (*).

Indichiamo con $\bar{x}^*$ la condizione di ottimo. Varranno le seguenti relazioni:

$$\bar{g}(\bar{x}^*) = 0$$

$$\bar{p}^T H(\bar{x}^*)\bar{p} \geq 0$$

La condizione $\bar{g}(\bar{x}^*) = 0$ potrà essere esaminata utilizzando il metodo di Newton. Avremo perciò:

$$\bar{g} + H\bar{p} = 0$$

Analogo multidimensionale della funzione (**).

Potremo calcolare dunque il passo come soluzione del sistema lineare:

$$H\bar{p} = -\bar{g}$$

Da notare che le derivate potranno essere calcolate tramite approssimazione numerica, in particolar modo usando il metodo delle differenze finite. Applicando la tecnica delle differenze finite alle derivate seconde, si ottengono i metodi di Quasi-Newton. Al posto della matrice Hessiana $\nabla^2 J(\bar{x})$ si userà una sua approssimazione, data da una matrice *B* definita in modo opportuno.

3.2.2.1 Equality Constraints

Immaginiamo un problema di ottimizzazione in cui ci troviamo a dover cercare le n variabili $\bar{x}$ che minimizzino la funzione obiettivo $J(\bar{x})$ e soddisfano gli m (con $m<n$) equality constraints:

$$\bar{c}(\bar{x}) = 0$$

Indichiamo con *G* la matrice Jacobiana dei vincoli:

$$G(\bar{x}) = \frac{\partial \bar{c}}{\partial \bar{x}} = \begin{bmatrix} \frac{\partial c_1}{\partial x_1} & \frac{\partial c_1}{\partial x_2} & \cdots & \frac{\partial c_1}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial c_n}{\partial x_1} & \frac{\partial c_n}{\partial x_2} & \cdots & \frac{\partial c_n}{\partial x_n} \end{bmatrix}$$

Introduciamo il Lagrangiano, una funzione scalare di $\bar{x}$ e degli m moltiplicatori di Lagrange $\bar{\lambda}$:

$$L(\bar{x}, \bar{\lambda}) = J(\bar{x}) - \bar{\lambda}^T \bar{c}(\bar{x})$$

Il teorema di Karush-Kuhn-Tucker afferma che se $(\bar{x}^*, \bar{\lambda}^*)$ è un punto di sella per il Lagrangiano, allora $\bar{x}^*$ è soluzione del problema di ottimizzazione.

Per far sì che la soluzione $(\bar{x}^*, \bar{\lambda}^*)$ sia una condizione di ottimo è necessario ricercare un punto stazionario del Lagrangiano definito come segue:

$$\nabla_x L(\bar{x}, \bar{\lambda}) = \bar{g}(\bar{x}) - G^T(\bar{x})\bar{\lambda} = 0$$
$$\nabla_\lambda L(\bar{x}, \bar{\lambda}) = -\bar{c}(\bar{x}) = 0$$

Per trovare la soluzione che soddisfa tali condizioni è possibile utilizzare il metodo di Newton. Costruiamo il sistema:

$$\begin{bmatrix} H_L & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} \bar{p} \\ -\bar{\lambda} \end{bmatrix} = \begin{bmatrix} -\bar{g} \\ -\bar{c} \end{bmatrix}$$

Tale sistema lineare prende il nome di sistema Kuhn-Tucker (KT) o sistema Karush-Kuhn-Tucker.

Abbiamo bisogno di calcolare l'hessiano del Lagrangiano:

$$H_L = \nabla_x^2 J - \sum_{i=1}^{m} \lambda_i \nabla_x^2 c_i$$

Notiamo che $\bar{p}$ è anche valutabile andando a minimizzare la seguente forma quadratica:

$$\bar{g}^T \bar{p} + \frac{1}{2} \bar{p}^T H_L \bar{p}$$

Soggetta ai vincoli lineari:

$$G\bar{p} = -\bar{c}$$

Ci riferiamo a tale problema come un problema QP (Quadratic Programming).

3.2.2.2 Inequality Constraints

Il problema della sezione precedente viene generalizzato andando ad imporre degli inequality constraints. Immaginiamo un problema di ottimizzazione in cui ci troviamo a dover cercare le n variabili $\bar{x}$ che minimizzino la funzione obiettivo $J(\bar{x})$ e soddisfano gli m inequality constraints:

$$\bar{c}(\bar{x}) \geq 0$$

Questa volta non imponiamo $m<n$, ma anzi, m potrebbe essere maggiore di n. Potremo dividere tali constraints in due categorie:

- Inactive constraints, per cui:
$$\bar{c}(\bar{x}) > 0$$
- Active constraints, per cui:
$$\bar{c}(\bar{x}) = 0$$

Gli active constraints potranno essere sviluppati come equality constraints, andando però ad utilizzare degli opportuni algoritmi di

distinzione fra active e inactive constraints. Per questa tipologia di problemi si fa ampio uso degli algoritmi di quadratic programming (QP) tra cui l'algoritmo SQP (sequential quadratic programming) che andremo ad utilizzare.

[credits: [5],[6],[7],[8],[28]]

3.2.3 OPTIMAL CONTROL

Attraverso tale metodo dovremo andare a risolvere un sistema di equazioni differenziali con date condizioni al contorno. Si vanno quindi a ricercare le condizioni necessarie di ottimo. Si inizia indicando perciò una soluzione iniziale, andando successivamente a determinare l'errore sia sulla soluzione tentativo, sia sulle condizioni a contorno. Si procede così iterativamente fin quando non si trova una soluzione ottimale. Si procede generalmente andando a suddividere il problema in diversi sotto-problemi finiti, risolti andando ad utilizzare il metodo di Newton o Quasi-Newton nelle iterazioni.

Si introduce una versione modificata della funzione obiettivo, che vede al suo interno l'utilizzo dei moltiplicatori di Lagrange μ e λ, associati alle equazioni differenziali delle variabili di stato:

$$J^* = \varphi + \bar{\mu}^T \psi + \int_{t_0}^{t_f} \left[\Phi + \bar{\lambda}^T (\bar{f} - \dot{\bar{x}})\right] dt$$

Nel caso in cui vengano rispettate sia le equazioni di stato che le condizioni a contorno avremo $J^* = J$.

Sviluppando e derivando la funzione J^* otteniamo:

$$\begin{aligned} dJ^* = &\left(\frac{\partial \varphi}{\partial t_f} + \bar{\mu}^T \frac{\partial \psi}{\partial t_f} + \Phi_f + \bar{\lambda}_f^T (\bar{f}_f - \dot{\bar{x}}_f)\right) dt_f + \left(\frac{\partial \varphi}{\partial t_0} + \bar{\mu}^T \frac{\partial \psi}{\partial t_0} + \Phi_0 + \bar{\lambda}_0^T (\bar{f}_0 - \dot{\bar{x}}_0)\right) dt_0 \\ &+ \left(\frac{\partial \varphi}{\partial x_f} + \bar{\mu}^T \frac{\partial \psi}{\partial x_f}\right) dx_f + \left(\frac{\partial \varphi}{\partial x_0} + \bar{\mu}^T \frac{\partial \psi}{\partial x_0}\right) dx_0 \\ &+ \int_{t_0}^{t_f} \left[\frac{\partial (\Phi + \bar{\lambda}^T \bar{f})}{\partial x} \delta x + \frac{\partial (\Phi + \bar{\lambda}^T \bar{f})}{\partial u} \delta u + \bar{\lambda}^T \delta \dot{x}\right] dt \end{aligned}$$

Definiamo l'Hamiltoniano:

$$H = \Phi + \bar{\lambda}^T \bar{f}$$

Lo sostituiamo all'interno della relazione precedente:

$$dJ^* = \left(\frac{\partial \varphi}{\partial t_f} + \bar{\mu}^T \frac{\partial \psi}{\partial t_f} + H_f\right) dt_f + \left(\frac{\partial \varphi}{\partial t_0} + \bar{\mu}^T \frac{\partial \psi}{\partial t_0} - H_0\right) dt_0 + \left(-\bar{\lambda}_f^T + \frac{\partial \varphi}{\partial x_f} + \bar{\mu}^T \frac{\partial \psi}{\partial x_f}\right) dx_f$$
$$+ \left(\bar{\lambda}_f^T + \frac{\partial \varphi}{\partial x_0} + \bar{\mu}^T \frac{\partial \psi}{\partial x_0}\right) dx_0 + \int_{t_0}^{t_f} \left[(\frac{\partial H}{\partial x} + \bar{\lambda}^T)\delta x + \frac{\partial H}{\partial u}\delta u\right] dt$$

Per avere una condizione di ottimo, dovremo imporre che la funzione obiettivo modificata sia stazionaria, il che significa che:

$$dJ^* = 0$$

Da tale relazione ricaviamo le equazioni di Eulero-Lagrange e le equazioni di controllo ottimale:

$$\frac{d\bar{\lambda}}{dt} = -(\frac{\partial H}{\partial x})^T$$

$$(\frac{\partial H}{\partial u})^T = 0$$

Per semplicità di scrittura introduciamo la funzione:

$$\Psi = \varphi + \bar{\mu}^T \psi$$

Definiamo quindi le condizioni di trasversalità:

$$\bar{\lambda}(t_f) = \Psi_x^T|_{t=t_f}$$

$$0 = (\Psi_t + H)|_{t=t_f}$$

Possiamo infine definire il nostro sistema di equazioni differenziali costituito dalle relazioni trovate:

$$\begin{cases} \dot{\bar{x}} = f(\bar{x}(t), \bar{u}(t), t) \\ \frac{d\bar{\lambda}}{dt} = -(\frac{\partial H}{\partial x})^T \\ (\frac{\partial H}{\partial u})^T = 0 \end{cases}$$

Le cui condizioni a contorno sono:

$$\psi_{0l} \leq \psi_0 \leq \psi_{0u}$$

$$\psi_{fl} \leq \psi_f \leq \psi_{fu}$$

$$\bar{\lambda}(t_f) = \Psi_x^T|_{t=t_f}$$

$$0 = (\Psi_t + H)|_{t=t_f}$$

Supponiamo ora di avere degli equality constraints così definiti:

$$\bar{g}(\bar{x}(t), \bar{u}(t), t) = 0$$

Si determina la matrice g_u come:

$$g_u = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial u_1} & \cdots & \frac{\partial g_n}{\partial u_n} \end{bmatrix}$$

E se ne calcola il rango. In caso di rango massimo il sistema di equazioni differenziali avrà indice 1 e l'Hemiltoniano diventa:

$$H = \Phi + \bar{\lambda}^T \bar{f} + \bar{\mu}^T \bar{g}$$

Se non abbiamo rango massimo dobbiamo derivare la $\bar{g}$:

$$0 = \bar{g}_x \dot{\bar{x}} + \bar{g}_u \dot{\bar{u}} + \bar{g}_t = \bar{g}'$$

Con le stesse modalità definite prima andremo a calcolare la matrice $\bar{g}'_u$ e ne determineremo il rango. In caso di rango massimo il sistema di equazioni differenziali avrà indice 2, altrimenti ripeteremo i passaggi appena eseguiti. È vantaggioso trovare un sistema di indice che sia il più possibile basso in quanto un indice elevato causa una maggiore probabilità di incorrere in errori numerici.

Supponiamo adesso di avere degli inequality constraints, così definiti:

$$\bar{g}(\bar{x}(t), \bar{u}(t), t) \geq 0$$

In questo caso andremo a riprendere la differenza fra active constraints e inactive constraints. Gli active constraints sono definiti come quegli inequality constraints per cui vale:

$$\bar{g}(\bar{x}(t), \bar{u}(t), t) = 0$$

Invece gli inactive constraints sono definiti come quegli inequality constraints per cui vale:

$$\bar{g}(\bar{x}(t), \bar{u}(t), t) > 0$$

Gli archi di traiettoria in cui il vincolo risulta attivo saranno vincolati, viceversa per gli archi in cui invece il vincolo risulta inattivo. Gli archi vincolati verranno analizzati con la trattazione vista per gli equality constraints. Il problema però sorge

nel momento in cui risulta sconosciuta a priori la quantità di sotto archi vincolati e l'ubicazione dei punti di giunzione tra sotto archi vincolati e sotto archi non vincolati.

[credits: [5],[6],[7],[8]]

3.3 GRADIENT BASED METHOD

In questa sezione andremo ad analizzare i metodi che si basano sulla tecnica della discesa del gradiente. Possiamo utilizzare un semplice esempio per permettere una migliore comprensione del metodo. Immaginiamo che la funzione sia una montagna. Il minimo di questa funzione sarà rappresentato dalla valle. Non sappiamo quale sia la direzione per raggiungere la valle, possiamo solo ricavare informazioni dall'ambiente direttamente intorno a noi. Intuitivamente cercheremo di fare un passo avanti lì dove il dislivello sembra più ripido, così da arrivare prima a destinazione. Facendo ogni passo con questa metodologia arriveremo al punto in cui non ci sarà più dislivello, ma avremo raggiunto una zona piana. Quella sarà il nostro arrivo. Possiamo applicare lo stesso ragionamento al metodo del gradiente. I passi fondamentali di tale metodo sono:

- Search direction
- Step size
- Convergence check

Il primo passo può essere visto come il momento in cui scegliamo la direzione di discesa. Prendendo in considerazione la funzione da minimizzare, sarà la derivata ad indicare la direzione di ricerca, in quanto ci darà informazione sulla crescita/decrescita della funzione. Ritorniamo alla nostra montagna. Una volta deciso in quale direzione andare, dobbiamo decidere per quanto tempo continuare in quella direzione. Il terreno potrebbe infatti cambiare mentre andiamo avanti e la direzione selezionata in precedenza potrebbe rivelarsi non più la corretta. Dal punto di vista matematico si procede perciò a definire uno step size che mi indichi la dimensione del passo da eseguire. Eseguendo questi due passaggi arriveremo ad una condizione di "pianura", la nostra condizione di minimo. Potremmo essere di fronte però ad un minimo locale e non al minimo globale ricercato. È per questo che dobbiamo eseguire il terzo step, per assicurarci di aver trovato un minimo globale. Diamo adesso una trattazione più rigorosa.

La condizione necessaria perché questo metodo possa essere applicato è: data una funzione $F(x_1, x_2, \dots, x_n)$ essa deve essere continua e derivabile nell'intorno di un punto $\bar{a}$. Il gradiente della funzione nel punto $\bar{a}$ sarà:

$$-\nabla F(\bar{a})$$

Iniziando da un valore tentativo $\bar{x}$ procederemo iterativamente secondo la procedura descritta:

$$\bar{x}_{n+1} = \bar{x}_n - \gamma_n \nabla F(\bar{x}_n)$$

Lo step size γ è una quantità reale, positiva, scelta in modo che, se sufficientemente piccola, la quantità:

$$\bar{a}_{n+1} = \bar{a}_n - \gamma_n \nabla F(\bar{a}_n)$$

Farà sì che la funzione assuma un valore sempre più piccolo. La velocità di convergenza e l'affidabilità del metodo dipendono dalla tipologia di funzione da ottimizzare. La declinazione di questa metodologia cambierà in base alla tipologia di funzione e alla tipologia di problema di ottimizzazione che dovremo affrontare. Potrà essere quindi applicata ai metodi visti precedentemente nel capitolo. In particolare la vedremo applicata al NLP. Possiamo introdurre una formulazione generale del metodo del gradiente discendente applicato ad un problema non lineare.

Dato il sistema iniziale, posso scrivere la funzione associata $G(\bar{x})$ che permetta di riscrivere la funzione obiettivo come:

$$F(\bar{x}) = \frac{1}{2} G(\bar{x})^T G(\bar{x})$$

Possiamo dare inizio alle iterazioni partendo dal valore tentativo $\bar{x}_n$:

$$\bar{x}_{n+1} = \bar{x}_n - \gamma \nabla F(\bar{x}_n) = \bar{x}_n - \gamma J_G(\bar{x}_n)^T G(\bar{x}_n)$$

In cui $J_G(\bar{x})$ è la matrice Jacobiana.

[credits: [30],[31],[32]]

3.4 SEQUENTIAL QUADRATIC PROGRAMMING (SQP)

Uno dei migliori metodi per affrontare problemi di ottimizzazione di traiettorie è il Sequential Quadratic Programming (SQP). Tale metodo va a suddividere il problema NLP in diversi sotto-problemi quadratici da risolvere in successione. In base alla selezione dei sotto-problemi quadratici il metodo potrebbe ricordare la risoluzione del metodo di Newton e potrebbe essere visto come una sua estensione. Il problema però si va a complicare rispetto al metodo di Newton anche a causa dell'introduzione dei vincoli.

Il metodo prevede innanzitutto l'approssimazione quadratica della funzione obiettivo e l'approssimazione lineare dei vincoli. È implicita l'approssimazione lineare del gradiente. Combinando la soluzione del metodo di Newton con l'utilizzo delle equazioni di Kuhn-Tucker viste nei capitoli precedenti, vengono avviate una serie di iterazioni che hanno come risultato finale il raggiungimento di una soluzione ottimale. A tal proposito bisogna però fare attenzione al momento dell'inserimento delle initial guesses. Infatti una loro sbagliata selezione potrebbe portarci ad ottenere un minimo locale invece che il minimo globale che cercavamo.

Un problema quadratico utilizza la seguente relazione per stimare le variabili $\bar{x}$ ad ogni iterazione:

$$\tilde{\bar{x}} = \bar{x} + \alpha\bar{p}$$

Definiamo $\bar{p}$ come la search direction e α come lo step length, solitamente impostato a 1. La search direction $\bar{p}$ dell'iterazione k-esima è computata nella seguente maniera:

$$p_k = x_{k+1} - x_k$$

Definiamo con $\bar{q}$ il vettore differenza tra i gradienti della Lagrangiana. All'iterazione k-esima è calcolato come segue:

$$q_k = \nabla_x L_{k+1} - \nabla_x L_k = \left(\nabla J(x_{k+1}) + \sum_{i=1}^{m} \lambda_i \nabla c_i(x_{k+1})\right) - \left(\nabla J(x_k) + \sum_{i=1}^{m} \lambda_i \nabla c_i(x_k)\right)$$

Con H indichiamo una matrice simmetrica, di dimensioni NxN, definita positiva, approssimazione dell'Hessiana. Verrà calcolata come segue:

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{H_k p_k p_k^T H_k^T}{p_k^T H_k p_k}$$

Il requisito riguardo l'ottenere una matrice Hessiana definita positiva dopo ogni iterazione è soddisfatto mantenendo $q_k^T p_k > 0$ dopo ogni iterazione.

La funzione di merito, da minimizzare simultaneamente alla funzione obiettivo, è:

$$M(x) = J(x) + \sum_{i=1}^{m_e} r_i c_i(x) + \sum_{i=m_e+1}^{m} r_i \max\,[0, c_i(x)]$$

In cui abbiamo indicato con m il numero totale di vincoli, mentre con m_e solo la somma tra gli equality constraints e gli active inequality constraints. Definiamo in ultima battuta il parametro di penalità r, utilizzato per pesare la correzione dell'errore sui diversi vincoli in modo da scegliere correttamente lo step length:

$$r_i = (r_{k+1})_i = \max_i \left[\lambda_i, \frac{(r_k)_i + \lambda_i}{2}\right], \qquad i = 1, \dots, m$$

[credits: [5],[6],[7],[8]]

3.5 STATE TRANSITION MATRIX

Risulta utile anche definire la State Transition Matrix (STM), utilizzata per computare il gradiente degli equality constraints.

Dato il sistema dinamico:

$$\dot{p} = F(p)$$

Ipotizziamo che il sistema sia inizialmente in p_0 e che dopo un certo tempo di propagazione T arrivi al punto p_f. Perturbiamo il punto iniziale:

$$p_0 \to p_0 + \delta p_0$$

Il punto di arrivo sarà influenzato da tale perturbazione:

$$p_f \to p_f + \delta p_f$$

Se δp_0 è sufficientemente piccolo, ci si aspetta che esista una relazione lineare tra δp_0 e δp_f, del tipo:

$$\delta p_f \sim A \delta p_0$$

Con A matrice indefinita.

Torniamo ora alle equazioni del moto. Abbiamo una traiettoria iniziale definita $\gamma(t)$ con condizioni iniziale e finale:

$$\gamma(0) = p_0$$
$$\gamma(T) = p_f$$

Definiamo:

$$p(t) = \gamma(t) + \delta p(t)$$

In cui $\delta p(t) \ll \gamma(t)$. Siamo interessati alla dinamica di $\delta p(t)$. Sviluppiamo dunque le equazioni del moto:

$$\frac{d}{dt}(\gamma(t) + \delta p) = F(\gamma(t) + \delta p)$$
$$\frac{d}{dt}\gamma(t) + \frac{d}{dt}\delta p + F\big(\gamma(t)\big) + \frac{\partial F}{\partial p}|_{\gamma(t)} \delta p$$

Da cui:

$$\frac{d}{dt}\delta p = \frac{\partial F}{\partial p}|_{\gamma(t)} \delta p$$

Quest'ultima equazione è della forma:

$$\dot{\boldsymbol{x}} = A(t)\boldsymbol{x}$$

Possiamo quindi ricavare dalla teoria delle equazioni differenziali ordinarie (ODE) la soluzione generale, scritta come segue:

$$\delta p(t) = M(t)\delta p_0$$

In cui $M(t)$ è una matrice dipendente dal tempo non definita e δp_0 è la perturbazione iniziale. Andando a sostituire la soluzione nell'equazione differenziale appena sopra otteniamo:

$$\frac{d}{dt}(M(t)\delta p_0) = \frac{\partial F}{\partial p}|_{\gamma(t)} M(t)\delta p_0$$

Quest'ultima equazione potrà sempre essere soddisfatta se:

$$\frac{d}{dt}M(t) = \frac{\partial F}{\partial p}|_{\gamma(t)} M(t)$$

Per ogni istante, con condizioni iniziali:

$$M(0) = I$$

Con *I* la matrice identità. Risolvendo questo sistema di equazioni per l'istante *T* otteniamo la State Transition Matrix, che lega δp_0 e δp_f con la seguente relazione:

$$\delta p_f = M(T)\delta p_0$$

Si potrebbe descrivere la dinamica anche utilizzando l'operatore di flusso $\Phi_T(\cdot)$, che ha lo scopo di relazionare un punto p_0 alla sua immagine dopo un intervallo di propagazione *T*:

$$\Phi_T(p_0) = p_f$$

Utilizzando tale scrittura la STM è data da:

$$M(T) = \frac{\partial}{\partial p_0}[\Phi_T(p_0)] = \frac{\partial p_f}{\partial p_0}$$

Nel momento in cui andiamo a modificare le condizioni iniziali come già visto e il tempo di integrazione $T \to T + \delta T$, possiamo ricavare la modifica delle condizioni finali con:

$$\delta p_f = M(T)\delta p_0 + F(p_f)\delta T$$

La State Transition Matrix è una matrice *6x6* costituita da 4 sottomatrici *3x3* così suddivise:

$$M(T) = \begin{bmatrix} M^{rr} & M^{rv} \\ M^{vr} & M^{vv} \end{bmatrix}$$

Ciò consente di scrivere le perturbazioni della posizione e velocità finali Δr e Δv come funzione delle perturbazioni della posizione e velocità iniziali Δr_0 e Δv_0:

$$\Delta r = M^{rr}\Delta r_0 + M^{rv}\Delta v_0$$
$$\Delta v = M^{vr}\Delta r_0 + M^{vv}\Delta v_0$$

[credits: [10]]

4. SVILUPPO

4.1 DEFINIZIONE DEL PROBLEMA

Un problema tipico dell'analisi di missione è la ricerca di traiettorie di trasferimento tra strutture orbitali. Solitamente uno dei parametri da minimizzare nella definizione di tali traiettorie è il ΔV propulsivo.

Lo scopo di questa tesi è proprio quello di portare avanti una valutazione preliminare della traiettoria di trasferimento da Near-Rectilinear Halo Orbit a Low Lunar Orbit, ottimizzandone il ΔV.

Prendiamo dunque in considerazione il sistema Terra-Luna-Lander. La dinamica del sistema verrà descritta dal CR3BP.

Riportiamo alcune grandezze note del sistema Terra-Luna:

Distanza Terra-Luna	a	*384400 Km*
Standard Gravitational Parameter (Terra)	μ_E	*398600.4415* Km^3/s^2
Standard Gravitational Parameter (Luna)	μ_M	*4902.8005821478* Km^3/s^2
Periodo orbitale siderale	T	*2357383.88 s*
Velocità angolare sistema Terra-Luna	ω	$2.665 \cdot 10^{-6}\ rad/s$

Tabella 3 - Grandezze note del Sistema Terra-Luna

Per poter effettuare un'analisi adimensionalizzata è necessario scalare le grandezze coinvolte. Usualmente si scala in modo da ottenere:

$$a = 1$$
$$\mu_E + \mu_M = 1$$
$$\frac{T}{2\pi} = 1$$

I fattori di conversione saranno quindi:

Time scale (s)	*375190.259*
Length scale (Km)	*384400*
Velocity scale (Km/s)	*1.02454685531*

Tabella 4 - Fattori di scala

Al fine di ottenere una trattazione indipendente dal parametro tempo, il sistema di riferimento considerato è non inerziale, con origine nel centro di massa del sistema. Il sistema ruoterà con velocità angolare ω. L'asse x si posizionerà lungo la congiungente Terra-Luna e sarà tale da avere la Luna nella direzione delle x positive e la Terra nella direzione delle x negative. L'asse z sarà determinato dalla direzione del vettore velocità angolare. L'asse y completerà la terna destrorsa. Per la successiva trattazione del CR3BP vedere il capitolo 2.

Ora prendiamo in esame le strutture orbitali di partenza e arrivo.

La NRHO di partenza sarà la stessa selezionata per la Lunar Orbital Platform-Gateway, ovvero una Southern Near-Rectilinear Halo Orbit di L_2. Di seguito viene riportato il vettore di stato che ne identifica un punto di riferimento:

$$p_{NRHO} = [396252.18309\ Km, 0\ Km, -72100.87794\ Km, 0\ {}^{Km}/_{S}, -0.12466\ {}^{Km}/_{S}, 0\ {}^{Km}/_{S}]$$

E le sue caratteristiche:

Informazione	***Dati***
Orbita	*L2 Southern NRHO 9:2 Resonance*
Periodo	*610695,3455 s*
Periselene	*3000 Km*
Aposelene	*70000 Km*

Tabella 5 - Caratteristiche della L2 Southern Near-Rectilinear Halo Orbit [credits: [34]]

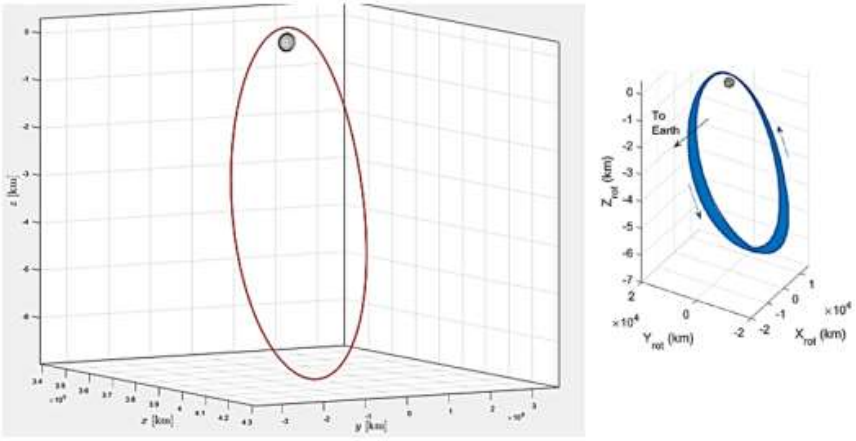

Figura 22 – NRHO of Departure [credits: [34]]

L'orbita di arrivo sarà invece una Low Lunar Orbit circolare, polare, posta a circa *100 Km* di altitudine.

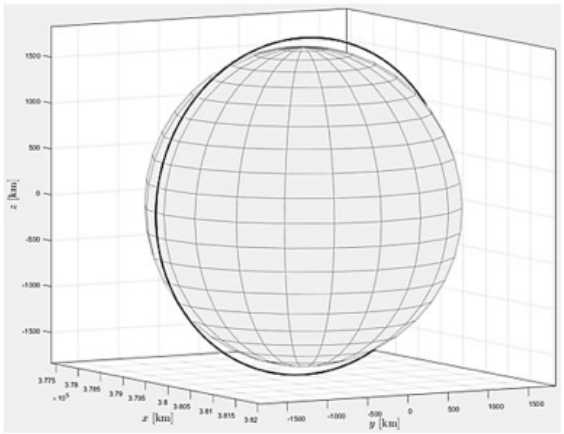

Figura 23 - Target LLO [credits: [34]]

Andiamo ora a definire il problema di ottimo che dovremo risolvere.
Prima di tutto si procede alla parametrizzazione del problema:

- Φ_A : Parametrizzazione del punto di partenza sulla struttura orbitale iniziale
- Φ_B : Parametrizzazione del punto di arrivo sulla struttura orbitale finale
- ToF : Time of flight
- $\Delta\bar{v}_k,\ con\ k = 1{,}2$: manovre impulsive di partenza (*k=1*) e di arrivo (*k=2*)

Una parametrizzazione possibile è il tempo di propagazione da un punto di riferimento. Quindi possiamo scrivere:

$$\Phi_A = \{\vec{p}_A^{\circ}, T_A\}$$

In cui T_A rappresenta il tempo di propagazione e $\vec{p}_A^{\circ}$ il punto di riferimento riguardanti la struttura orbitale di partenza. Il punto di arrivo viene invece parametrizzato attraverso i valori di ascensione retta del nodo ascendente (RAAN) e true anomaly:

$$\Phi_B = \{long, inc\}$$

In cui $long$ rappresenta la RAAN e inc rappresenta la true anomaly.

La funzione obiettivo sarà il Δv totale:

$$\Delta v = \sum_k |\Delta\bar{v}_k|$$

Individuiamo due equality constraints sulla traiettoria: bisogna raggiungere il punto target sull'orbita finale e la velocità raggiunta deve corrispondere alla velocità orbitale del punto di arrivo:

- $E_r = \bar{r}_f - \bar{r}_B$: Errore di posizione al punto di arrivo
- $E_v = \bar{v}_f - \bar{v}_B$: Errore di velocità al punto di arrivo

Individuiamo inoltre un inequality constraint sul Time of flight:

- $ToF \leq T_{max}$

La variabile $\bar{x}$ sarà libera di variare entro alcuni valori limite:

- $lb_i \leq x_i \leq ub_i,\ con\ i = 1, \dots, N$ con N pari alla dimensione del vettore di variabili $\bar{x}$

Possiamo ora scrivere formalmente il nostro problema di ottimizzazione:

$$\min_x \Delta v(x)$$
$$Subject\ to:$$
$$E_r(x) = 0$$
$$E_v(x) = 0$$
$$ToF \leq T_{max}$$
$$\overline{lb} \leq \bar{x} \leq \overline{ub}$$

Dove $\bar{x} = [T_A, long, inc, ToF, \Delta\bar{v}_k]$.

Per la soluzione di questo problema è necessario applicare un algoritmo di risoluzione non lineare. Questo tipo di problema è molto sensibile alla scelta delle variabili iniziali. A causa di questa sensibilità è preferibile fornire al solutore anche il gradiente della funzione obiettivo e dei vincoli. Il gradiente della funzione obiettivo è:

$$\delta[\Delta v] = \sum_k \frac{1}{|\Delta\bar{v}_k|} \Delta\bar{v}_k \delta(\Delta\bar{v}_k)$$

Il gradiente dei vincoli deve essere calcolato rispetto a $T_A, long, inc, ToF, \Delta\bar{v}_k$. Per farlo ci avvaliamo dell'uso della State Transition Matrix (M) definita nei capitoli precedenti. Otteniamo:

$$\begin{pmatrix} \partial E_r \\ \partial E_v \end{pmatrix} = \begin{pmatrix} M_0^{rr} & M_0^{rv} \\ M_0^{vr} & M_0^{vv} \end{pmatrix} \begin{pmatrix} \bar{v}_A \\ \bar{g}_A \end{pmatrix} \partial T_A$$

$$\begin{pmatrix} \partial E_r \\ \partial E_v \end{pmatrix} = -\begin{pmatrix} \partial \bar{r}_B / \partial long \\ \partial \bar{v}_B / \partial long \end{pmatrix} \partial long$$

$$\begin{pmatrix} \partial E_r \\ \partial E_v \end{pmatrix} = -\begin{pmatrix} \partial \bar{r}_B / \partial inc \\ \partial \bar{v}_B / \partial inc \end{pmatrix} \partial inc$$

$$\begin{pmatrix} \partial E_r \\ \partial E_v \end{pmatrix} = \begin{pmatrix} \bar{v}_B \\ \bar{g}_B \end{pmatrix} \partial ToF$$

$$\begin{pmatrix} \partial E_r \\ \partial E_v \end{pmatrix} = \begin{pmatrix} M_k^{rv} \\ M_k^{vv} \end{pmatrix} \partial(\Delta v_k)$$

In cui:

$$M_k = M(t_k, ToF) = M(ToF)M(t_k)^{-1}$$

Con:

$$t_k = \sum_{i=1}^{k} T_i$$

Quando avremo risolto il problema di ottimo potremmo infine definire i parametri orbitali del punto di arrivo, il combustibile necessario a eseguire la traiettoria di trasferimento, i valori degli angoli di assetto di partenza per la manovra successiva di Descent sulla superficie lunare. Di seguito verrà riportato il codice di risoluzione commentato.

[credits: [10],[34],[35]]

4.2 ANALISI ALGORITMO

Per comprendere meglio la struttura del codice, è utile osservarne il diagramma di flusso, riportato in figura:

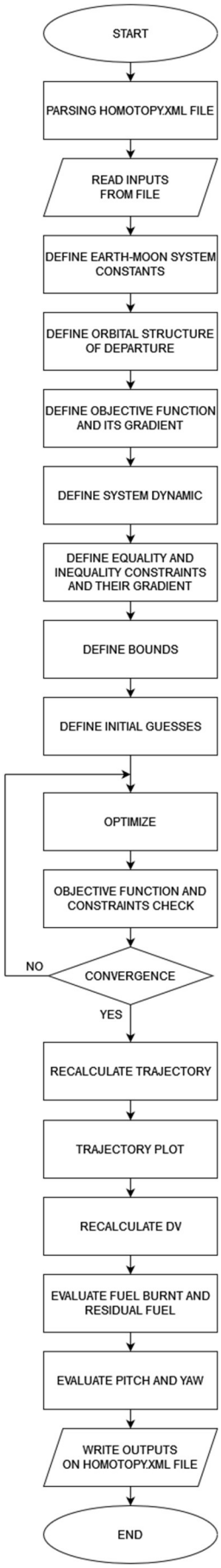

Figura 24 - Diagramma di Flusso del Codice Python

Il codice, dunque, prende inizialmente in input alcuni dati, attraverso la lettura dal file *"homotopy.xml"*, l'inizializzazione di alcune variabili e il calcolo di altre, che rimarranno costanti durante l'esecuzione del programma. Successivamente definisce le funzioni che verranno utilizzate all'interno dell'algoritmo, tra cui quella di definizione della dinamica del sistema CR3BP, della funzione obiettivo e del suo gradiente e dei vincoli e del loro gradiente, oltre che le funzioni utili per il plottaggio e l'integrazione numerica. Definisce quindi i limiti entro i quali possono muoversi le variabili di ottimizzazione e i loro valori tentativo iniziali. Viene perciò cercata la soluzione di ottimo. Attraverso la soluzione ottimizzata è possibile ricalcolare la traiettoria migliore, in termini di Δv, per il trasferimento tra la NRHO di partenza e la LLO di arrivo. L'algoritmo procede successivamente al plottaggio della traiettoria, al calcolo della massa di combustibile consumato per effettuare la manovra e alla valutazione degli angoli di assetto. Infine, i risultati necessari alla definizione dello scenario per la fase successiva di Descent verranno salvati sul file *"homotopy.xml"*.

Andiamo però ora ad analizzare lo script in dettaglio.

In prima battuta vengono importate le librerie necessarie al funzionamento dell'algoritmo. Saranno utilizzate:

- Numpy: consente di definire e operare con vettori e matrici
- Math: consente di ampliare le funzionalità matematiche
- Matplotlib: utilizzata per la realizzazione di grafici
- Scipy: composta da moduli, ognuno dei quali attinente ad un argomento del calcolo scientifico. Sono stati utilizzati i moduli "integrate" per l'integrazione numerica e "optimize" per l'ottimizzazione.
- Xml.etree.ElementTree: modulo utile per la lettura, scrittura, modifica dei file xml

```
import numpy as np
import math
from scipy.integrate import ode
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import xml.etree.ElementTree as ET
```

Apriamo quindi il file *"homotopy.xml"*, necessario per la lettura di alcuni dati indispensabili per l'esecuzione del codice e per la modifica successiva di alcuni valori con i risultati ottenuti, per poter essere poi usati nella simulazione della fase di Descent in Astos. Per la gestione dei file xml, è necessario eseguire il "parsing"

del file, ovvero il processo di lettura e creazione di un'interfaccia di accesso al documento.

```
#PARSING
#it let me utilize data from the xml file

#parsing
mytree=ET.parse("homotopy.xml",parser=ET.XMLParser(encoding='iso-8859-1'))
#finding the root of the document
myroot=mytree.getroot()
#we have to register the namespace
ET.register_namespace('','http://www.astos.de/schema/astos/9.17/scenario')
```

Successivamente si vanno a esplicitare tutti i parametri costanti che utilizzeremo all'interno dello script, scalati opportunamente. Verranno definiti la distanza Terra-Luna, le costanti gravitazionali planetarie dei due corpi celesti, il periodo orbitale e la velocità angolare del sistema Terra-Luna, il raggio terrestre e lunare, la distanza della Terra e della Luna dal centro di massa del sistema, il periodo e il vettore di stato di riferimento della Near-Rectilinear Halo Orbit di partenza, i fattori di scala con cui poi andranno riconvertite le grandezze risultanti. Leggeremo inoltre da file l'altitudine della Low Lunar Orbit target, il valore iniziale di massa di propellente a bordo, la massa a vuoto del velivolo, l'impulso specifico del propulsore.

```
#COSTANTS
#Constant parameters used in the script:

#Earth-Moon distance scaled to unity
a=1 #384400*1000
#standard gravitational parameters scaled to mu_M+mu_E=1
chi=81.3005617547232
mu_E=chi/(chi+1) #3.986004418*10^14
mu_M=1/(chi+1) #4.9048695*10^12

#Adimensional time scale equal to 1
T_=math.sqrt(a**3/(mu_E+mu_M)) #375190.258892627

#Ammount of time it takes the Earth-Moon system to turn:
T_orbit=2*math.pi*T_

#Actual Scales in actual units. After you calculated whatever you need, during post-process you
go back to the real time and distance scales
T_scale=375190.258892627 #Time scale in second
a_scale=384400*1000 #Length scale in meter

#Radius of the Earth and Moon (for plotting only)
```

```
r_E = a*6371*1000/(a_scale)
r_M = a*1737*1000/(a_scale)

#Scalar position of the Moon and Earth in respect to the center of mass:
p_M=a*mu_E/(mu_M+mu_E)
p_E=a*mu_M/(mu_M+mu_E)

#Vectorial position of the Moon and Earth in respect to the center of mass:
P_M=np.array([p_M,0,0])
P_E=np.array([-p_E,0,0])

#Scalar angular speed and vectorial angular speed of the Earth-Moon system:
omega0=math.sqrt((mu_E+mu_M)/a**3)
Omega0=np.array([0,0,omega0])

#Target orbit LLO altitude (taken from the xml file, it is a user input and it's equal to the
initial periapsis(LLO)):
h=float(myroot[9].text)*1000/a_scale
#Even if it's a circular orbit it is suggested to set the initial apoapsis(LLO) a little
different from the initial periapsis:
In_Apo=float(myroot[9].text)+1

#Period of the NRHO
T_0_NRHO=1.627695098326531 # That value should be multiplied by the time scale of the CR3BP
#State on the NRHO - That is a choosen NRHO - is the most commun orbit
#If your end objective is a delta-V, that doesn't change much witch NRHO you are using
p_0_NRHO=np.array([1.030832942481334e+00,0,-1.875673203486770e-01,0,-1.216688050903086e-01,0])

#Data to obtain the residual propellant mass
Empty_mass=float(myroot[48].text)+float(myroot[49].text)
Initial_total_mass=Empty_mass+float(myroot[47].text)
g_0=9.807
Isp=float(myroot[10].text)
```

La seconda parte dello script vede la definizione delle funzioni che verranno usate nell'esecuzione del codice. Avremo:

- Funzione integratore: viene usata per eseguire l'integrazione numerica quando necessaria. Utilizza il metodo Runge Kutta 4(5).
  La funzione prende in input il passo di integrazione, l'intervallo di tempo in cui avviene l'integrazione, la dimensione del vettore soluzione, il vettore soluzione iniziale, la funzione integranda. Restituisce in output il vettore soluzione, che contiene le soluzioni trovate ad ogni passo di integrazione, e il vettore tempo, che contiene il valore di tempo di ogni passo di integrazione.

```
#INTEGRATOR
#This function is only used to integrate when necessary

def integrator(dt,t_0,dim,y_0,fun):
    n_steps=int(np.ceil(abs(t_0)/dt))
    t_int=np.zeros((n_steps,1))
    y_int=np.zeros((n_steps,dim))

    y_int[0]=y_0

    step=1

    solver=ode(fun)
    solver.set_integrator('dopri5') #runge kutta 4(5)
    solver.set_initial_value(y_0,0)

    while solver.successful() and step<n_steps:
        solver.integrate(solver.t+dt)
        t_int[step]=solver.t
        y_int[step]=solver.y
        step+=1

    return t_int,y_int
```

- Funzione plotter: viene usata per la visualizzazione grafica della traiettoria tridimensionale calcolata nell'esecuzione del codice. Prende in input il vettore di stato del sistema, contenente i valori di posizione e velocità ad ogni istante, e utilizza il vettore posizione per tracciare la traiettoria percorsa dal lander. Permette inoltre di visualizzare l'orbita NRHO di partenza, il centro della Terra e della Luna, il centro di massa del sistema Terra-Luna e i punti iniziale e finale della traiettoria di trasferimento ottimizzata.

```
#PLOTTER
#This function is only used for the 3D plot of the optimizated trajectory

def plot(p_t):
    #NRHO calculation
    tA_orbit,pA_orbit=integrator(0.01,T_0_NRHO,6,p_0_NRHO,dyn_no_STM)

    fig=plt.figure()
    ax=fig.add_subplot(111,projection='3d')
    ax.plot(pA_orbit[:,0],pA_orbit[:,1],pA_orbit[:,2],color='b',label='NRHO')
    ax.plot(p_t[0,0],p_t[0,1],p_t[0,2],marker='*',color='b',label='Initial position')
    ax.plot(p_t[-1,0],p_t[-1,1],p_t[-1,2],marker='*',color='m',label='Final position')
    ax.plot(p_t[:,0],p_t[:,1],p_t[:,2],color='m',label='From NRHO to LLO trajectory')
```

```
    ax.plot(-p_E,0,0,marker='o',color='c',label='Centre of Earth')
    ax.plot(0,0,0,marker='o',color='k',label='G')
    ax.plot(p_M,0,0,marker='o',color='b',label='Centre of Moon')
    ax.set_xlabel(['X'])
    ax.set_ylabel(['Y'])
    ax.set_zlabel(['Z'])
    ax.set_title('NRHO to LLO trajectory')
    plt.legend(['NRHO','Initial position','Final position','From NRHO to LLO
trajectory','Centre of Earth','G','Centre of Moon'])
    plt.show()
```

- Funzione dynamics: contiene la dinamica del sistema, con e senza utilizzo della State Transition Matrix. La dinamica implementata è quella del Circular Restricted Three-Body Problem, ampiamente discusso nel capitolo 2. Analizziamo prima la funzione dynamics senza computazione della STM: essa prende in input il vettore di stato del sistema all'istante di tempo zero. Lo scompone in vettore posizione e vettore velocità. Definisce dunque le componenti di accelerazione: gravitazionale dovuta all'azione della Terra e della Luna, centrifuga e di Coriolis. Sommandole ricava l'accelerazione totale. Restituisce dunque il vettore di stato derivato che andrà successivamente integrato. Esso si compone delle tre componenti di velocità e accelerazione appena trovate. Andando a considerare anche la STM, la funzione dynamics subisce delle modifiche. Prende in input il vettore contenente non solo lo stato iniziale del sistema, ma anche il valore iniziale della STM. Procede con il calcolo del vettore di stato derivato come in precedenza. Vengono poi calcolati i gradienti, rispetto alla posizione, dei campi gravitazionali e delle forze centrifughe e il gradiente, rispetto alla velocità, delle forze di Coriolis. Viene così definito il gradiente totale del sistema calcolato rispetto al vettore di stato. Viene dunque restituito in output il vettore contenente lo stato derivato del sistema e il prodotto tra il gradiente totale e la State Transition Matrix iniziale. Tale vettore verrà successivamente integrato e risulterà utile per la computazione del gradiente degli equality constraints.

```
#DYNAMICS
#circular restricted three body problem(with and without state transition matrix(STM))

def dyn_no_STM(t_,p):
    #CR3BP (Without STM)
    r=np.zeros(3)
    v=np.zeros(3)
    r=p[:3]
```

```
    v=p[3:]

    Diff_r_P_M0=np.subtract(r,P_M)
    Diff_r_P_E0=np.subtract(r,P_E)
    mult_diff_M_scalare0=np.dot(Diff_r_P_M0,Diff_r_P_M0)
    mult_diff_E_scalare0=np.dot(Diff_r_P_E0,Diff_r_P_E0)
    #Gravity acceleration from the Moon and Earth on the spacecraft
    g_M=(-mu_M/(mult_diff_M_scalare0**(3/2)))*Diff_r_P_M0
    g_E=(-mu_E/(mult_diff_E_scalare0**(3/2)))*Diff_r_P_E0
    #Centrifugal and coriolis acceleration
    Omega=Omega0
    Omega=Omega[np.newaxis]
    omega_dot=np.dot(Omega.T,Omega)
    dot_om_r=np.dot(omega_dot,r)
    g_cen=np.subtract(omega0**2*r,dot_om_r)
    g_cor=2*omega0*np.array([v[1],-v[0],0])
    #Total gravitational acceleration (component of the gravitations that are
dependent by the position)
    g_grav=np.add(g_M,g_E)
    g_r=np.add(g_grav,g_cen)
    #Total acceleration
    g_tot=np.add(g_r,g_cor)
    #CR3BP equations (Without STM)
    F_p=np.array([v[0],v[1],v[2],g_tot[0],g_tot[1],g_tot[2]])
    return F_p

def dyn_STM(t,S):
    p_=np.zeros(6)
    p_=S[:6]
    r_=np.zeros(3)
    v_=np.zeros(3)
    r_=p_[:3]
    v_=p_[3:]
    #CR3BP (Without STM) (i copied the CR3BP without STM script here)
    Diff_r_P_M0_=np.subtract(r_,P_M)
    Diff_r_P_E0_=np.subtract(r_,P_E)
    mult_diff_M_scalare0_=np.dot(Diff_r_P_M0_,Diff_r_P_M0_)
    mult_diff_E_scalare0_=np.dot(Diff_r_P_E0_,Diff_r_P_E0_)
    #Gravity acceleration from the Moon and Earth on the spacecraft
    g_M_=(-mu_M/(mult_diff_M_scalare0_**(3/2)))*Diff_r_P_M0_
    g_E_=(-mu_E/(mult_diff_E_scalare0_**(3/2)))*Diff_r_P_E0_
    #Centrifugal and coriolis acceleration
    Omega=Omega0
    Omega=Omega[np.newaxis]
    omega_dot=np.dot(Omega.T,Omega)
    dot_om_r_=np.dot(omega_dot,r_)
    g_cen_=np.subtract(omega0**2*r_,dot_om_r_)
    g_cor_=2*omega0*np.array([v_[1],-v_[0],0])
```

```
    #Total gravitational acceleration (component of the gravitations that are
dependent by the position)
    g_grav_=np.add(g_M_,g_E_)
    g_r_=np.add(g_grav_,g_cen_)
    g_tot_=np.add(g_r_,g_cor_)
    #CR3BP equations (Without STM)
    F_p_=np.array([v_[0],v_[1],v_[2],g_tot_[0],g_tot_[1],g_tot_[2]])
    #now the new part is added
    #gradients calculation
    D_M=Diff_r_P_M0_[np.newaxis]
    D_E=Diff_r_P_E0_[np.newaxis]
    mult_diff_M_vett=np.dot(D_M.T,D_M)
    mult_diff_E_vett=np.dot(D_E.T,D_E)
    #Gradient (in respect to the position) of the gravitational field:
    Jg_M=np.subtract(((3*mu_M)/(mult_diff_M_scalare0_**(5/2)))*mult_diff_M_vett,(mu_M/
(mult_diff_M_scalare0_**(3/2)))*np.eye(3))
    Jg_E=np.subtract(((3*mu_E)/(mult_diff_E_scalare0_**(5/2)))*mult_diff_E_vett,(mu_E/
(mult_diff_E_scalare0_**(3/2)))*np.eye(3))
    #Gradient (in respect to the position) of the centrifugal forces:
    Jg_cen=np.subtract((omega0**2)*np.eye(3),omega_dot)
    #Gradient (in respect to the velocity) of the coriolis forces:
    Jg_cor=np.array([[0,2*omega0,0],[-2*omega0,0,0],[0,0,0]])
    #Total gravitational acceleration (component of the gravitations that are
dependent by the position) and total gradient of the gravity forces
    J_grav=np.add(Jg_M,Jg_E)
    Jg_r=np.add(J_grav,Jg_cen)
    #Total gradient of the system in respect to p = [x,y,z,vx,vy,vz]
    J_total=np.array([[0,0,0,1,0,0],[0,0,0,0,1,0],[0,0,0,0,0,1],[Jg_r[0,0],Jg_r[0,1],J
g_r[0,2],Jg_cor[0,0],Jg_cor[0,1],Jg_cor[0,2]],[Jg_r[1,0],Jg_r[1,1],Jg_r[1,2],Jg_cor[1,
0],Jg_cor[1,1],Jg_cor[1,2]],[Jg_r[2,0],Jg_r[2,1],Jg_r[2,2],Jg_cor[2,0],Jg_cor[2,1],Jg_
cor[2,2]]])
    #CR3BP equations (With STM)
    S_mat=np.reshape(S[6:42],(6,6))
    prod_J_S=(np.matmul(J_total,S_mat))
    F_0=np.reshape(prod_J_S,(1,36))
    F=np.zeros(42)
    F[:6]=F_p_
    F[6:]=F_0
    return F
```

- Funzione objective function: ha l'obiettivo di definire la funzione obiettivo da minimizzare e il suo gradiente. Come indicato nella sezione di definizione del problema, la funzione di cui si ricerca il minimo è il Δv totale del sistema, considerando due manovre impulsive compiute dallo spacecraft. Le funzioni analizzate prenderanno entrambe in input il vettore di variabili dell'ottimizzazione e restituiranno in output rispettivamente il Δv totale e il suo gradiente.

```
#OBJECTIVE FUNCTION (AND ITS GRADIENT)
#The function to minimize is the total dV characterized by the dV of two impulsive
manouvers

def dv_calc_LLO(__x__):
    dv=np.linalg.norm(__x__[4:7])+np.linalg.norm(__x__[7:10])
    return dv

def dv_calc_LLO_GRAD(_x_):
    dv_grad=np.zeros(10)
    i=4
    while i<7:
        dv_grad[i]=_x_[i]/np.linalg.norm(_x_[4:7])
        i+=1
    j=7
    while j<10:
        dv_grad[j]=_x_[j]/np.linalg.norm(_x_[7:10])
        j+=1
    return dv_grad
```

- Funzione coordinate: ha lo scopo di permettere il cambio di sistema di riferimento. Le coordinate del punto target di arrivo, espresse come RAAN e True Anomaly, si trasformano nelle coordinate cartesiane espresse nel sistema di riferimento non inerziale del CR3BP, definito nei capitoli precedenti. Vengono inoltre calcolati i gradienti dei vettori posizione e velocità rispetto alla RAAN e alla True Anomaly. Per calcolare però il vettore di stato e il suo gradiente all'interno del sistema del CR3BP, viene effettuata una trasformazione intermedia verso un sistema di riferimento non inerziale, centrato nel centro della Luna e i quali assi rappresentano la traslazione di quelli del sistema di riferimento del CR3BP.

```
#COORDINATES MOVED IN THE CR3BP REFERENCE FRAME AND THEIR GRADIENTS
#This function is only used to change reference frame

def coordinates(__x_):
    long=__x_[1]
    inc=__x_[2]

    #Cartesian components starting from incl and long of the final point
    #They indicates directions of vectors in the cartesian space centered on the Moon
not magnitude
```

```
    r_unit=np.array([math.cos(long)*math.sin(inc),math.sin(long)*math.sin(inc),math.co
s(inc)])
    v_unit=np.array([math.cos(long)*math.cos(inc),math.sin(long)*math.cos(inc),-
math.sin(inc)])
    #Now you get the full vector with the magnitude involved!!! Always centered on the
Moon yet
    r_iner=(r_M+h)*r_unit
    v_iner=math.sqrt(mu_M/(r_M+h))*v_unit

    #Gradient of the position vector in respect to longitude
    r_iner_long=(r_M+h)*np.array([-
math.sin(long)*math.sin(inc),math.cos(long)*math.sin(inc),0])
    #Gradient of the position vector in respect to inclination
    r_iner_inc=(r_M+h)*v_unit
    #Same for the velocity
    v_iner_long=math.sqrt(mu_M/(r_M+h))*np.array([-
math.sin(long)*math.cos(inc),math.cos(long)*math.cos(inc),0])
    v_iner_inc=-math.sqrt(mu_M/(r_M+h))*r_unit

    #Now the coordinates are centered in the center of the Moon - we have to convert
them to the reference frame of the CR3BP (non inertial frame)
    _r=np.add(r_iner,P_M)
    _v=np.subtract(v_iner,np.cross(Omega0,r_iner))

    r_long=r_iner_long
    r_inc=r_iner_inc

    v_long=np.subtract(v_iner_long,np.cross(Omega0,r_iner_long))
    v_inc=np.subtract(v_iner_inc,np.cross(Omega0,r_iner_inc))

    return _r,_v,r_long,r_inc,v_long,v_inc
```

- Funzione trajectory and equality constraints: viene utilizzata per la computazione della traiettoria da Near-Rectilinear Halo Orbit a Low Lunar Orbit e degli equality constraints. La funzione prende in input il vettore variabili dell'ottimizzazione. Ne ricava il valore di tempo che identifica il punto di partenza della traiettoria, il Time of Flight e i vettori Δv_1 e Δv_2. Viene inoltre utilizzato il vettore variabili per ricavare il vettore di stato, con relativo gradiente, del punto target di arrivo, tramite la funzione coordinate definita sopra. Tale vettore di stato verrà utilizzato per il calcolo dell'errore di posizione e velocità. Andiamo quindi a estrapolare il vettore di stato di partenza della traiettoria: viene integrata la dinamica del CR3BP, partendo dal vettore di stato del punto di riferimento della NRHO, sul valore di tempo T_A che definisce appunto lo stato iniziale del sistema. Bisogna poi aggiornare le componenti di velocità del vettore di stato appena trovato, sommandovi le

componenti del Δv_1. In seguito viene definito lo stato iniziale della STM, definito come una matrice identità. Costruiamo quindi il vettore iniziale per la prossima integrazione numerica. Sarà costituito dalle componenti del vettore di stato iniziale, trovato tramite integrazione appena prima, e della STM iniziale, appena definita. Integrando sul Time of Flight otteniamo il vettore di stato e la STM ad ogni passo della traiettoria. Il vettore di stato del punto di arrivo dovrà essere aggiornato sommando alle componenti di velocità le componenti del vettore Δv_2. Andremo così a calcolare gli equality constraints, sottraendo al vettore di stato finale appena computato il vettore di stato del punto di arrivo target definito inizialmente nella funzione, e il loro gradiente, sfruttando le quantità ricavate. La funzione restituirà infine lo stato del lander in ogni punto della traiettoria, gli equality constraints e il loro gradiente. Questi ultimi outputs verranno estratti singolarmente tramite apposite funzioni. In ultima analisi, è necessario porre particolare attenzione alla scelta del passo di integrazione. Deve essere sufficientemente piccolo, altrimenti si rischia di avere valori troppo elevati di equality constraints in seguito all'ottimizzazione, lontani dal valore nullo a cui dovrebbero tendere. A risentirne però sarà la velocità di esecuzione dell'algoritmo.

```
#TRAJECTORY AND EQ CONSTRAINTS FUNCTION
#This function calculates the trajectory from the NRHO to the LLO using the dynamic
equations and the integrator
#It also calculates the equality constraints and their gradient

def traj_calc_LLO(x):
    TA=x[0]

    ToF=x[3]

    dv1=x[4:7]
    dv2=x[7:10]

    #we calculate the coordinates in the CR3BP reference frame
    _r_,_v_,_r_long_,_r_inc_,_v_long_,_v_inc_=coordinates(x)

    #State vector with position and velocity
    pB=np.zeros(6)
    pB[:3]=_r_
    pB[3:]=_v_

    #if TA is longer that the period of the initial orbit this will give you the
correct initial position
    TA=TA%T_0_NRHO
```

```
    #now we integrate
    tAlist,pAlist=integrator(0.00001,TA,6,p_0_NRHO,dyn_no_STM)

    pA=pAlist[-1,:]
    p0=pA

    #Update the velocity vector with the delta_V of the manouver to have the full
initial condition to compute the transfer arc
    p0[3:]=np.add(pA[3:],dv1)

    #Initial State transition matrix - will change in the integration
    M0=np.eye(6,6)
    s0=np.reshape(M0,(36))
    S0=np.zeros(42)
    S0[:6]=p0
    S0[6:]=s0

    #we integrate again
    t_transfer,S_transfer=integrator(0.00001,ToF,42,S0,dyn_STM)

    p_transfer_=S_transfer[:,:6]

    M_transfer=np.zeros((6,6,len(t_transfer)))
    for i in range(len(t_transfer)):
        M_transfer[:,:,i]=np.reshape(S_transfer[i,6:42],(6,6))

    p_end=p_transfer_[-1,:]
    pend=p_end

    #Update the velocity vector with the delta_V of the manouver to have the full
final condition
    p_end[3:]=np.add(pend[3:],dv2)

    #Evaluation of the error in position and velocity (this are the equality
constraints)
    ceq_=np.subtract(p_end,pB)

    #gradient of the equality constraints in respect to your x-variable vector
(variables you try to optimize)
    ceq_grad_=np.zeros((6,10))
    ceq_grad_[:,0]=np.matmul(M_transfer[:,:,-1],dyn_no_STM(0,pA))
    ceq_grad_[:3,1]=-_r_long_
    ceq_grad_[3:,1]=-_v_long_
    ceq_grad_[:3,2]=-_r_inc_
    ceq_grad_[3:,2]=-_v_inc_
    ceq_grad_[:,3]=dyn_no_STM(0,p_end)
    ceq_grad_[:,4:7]=M_transfer[:,3:6,-1]
    ceq_grad_[:,7:10]=M_transfer[:,3:6,0]
```

```
    return p_transfer_,ceq_,ceq_grad_

#we extract only the equality constraints from the traj_calc_LLO returned values
def get_ceq(_x):
    _p_transfer,_ceq_,_ceq_grad=traj_calc_LLO(_x)
    return _ceq_

#we extract only the gradient of the equality constraints from the traj_calc_LLO
returned values
def get_ceq_grad(x_):
    _p_transfer,_ceq,_ceq_grad_=traj_calc_LLO(x_)
    return _ceq_grad_
```

- Funzione inequality constraints: prende in input il vettore variabili di ottimizzazione e restituisce in output gli inequality constraints, calcolati tramite una semplice equazione matriciale.

```
#INEQ CONSTRAINTS FUNCTION
#This function returns the inequality constraint

def ineq_constraint(_x__):
    A=np.zeros(10)
    A[3]=1
    Tmax=10
    B=Tmax
    return -np.matmul(A,_x__) + B #-Ax+b>=0
```

Terminata la definizione delle funzioni necessarie all'esecuzione del programma, dovremo effettuare l'ottimizzazione vera e propria. Prima di ciò però andremo a definire bounds e initial guesses. I bounds definiscono i limiti entro i quali le variabili di ottimizzazione possono muoversi. Possiamo semplicemente lasciar variare i valori tra $\pm\infty$. Un unico lower bound è posto in corrispondenza del Time of Flight. Non ha senso fisico infatti permettergli di variare verso il semiasse negativo dell'asse reale. Vengono inoltre esclusi valori eccessivamente piccoli per evitare che l'ottimizzatore selezioni grandezze di tempo che non permetterebbero di raggiungere l'orbita finale desiderata.

```
#BOUNDS
#we define the limits within which the x-variable vector can vary

l_b=-math.inf*np.ones(10)
u_b=-l_b
l_b[3]=0.5
bnds=[]
for i in range(len(l_b)):
```

```
    bnds.append((l_b[i],u_b[i]))
```

Gli initial guesses si compongono dei valori di:

- Punto di partenza, espresso come tempo di percorrenza dell'orbita iniziale rispetto ad un punto di riferimento sulla suddetta orbita
- Punto di arrivo, espresso in termini di RAAN e True Anomaly
- Time of Flight
- Δv_1
- Δv_2

Bisogna selezionare i valori iniziali con una elevata accuratezza per far sì che l'ottimizzatore trovi il minimo globale della funzione. I valori selezionati sono stati scelti tramite l'aiuto di un codice di ottimizzazione sviluppato in Matlab.

```
#INITIAL GUESS
#we define the vector with the initial guess
#The initial guess have to be very accurate or the optimization will not be able to find the
correct global minimum

#initial position:
TA_0=0.1928#0.1876
#Arrival point in LLo with longitude and inclination in the CR3TBP rotating frame (respect to the
x-axis):
long_0=1.8297#1.8345
inc_0=6.3785#6.3804# #from 0 to 2pi - important to know if the trajectory is ascending or
descending
#time of flight (adimensional):
ToF_0=0.5959#0.6001
#Two burn manouvers:
dv1_0=np.array([-0.0088,0.0359,-0.0094])#([-0.0096,0.0368,-0.0095])
dv2_0=np.array([0.1583,-0.6165,0.0606])#([0.1652,-0.6303,0.0633])
#Initial vector to optimize:
x0=np.array([TA_0,long_0,inc_0,ToF_0,dv1_0[0],dv1_0[1],dv1_0[2],dv2_0[0],dv2_0[1],dv2_0[2]])
```

Ora che tutti i parametri necessari sono stati definiti, è possibile eseguire l'ottimizzazione. Andremo ad utilizzare "*scipy.optimize.minimize*", che permette la ricerca del minimo per una funzione scalare di una o più variabili. Verranno forniti i seguenti parametri:

- La funzione obiettivo da minimizzare
- Il vettore contenente le initial guesses
- Il metodo di risoluzione

- Il gradiente della funzione obiettivo
- I bounds per le variabili
- I vincoli, specificandone la tipologia ed eventuale gradiente
- La tolleranza

```
#OPTIMIZATION

sol=minimize(dv_calc_LLO,x0,method='SLSQP',jac=dv_calc_LLO_GRAD,bounds=bnds,constraints={'type':
'ineq', 'fun': ineq_constraint,'type': 'eq', 'fun': get_ceq, 'jac':
get_ceq_grad},options={"ftol": 1e-6,"disp": True})
x_sol=sol.x
```

Le variabili soluzione ricavate verranno utilizzate per il ricalcolo della traiettoria ottimizzata, tramite l'apposita funzione sopra definita. Possiamo calcolare e visualizzare anche i valori finali di equality e inequality constraints in modo da verificarne la conformità.

```
#TRAJECTORY AND CONSTRAINTS RECALCULATION
#We use the solution of the optimization to calculate the final optimized trajectory and
constraints values

p_trans,c_eq,c_eq_grad=traj_calc_LLO(x_sol)
print('eq_con:',c_eq)
ineq_con=ineq_constraint(x_sol)
print('ineq_con:',ineq_con)
```

Avendo ricalcolato la traiettoria, potremo ora stamparla a schermo tramite la funzione plot già definita.

```
#PLOT
#We plot the optimized trajectory just calculated

plot(p_trans)
```

Dovremo inoltre ricalcolare il valore di Δv della traiettoria effettuata. Esso rappresenterà il Δv minimo necessario per effettuare un trasferimento dalla NRHO alla LLO di riferimento. È importante ricordare di riportare a valori reali i valori scalati inizialmente.

```
#DELTA V RECALCULATION
#We use the solution of the optimization to calculate the value of the minimum delta V

DV=dv_calc_LLO(x_sol)
```

```
DV=DV*a_scale/T_scale
print('DV:',DV)
```

Conoscendo il Δv, possiamo inserirlo all'interno dell'equazione del razzo di Ciolkovskij (Tsiolkovsky con translitterazione anglosassone). Tramite questa equazione sarà possibile ricavare la massa di propellente consumata per l'esecuzione della manovra e quindi la massa di propellente residua per i successivi trasferimenti.

```
#FUL BURNT AND RESIDUAL PROPELLANT MASS
#I used the Tsiolkovsky rocket equation

#The propellant mass burnt for our DV is:
Fuel_burnt=Initial_total_mass*(1-(math.exp(-(DV/(Isp*g_0)))))
#Now we calculate the residual propellant mass:
Residual_fuel=float(myroot[47].text)-Fuel_burnt
print('Fuel burnt:',Fuel_burnt)
```

In ultima analisi dobbiamo valutare gli angoli di pitch e yaw al punto di arrivo dell'orbita target. Per ricavarli sfrutteremo la direction cosine matrix per effettuare una rotazione in assi body.

```
#PITCH AND YAW CALCULATION
#We operate a rotation between the reference frames to obtain pitch and yaw values

Q=np.array([p_trans[-1,3]*a_scale/T_scale,p_trans[-1,4]*a_scale/T_scale,p_trans[-
1,5]*a_scale/T_scale])
Q_norm=Q/np.linalg.norm(Q)
if Q_norm[0]!=0 and Q_norm[1]!=0:
    pitch=math.degrees(math.atan2(Q_norm[0],Q_norm[1]))
else:
    pitch=0
if Q_norm[2]!=0:
    yaw=math.degrees(math.asin(Q_norm[2]))
else:
    yaw=0
print('Pitch:',pitch)
print('Yaw:',yaw)
```

Possiamo infine modificare il file *"homotopy.xml"* aggiornandolo con i risultati ottenuti grazie al codice in esame. In particolare andremo ad inserire i nuovi valori di:

- Inclinazione dell'orbita LLO di arrivo, impostata a 90° poiché abbiamo semplificato l'algoritmo considerando un'orbita polare
- Argomento del pericentro dell'orbita LLO di arrivo, che per orbite circolari può essere assunto pari a 0°
- Ascensione retta del nodo ascendente dell'orbita LLO di arrivo, calcolata tramite ottimizzazione
- Anomalia vera dell'orbita LLO di arrivo, calcolata anch'essa tramite ottimizzazione
- Periasse e apoasse dell'orbita LLO di arrivo. Essendo l'orbita circolare dovrebbero avere un valore pari all'altitudine dell'orbita stessa, tuttavia, è consigliabile imporre un leggero discostamento per facilitare la simulazione
- Combustibile residuo, calcolato precedentemente
- Angoli di pitch e yaw, calcolati sopra

```
#HOMOTOPY FILE UPDATE
#Now we have to modify the orbital parameters and the residual propellant values in the xml file
"homotopy"

#Inclination LLO update: 90 degrees (polar orbit):
for elem in myroot[4].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem.text = "90.0"
#Initial Arg of Periapsis (LLO) update: (assumed to be 0 for circular orbits):
for elem0 in myroot[5].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem0.text = "0.0"
#initial RAAN (LLO) update: (variable long):
if math.degrees(x_sol[1])<=360:
    RAAN=math.degrees(x_sol[1])
else:
    RAAN=math.degrees(x_sol[1])-360
for elem1 in myroot[6].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem1.text = "{:.1f}".format(RAAN)
#Initial True Anomaly (LLO) update: (variable inc, zero defined by the north pole):
if math.degrees(x_sol[2])<=360:
    True_Anomaly=math.degrees(x_sol[2])
else:
    True_Anomaly=math.degrees(x_sol[2])-360
for elem2 in myroot[7].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem2.text = "{:.1f}".format(True_Anomaly)
#Initial Periapsis (LLO) update:
for elem3 in myroot[9].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem3.text = "{:.1f}".format(h*a_scale/1000)
#Initial Apoapsis (LLO) update:
for elem4 in myroot[8].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem4.text = "{:.1f}".format(In_Apo)
#Residual fuel:
for elem5 in myroot[47].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
```

```
    elem5.text = "{:.1f}".format(Residual_fuel)
#Pitch update:
for elem6 in myroot[21].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem6.text = "{:.1f}".format(pitch)
#Yaw update:
for elem7 in myroot[22].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem7.text = "{:.1f}".format(yaw)

mytree=ET.ElementTree(myroot)

mytree.write("homotopy.xml")
```

4.3 CODICE COMPLETO

```
import numpy as np
import math
from scipy.integrate import ode
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import xml.etree.ElementTree as ET
#N.B. Put all the files that we open in the same folder in which the script is saved


#PARSING
#it let me utilize data from the xml file

#parsing
mytree=ET.parse("homotopy.xml",parser=ET.XMLParser(encoding='iso-8859-1'))
#finding the root of the document
myroot=mytree.getroot()
#we have to register the namespace
ET.register_namespace('','http://www.astos.de/schema/astos/9.17/scenario')

#COSTANTS
#Constant parameters used in the script:

#Earth-Moon distance scaled to unity
a=1 #384400*1000
#standard gravitational parameters scaled to mu_M+mu_E=1
chi=81.3005617547232
mu_E=chi/(chi+1) #3.986004418*10^14
mu_M=1/(chi+1) #4.9048695*10^12

#Adimensional time scale equal to 1
T_=math.sqrt(a**3/(mu_E+mu_M)) #375190.258892627

#Ammount of time it takes the Earth-Moon system to turn:
T_orbit=2*math.pi*T_
```

```
#Actual Scales in actual units. After you calculated whatever you need, during post-process you
go back to the real time and distance scales
T_scale=375190.258892627 #Time scale in second
a_scale=384400*1000 #Length scale in meter

#Radius of the Earth and Moon (for plotting only)
r_E = a*6371*1000/(a_scale)
r_M = a*1737*1000/(a_scale)

#Scalar position of the Moon and Earth in respect to the center of mass:
p_M=a*mu_E/(mu_M+mu_E)
p_E=a*mu_M/(mu_M+mu_E)

#Vectorial position of the Moon and Earth in respect to the center of mass:
P_M=np.array([p_M,0,0])
P_E=np.array([-p_E,0,0])

#Scalar angular speed and vectorial angular speed of the Earth-Moon system:
omega0=math.sqrt((mu_E+mu_M)/a**3)
Omega0=np.array([0,0,omega0])

#Target orbit LLO altitude (taken from the xml file, it is a user input and it's equal to the
initial periapsis(LLO)):
h=float(myroot[9].text)*1000/a_scale
#Even if it's a circular orbit it is suggested to set the initial apoapsis(LLO) a #little
different from the initial periapsis:
In_Apo=float(myroot[9].text)+1

#Period of the NRHO
T_0_NRHO=1.627695098326531 # That value should be multiplied by the time scale of #the CR3BP
#State on the NRHO - That is a choosen NRHO - is the most commun orbit
#If your end objective is a delta-V, that doesn't change much witch NRHO you are using
p_0_NRHO=np.array([1.030832942481334e+00,0,-1.875673203486770e-01,0,-1.216688050903086e-01,0])

#Data to obtain the residual propellant mass
Empty_mass=float(myroot[48].text)+float(myroot[49].text)
Initial_total_mass=Empty_mass+float(myroot[47].text)
g_0=9.807
Isp=float(myroot[10].text)

#INTEGRATOR
#This function is only used to integrate when necessary

def integrator(dt,t_0,dim,y_0,fun):
    n_steps=int(np.ceil(abs(t_0)/dt))
    t_int=np.zeros((n_steps,1))
    y_int=np.zeros((n_steps,dim))
```

```
    y_int[0]=y_0

    step=1

    solver=ode(fun)
    solver.set_integrator('dopri5') #runge kutta 4(5)
    solver.set_initial_value(y_0,0)

    while solver.successful() and step<n_steps:
        solver.integrate(solver.t+dt)
        t_int[step]=solver.t
        y_int[step]=solver.y
        step+=1

    return t_int,y_int

#PLOTTER
#This function is only used for the 3D plot of the optimizated trajectory

def plot(p_t):
    #NRHO calculation
    tA_orbit,pA_orbit=integrator(0.01,T_0_NRHO,6,p_0_NRHO,dyn_no_STM)

    fig=plt.figure()
    ax=fig.add_subplot(111,projection='3d')
    ax.plot(pA_orbit[:,0],pA_orbit[:,1],pA_orbit[:,2],color='b',label='NRHO')
    ax.plot(p_t[0,0],p_t[0,1],p_t[0,2],marker='*',color='b',label='Initial position')
    ax.plot(p_t[-1,0],p_t[-1,1],p_t[-1,2],marker='*',color='m',label='Final position')
    ax.plot(p_t[:,0],p_t[:,1],p_t[:,2],color='m',label='From NRHO to LLO trajectory')
    ax.plot(-p_E,0,0,marker='o',color='c',label='Centre of Earth')
    ax.plot(0,0,0,marker='o',color='k',label='G')
    ax.plot(p_M,0,0,marker='o',color='b',label='Centre of Moon')
    ax.set_xlabel(['X'])
    ax.set_ylabel(['Y'])
    ax.set_zlabel(['Z'])
    ax.set_title('NRHO to LLO trajectory')
    plt.legend(['NRHO','Initial position','Final position','From NRHO to LLO trajectory','Centre
of Earth','G','Centre of Moon'])
    plt.show()

#DYNAMICS
#circular restricted three body problem(with and without state transition matrix(STM))

def dyn_no_STM(t_,p):
    #CR3BP (Without STM)
    r=np.zeros(3)
    v=np.zeros(3)
    r=p[:3]
    v=p[3:]
```

```
    Diff_r_P_M0=np.subtract(r,P_M)
    Diff_r_P_E0=np.subtract(r,P_E)
    mult_diff_M_scalare0=np.dot(Diff_r_P_M0,Diff_r_P_M0)
    mult_diff_E_scalare0=np.dot(Diff_r_P_E0,Diff_r_P_E0)
    #Gravity acceleration from the Moon and Earth on the spacecraft
    g_M=(-mu_M/(mult_diff_M_scalare0**(3/2)))*Diff_r_P_M0
    g_E=(-mu_E/(mult_diff_E_scalare0**(3/2)))*Diff_r_P_E0
    #Centrifugal and coriolis acceleration
    Omega=Omega0
    Omega=Omega[np.newaxis]
    omega_dot=np.dot(Omega.T,Omega)
    dot_om_r=np.dot(omega_dot,r)
    g_cen=np.subtract(omega0**2*r,dot_om_r)
    g_cor=2*omega0*np.array([v[1],-v[0],0])
    #Total gravitational acceleration (component of the gravitations that are dependent by the
position)
    g_grav=np.add(g_M,g_E)
    g_r=np.add(g_grav,g_cen)
    #Total acceleration
    g_tot=np.add(g_r,g_cor)
    #CR3BP equations (Without STM)
    F_p=np.array([v[0],v[1],v[2],g_tot[0],g_tot[1],g_tot[2]])
    return F_p

def dyn_STM(t,S):
    p_=np.zeros(6)
    p_=S[:6]
    r_=np.zeros(3)
    v_=np.zeros(3)
    r_=p_[:3]
    v_=p_[3:]
    #CR3BP (Without STM) (i copied the CR3BP without STM script here)
    Diff_r_P_M0_=np.subtract(r_,P_M)
    Diff_r_P_E0_=np.subtract(r_,P_E)
    mult_diff_M_scalare0_=np.dot(Diff_r_P_M0_,Diff_r_P_M0_)
    mult_diff_E_scalare0_=np.dot(Diff_r_P_E0_,Diff_r_P_E0_)
    #Gravity acceleration from the Moon and Earth on the spacecraft
    g_M_=(-mu_M/(mult_diff_M_scalare0_**(3/2)))*Diff_r_P_M0_
    g_E_=(-mu_E/(mult_diff_E_scalare0_**(3/2)))*Diff_r_P_E0_
    #Centrifugal and coriolis acceleration
    Omega=Omega0
    Omega=Omega[np.newaxis]
    omega_dot=np.dot(Omega.T,Omega)
    dot_om_r_=np.dot(omega_dot,r_)
    g_cen_=np.subtract(omega0**2*r_,dot_om_r_)
    g_cor_=2*omega0*np.array([v_[1],-v_[0],0])
    #Total gravitational acceleration (component of the gravitations that are dependent by the
position)
```

```
    g_grav_=np.add(g_M_,g_E_)
    g_r_=np.add(g_grav_,g_cen_)
    g_tot_=np.add(g_r_,g_cor_)
    #CR3BP equations (Without STM)
    F_p_=np.array([v_[0],v_[1],v_[2],g_tot_[0],g_tot_[1],g_tot_[2]])
    #now the new part is added
    #gradients calculation
    D_M=Diff_r_P_M0_[np.newaxis]
    D_E=Diff_r_P_E0_[np.newaxis]
    mult_diff_M_vett=np.dot(D_M.T,D_M)
    mult_diff_E_vett=np.dot(D_E.T,D_E)
    #Gradient (in respect to the position) of the gravitational field:
    Jg_M=np.subtract(((3*mu_M)/(mult_diff_M_scalare0_**(5/2)))*mult_diff_M_vett,(mu_M/(mult_diff_
M_scalare0_**(3/2)))*np.eye(3))
    Jg_E=np.subtract(((3*mu_E)/(mult_diff_E_scalare0_**(5/2)))*mult_diff_E_vett,(mu_E/(mult_diff_
E_scalare0_**(3/2)))*np.eye(3))
    #Gradient (in respect to the position) of the centrifugal forces:
    Jg_cen=np.subtract((omega0**2)*np.eye(3),omega_dot)
    #Gradient (in respect to the velocity) of the coriolis forces:
    Jg_cor=np.array([[0,2*omega0,0],[-2*omega0,0,0],[0,0,0]])
    #Total gravitational acceleration (component of the gravitations that are dependent by the
position) and total gradient of the gravity forces
    J_grav=np.add(Jg_M,Jg_E)
    Jg_r=np.add(J_grav,Jg_cen)
    #Total gradient of the system in respect to p = [x,y,z,vx,vy,vz]
    J_total=np.array([[0,0,0,1,0,0],[0,0,0,0,1,0],[0,0,0,0,0,1],[Jg_r[0,0],Jg_r[0,1],Jg_r[0,2],Jg
_cor[0,0],Jg_cor[0,1],Jg_cor[0,2]],[Jg_r[1,0],Jg_r[1,1],Jg_r[1,2],Jg_cor[1,0],Jg_cor[1,1],Jg_cor[
1,2]],[Jg_r[2,0],Jg_r[2,1],Jg_r[2,2],Jg_cor[2,0],Jg_cor[2,1],Jg_cor[2,2]]])
    #CR3BP equations (With STM)
    S_mat=np.reshape(S[6:42],(6,6))
    prod_J_S=(np.matmul(J_total,S_mat))
    F_0=np.reshape(prod_J_S,(1,36))
    F=np.zeros(42)
    F[:6]=F_p_
    F[6:]=F_0
    return F

#OBJECTIVE FUNCTION (AND ITS GRADIENT)
#The function to minimize is the total dV characterized by the dV of two impulsive manouvers

def dv_calc_LLO(__x__):
    dv=np.linalg.norm(__x__[4:7])+np.linalg.norm(__x__[7:10])
    return dv

def dv_calc_LLO_GRAD(_x_):
    dv_grad=np.zeros(10)
    i=4
    while i<7:
        dv_grad[i]=_x_[i]/np.linalg.norm(_x_[4:7])
```

```
        i+=1
    j=7
    while j<10:
        dv_grad[j]=_x_[j]/np.linalg.norm(_x_[7:10])
        j+=1
    return dv_grad

#COORDINATES MOVED IN THE CR3BP REFERENCE FRAME AND THEIR GRADIENTS
#This function is only used to change reference frame

def coordinates(__x_):
    long=__x_[1]
    inc=__x_[2]

    #Cartesian components starting from incl and long of the final point
    #They indicates directions of vectors in the cartesian space centered on the Moon not
magnitude
    r_unit=np.array([math.cos(long)*math.sin(inc),math.sin(long)*math.sin(inc),math.cos(inc)])
    v_unit=np.array([math.cos(long)*math.cos(inc),math.sin(long)*math.cos(inc),-math.sin(inc)])
    #Now you get the full vector with the magnitude involved!!! Always centered on the Moon yet
    r_iner=(r_M+h)*r_unit
    v_iner=math.sqrt(mu_M/(r_M+h))*v_unit

    #Gradient of the position vector in respect to longitude
    r_iner_long=(r_M+h)*np.array([-math.sin(long)*math.sin(inc),math.cos(long)*math.sin(inc),0])
    #Gradient of the position vector in respect to inclination
    r_iner_inc=(r_M+h)*v_unit
    #Same for the velocity
    v_iner_long=math.sqrt(mu_M/(r_M+h))*np.array([-
math.sin(long)*math.cos(inc),math.cos(long)*math.cos(inc),0])
    v_iner_inc=-math.sqrt(mu_M/(r_M+h))*r_unit

    #Now the coordinates are centered in the center of the Moon - we have to convert them to the
reference frame of the CR3BP (non inertial frame)
    _r=np.add(r_iner,P_M)
    _v=np.subtract(v_iner,np.cross(Omega0,r_iner))

    r_long=r_iner_long
    r_inc=r_iner_inc

    v_long=np.subtract(v_iner_long,np.cross(Omega0,r_iner_long))
    v_inc=np.subtract(v_iner_inc,np.cross(Omega0,r_iner_inc))

    return _r,_v,r_long,r_inc,v_long,v_inc

#TRAJECTORY AND EQ CONSTRAINTS FUNCTION
#This function calculates the trajectory from the NRHO to the LLO using the dynamic equations and
the integrator
#It also calculates the equality constraints and their gradient
```

```
def traj_calc_LLO(x):
    TA=x[0]

    ToF=x[3]

    dv1=x[4:7]
    dv2=x[7:10]

    #we calculate the coordinates in the CR3BP reference frame
    _r_,_v_,_r_long_,_r_inc_,_v_long_,_v_inc_=coordinates(x)

    #State vector with position and velocity
    pB=np.zeros(6)
    pB[:3]=_r_
    pB[3:]=_v_

    #if TA is longer that the period of the initial orbit this will give you the correct initial
position
    TA=TA%T_0_NRHO

    #now we integrate
    tAlist,pAlist=integrator(0.00001,TA,6,p_0_NRHO,dyn_no_STM)

    pA=pAlist[-1,:]
    p0=pA

    #Update the velocity vector with the delta_V of the manouver to have the full initial
condition to compute the transfer arc
    p0[3:]=np.add(pA[3:],dv1)

    #Initial State transition matrix - will change in the integration
    M0=np.eye(6,6)
    s0=np.reshape(M0,(36))
    S0=np.zeros(42)
    S0[:6]=p0
    S0[6:]=s0

    #we integrate again
    t_transfer,S_transfer=integrator(0.00001,ToF,42,S0,dyn_STM)

    p_transfer_=S_transfer[:,:6]

    M_transfer=np.zeros((6,6,len(t_transfer)))
    for i in range(len(t_transfer)):
        M_transfer[:,:,i]=np.reshape(S_transfer[i,6:42],(6,6))

    p_end=p_transfer_[-1,:]
    pend=p_end
```

```
    #Update the velocity vector with the delta_V of the manouver to have the full final condition
    p_end[3:]=np.add(pend[3:],dv2)

    #Evaluation of the error in position and velocity (this are the equality constraints)
    ceq_=np.subtract(p_end,pB)

    #gradient of the equality constraints in respect to your x-variable vector (variables you try
to optimize)
    ceq_grad_=np.zeros((6,10))
    ceq_grad_[:,0]=np.matmul(M_transfer[:,:,-1],dyn_no_STM(0,pA))
    ceq_grad_[:3,1]=-_r_long_
    ceq_grad_[3:,1]=-_v_long_
    ceq_grad_[:3,2]=-_r_inc_
    ceq_grad_[3:,2]=-_v_inc_
    ceq_grad_[:,3]=dyn_no_STM(0,p_end)
    ceq_grad_[:,4:7]=M_transfer[:,3:6,-1]
    ceq_grad_[:,7:10]=M_transfer[:,3:6,0]

    return p_transfer_,ceq_,ceq_grad_

#we extract only the equality constraints from the traj_calc_LLO returned values
def get_ceq(_x):
    _p_transfer,_ceq_,_ceq_grad=traj_calc_LLO(_x)
    return _ceq_

#we extract only the gradient of the equality constraints from the traj_calc_LLO returned values
def get_ceq_grad(x_):
    _p_transfer,_ceq,_ceq_grad_=traj_calc_LLO(x_)
    return _ceq_grad_

#INEQ CONSTRAINTS FUNCTION
#This function returns the inequality constraint

def ineq_constraint(_x__):
    A=np.zeros(10)
    A[3]=1
    Tmax=10
    B=Tmax
    return -np.matmul(A,_x__) + B #-Ax+b>=0


########## MAIN ##########


#BOUNDS
#we define the limits within which the x-variable vector can vary

l_b=-math.inf*np.ones(10)
u_b=-l_b
```

```
l_b[3]=0.5
bnds=[]
for i in range(len(l_b)):
    bnds.append((l_b[i],u_b[i]))

#INITIAL GUESS
#we define the vector with the initial guess
#The initial guess have to be very accurate or the optimization will not be able to find the
correct global minimum

#initial position:
TA_0=0.1928#0.1876
#Arrival point in LLo with longitude and inclination in the CR3TBP rotating frame (respect to the
x-axis):
long_0=1.8297#1.8345
inc_0=6.3785#6.3804# #from 0 to 2pi - important to know if the trajectory is ascending or
descending
#time of flight (adimensional):
ToF_0=0.5959#0.6001
#Two burn manouvers:
dv1_0=np.array([-0.0088,0.0359,-0.0094])#([-0.0096,0.0368,-0.0095])
dv2_0=np.array([0.1583,-0.6165,0.0606])#([0.1652,-0.6303,0.0633])
#Initial vector to optimize:
x0=np.array([TA_0,long_0,inc_0,ToF_0,dv1_0[0],dv1_0[1],dv1_0[2],dv2_0[0],dv2_0[1],dv2_0[2]])

#OPTIMIZATION

sol=minimize(dv_calc_LLO,x0,method='SLSQP',jac=dv_calc_LLO_GRAD,bounds=bnds,constraints={'type':
'ineq', 'fun': ineq_constraint,'type': 'eq', 'fun': get_ceq, 'jac':
get_ceq_grad},options={"ftol": 1e-6,"disp": True})
x_sol=sol.x

#TRAJECTORY AND CONSTRAINTS RECALCULATION
#We use the solution of the optimization to calculate the final optimized trajectory and
constraints values

p_trans,c_eq,c_eq_grad=traj_calc_LLO(x_sol)
print('eq_con:',c_eq)
ineq_con=ineq_constraint(x_sol)
print('ineq_con:',ineq_con)

#PLOT
#We plot the optimized trajectory just calculated

plot(p_trans)

#DELTA V RECALCULATION
#We use the solution of the optimization to calculate the value of the minimum delta V
```

```
DV=dv_calc_LLO(x_sol)
DV=DV*a_scale/T_scale
print('DV:',DV)

#FUL BURNT AND RESIDUAL PROPELLANT MASS
#I used the Tsiolkovsky rocket equation

#The propellant mass burnt for our DV is:
Fuel_burnt=Initial_total_mass*(1-(math.exp(-(DV/(Isp*g_0)))))
#Now we calculate the residual propellant mass:
Residual_fuel=float(myroot[47].text)-Fuel_burnt
print('Fuel burnt:',Fuel_burnt)

#PITCH AND YAW CALCULATION
#We operate a rotation between the reference frames to obtain pitch and yaw values (from CR3BP
reference frame to body reference frame)

Q=np.array([p_trans[-1,3]*a_scale/T_scale,p_trans[-1,4]*a_scale/T_scale,p_trans[-
1,5]*a_scale/T_scale])
Q_norm=Q/np.linalg.norm(Q)
if Q_norm[0]!=0 and Q_norm[1]!=0:
    pitch=math.degrees(math.atan2(Q_norm[0],Q_norm[1]))
else:
    pitch=0
if Q_norm[2]!=0:
    yaw=math.degrees(math.asin(Q_norm[2]))
else:
    yaw=0
print('Pitch:',pitch)
print('Yaw:',yaw)

#HOMOTOPY FILE UPDATE
#Now we have to modify the orbital parameters and the residual propellant values in the xml file
"homotopy"

#Inclination LLO update: 90 degrees (polar orbit):
for elem in myroot[4].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem.text = "90.0"
#Initial Arg of Periapsis (LLO) update: (assumed to be 0 for circular orbits):
for elem0 in myroot[5].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem0.text = "0.0"
#initial RAAN (LLO) update: (variable long):
if math.degrees(x_sol[1])<=360:
    RAAN=math.degrees(x_sol[1])
else:
    RAAN=math.degrees(x_sol[1])-360
for elem1 in myroot[6].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem1.text = "{:.1f}".format(RAAN)
#Initial True Anomaly (LLO) update: (variable inc, zero defined by the north pole):
```

```
if math.degrees(x_sol[2])<=360:
    True_Anomaly=math.degrees(x_sol[2])
else:
    True_Anomaly=math.degrees(x_sol[2])-360
for elem2 in myroot[7].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem2.text = "{:.1f}".format(True_Anomaly)
#Initial Periapsis (LLO) update:
for elem3 in myroot[9].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem3.text = "{:.1f}".format(h*a_scale/1000)
#Initial Apoapsis (LLO) update:
for elem4 in myroot[8].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem4.text = "{:.1f}".format(In_Apo)
#Residual fuel:
for elem5 in myroot[47].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem5.text = "{:.1f}".format(Residual_fuel)
#Pitch update:
for elem6 in myroot[21].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem6.text = "{:.1f}".format(pitch)
#Yaw update:
for elem7 in myroot[22].iter("{http://www.astos.de/schema/astos/9.17/scenario}Variable"):
    elem7.text = "{:.1f}".format(yaw)

mytree=ET.ElementTree(myroot)

mytree.write("homotopy.xml")
```

4.4 RISULTATI

In questa sezione verranno riportati i risultati ottenuti dall'esecuzione del codice.

Al termine dell'ottimizzazione le condizioni di uscita saranno le seguenti:

Valore corrente della funzione obiettivo	*0.6773944696138553*
Iterazioni	*100*
Valutazioni di funzione	*895*
Valutazioni del gradiente	*100*

Tabella 6 - Condizione di uscita dall'ottimizzazione

Le variabili assumeranno i seguenti valori soluzione:

T_A	*73087.0624 s*
RAAN	*1.8275 rad*
True Anomaly	*6.3594 rad*
ToF	*223425.7992 s*
Δv_1	*[-0.0089,0.0366,-0.0107]* $Km/_s$
Δv_2	*[0.1565,-0.6338,0.0522]* $Km/_s$

Tabella 7 - Valori soluzione delle variabili di ottimizzazione

È opportuno valutare il valore dei vincoli al termine dell'ottimizzazione per controllare che essi siano stati rispettati nella ricerca della soluzione di minimo. Gli equality constraints su posizione e velocità assumeranno i seguenti valori:

E_r	$[1.5992\ 10^{-5}, -6.3641\ 10^{-5}, 4.1737\ 10^{-6}]$
E_v	$[4.2774\ 10^{-4}, 5.4687\ 10^{-5}, -3.9507\ 10^{-3}]$

Tabella 8 - Equality constraints al termine dell'ottimizzazione

Come possiamo osservare, gli ordini di grandezza sono ancora piuttosto elevati. Ciò è dovuto innanzitutto alla selezione della funzione di ottimizzazione *"scipy.optimize.minimize"*. Essa ha il vantaggio di essere relativamente semplice e immediata da utilizzare, ma risulta purtroppo limitata rispetto ad altre metodologie più raffinate, ma anche più complesse. Ad alzare ulteriormente gli ordini di grandezza è inoltre il passo di integrazione utilizzato per computare la traiettoria. Rimpicciolendo ulteriormente il passo, infatti, si otterrebbero risultati migliori, ma l'esecuzione del codice risulterebbe eccessivamente lenta.
Riportiamo ora il valore degli equality constraints:

$-ToF + T_{max}$	*3528476.7908 s*

Tabella 9 - Inequality constraints al termine dell'ottimizzazione

Maggiore di zero, come dovrebbe essere.
Abbiamo riportato sopra il valore della funzione obiettivo Δv al termine dell'ottimizzazione. Tale valore è però ancora adimensionalizzato. Riportiamo quindi il valore finale di Δv ricondotto a dimensioni reali:

Δv	$694.0224\ {}^{m}/_{s}$

Tabella 10 - DV al termine dell'ottimizzazione

Tale valore di Δv richiede il consumo della seguente quantità di combustibile:

Fuel burnt	*6011.3051 Kg*

Tabella 11 - Combustibile necessario ad eseguire la manovra di trasferimento

Riproduciamo adesso la traiettoria compiuta dallo spacecraft che si muove dalla Near-Rectlininear Halo Orbit di partenza alla Low Lunar Orbit di arrivo:

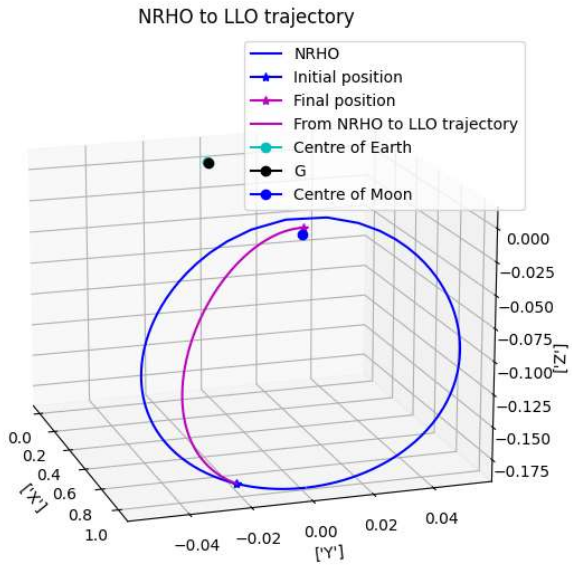

Figura 25 - NRHO-LLO trajectory (1)

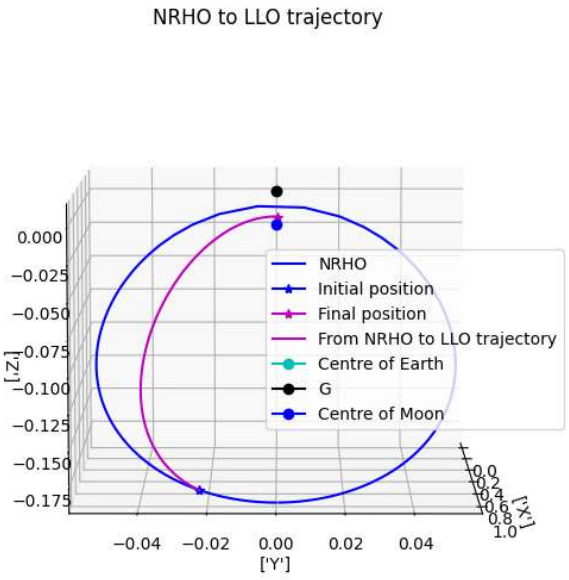

Figura 26 - NRHO-LLO trajectory (2)

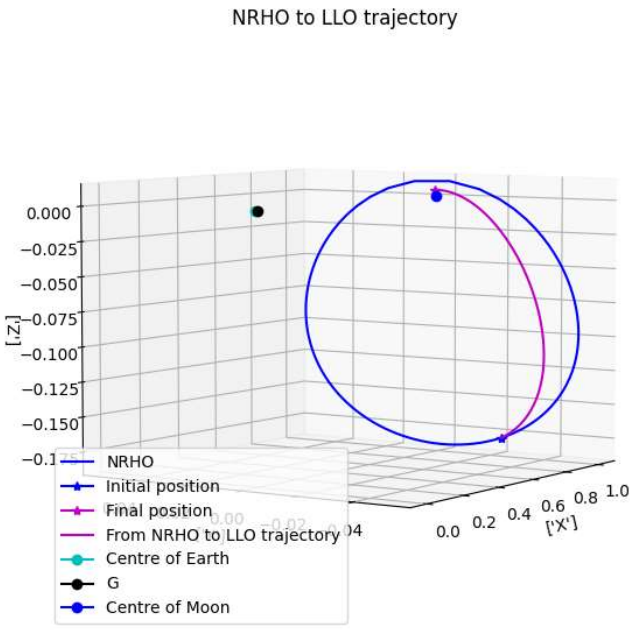

Figura 27 - NRHO-LLO trajectory (3)

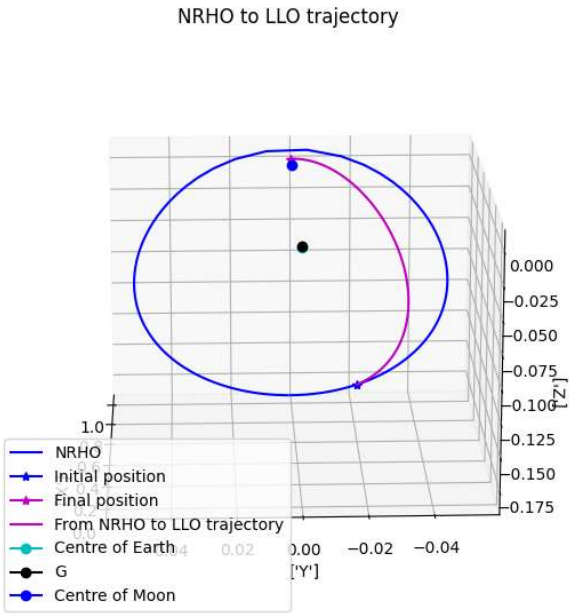

Figura 28 - NRHO-LLO trajectory (4)

La posizione del punto di arrivo verrà rappresentata dai seguenti parametri orbitali:

Inclinazione	*90°*
Argomento del Perigeo	*0°*
RAAN	*104.7°*
Anomalia Vera	*4.4°*
Apogeo	*101 Km*
Perigeo	*100 Km*

Tabella 12 - Parametri orbitali punto di arrivo

Riportiamo infine gli angoli di assetto del velivolo al punto di arrivo:

Pitch	*-14.7°*
Yaw	*-4.5°*

Tabella 13 - Pitch e Yaw al punto di arrivo

4.5 VALIDAZIONE

La validazione dei risultati, in termini di Δv minimo trovato tramite ottimizzazione, è effettuata grazie al confronto con i valori tipici trovati in letteratura. In particolare i valori riportati sotto si riferiscono a studi eseguiti e pubblicati da alcuni ricercatori NASA.

Il ΔV necessario ad eseguire una manovra di trasferimento da una Near-Rectilinear Halo Orbit ad una Low Lunar Orbit è indicato come:

$$\Delta V = 730\ {}^{m}/_{s}$$

Il carburante necessario ad eseguire tale manovra sarà dunque:

$$Fuel = 6298.0220\ Kg$$

Abbiamo quindi un discostamento compreso tra il 4-5% del valore indicato di ΔV e combustibile consumato rispetto ai valori trovati tramite lo script. Il margine può essere ritenuto accettabile se si considerano i limiti del metodo di ottimizzazione utilizzato e l'ampio passo di integrazione impostato.

[credits: [3]]

5. SIMULAZIONE COMPLETA IN ASTOS

Come già affermato in precedenza, i risultati provenienti dall'algoritmo in Python verranno salvati nel file *"homotopy.xml"*. Il file viene poi utilizzato dallo scenario di missione impostato in Astos per compiere la simulazione della seconda parte della traiettoria, dalla Low Lunar Orbit alla superficie lunare. L'interfaccia è manuale. Il file viene modificato automaticamente tramite i comandi stabiliti all'interno del codice Python e successivamente deve essere spostato manualmente dall'utente nella sottocartella *"bach"* all'interno della cartella dello scenario. Si apre quindi lo scenario in ambiente Astos e si effettua la simulazione. Si valuta dunque la convergenza della simulazione e la sua coerenza con i risultati attesi. Vengono perciò riportati di seguito i risultati ottenuti in termini di:

- Altitudine:

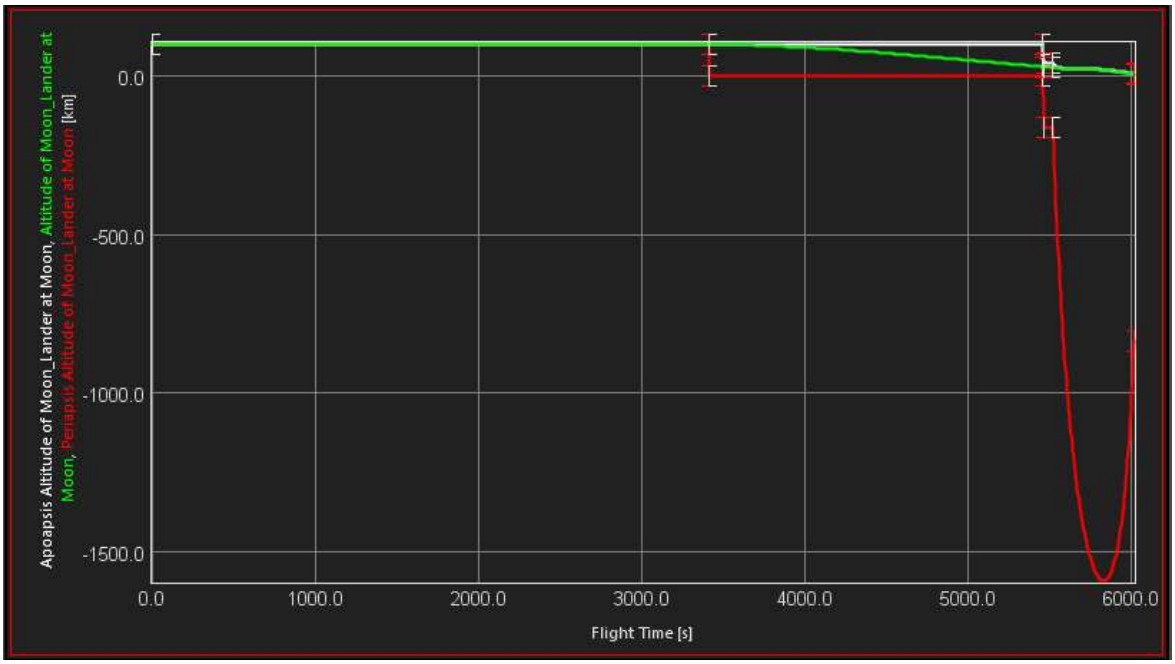

Figura 29 - Risultati Astos: Altitudine

- Angoli di Yaw e Pitch:

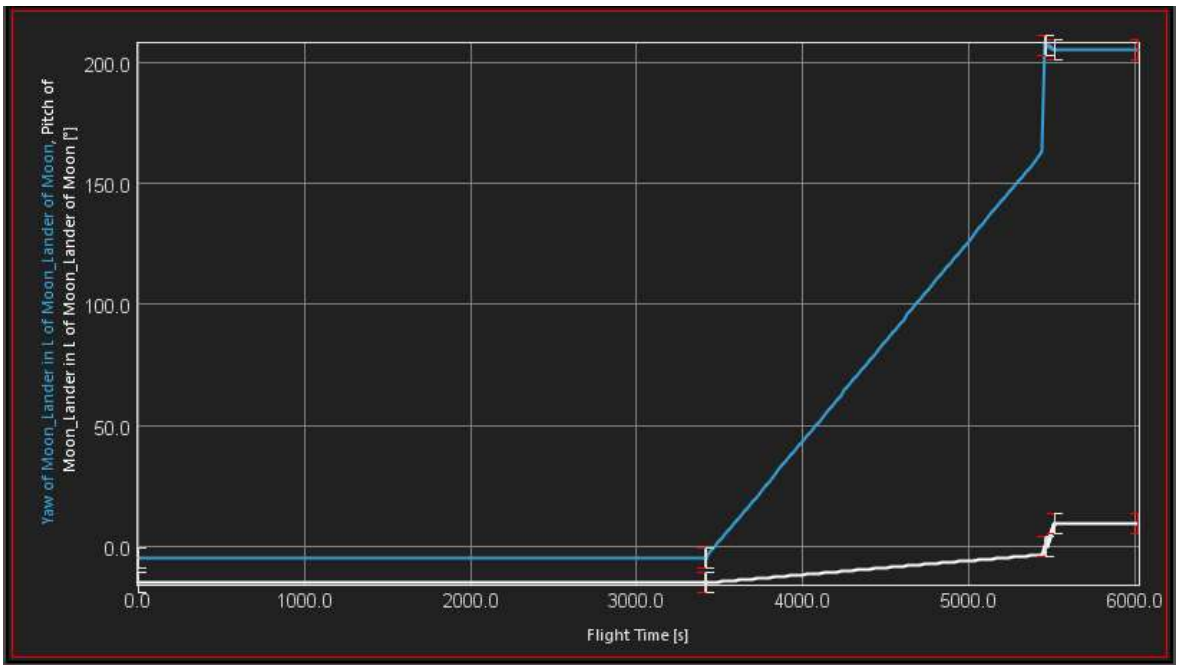

Figura 30 - Risultati Astos: Yaw&Pitch

- Velocità:

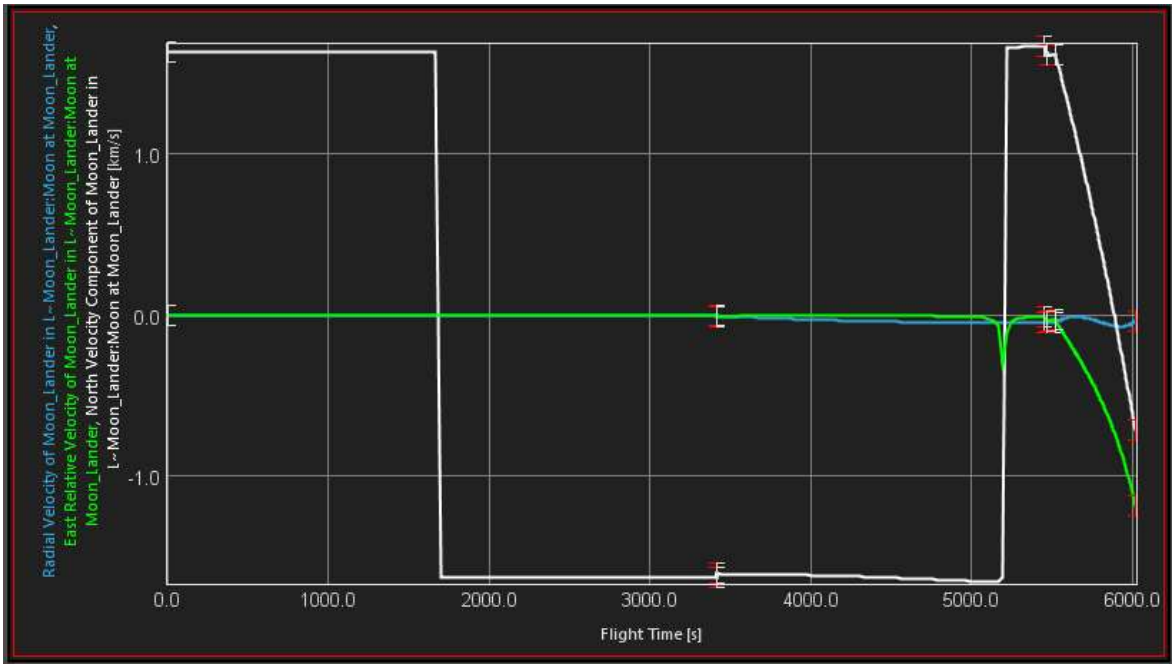

Figura 31 - Risultati Astos: Velocità

- True Anomaly:

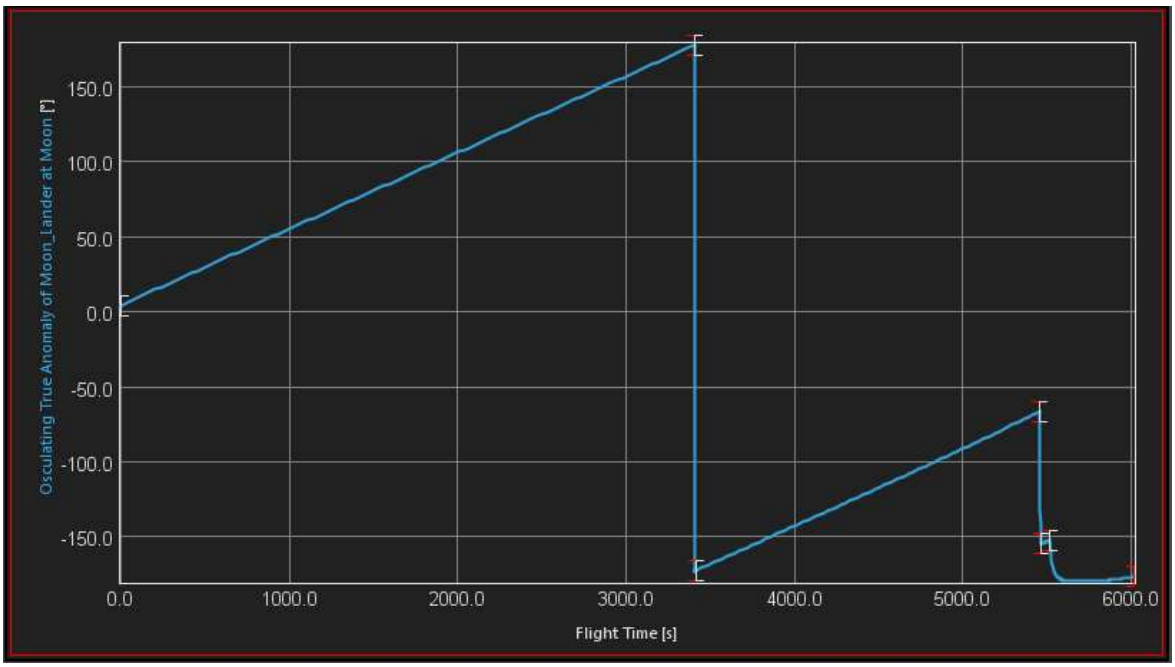

Figura 32 - Risultati Astos: True Anomaly

- Massa di Propellente:

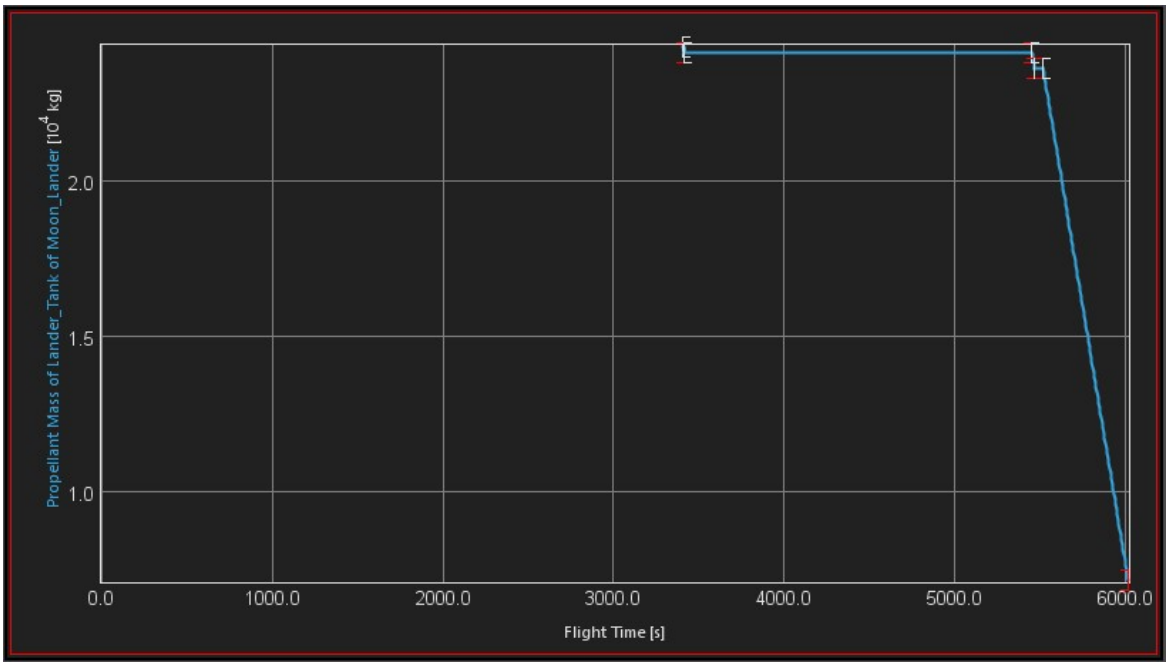

Figura 33 - Risultati Astos: Propellente

- Visualizzazione 3D:

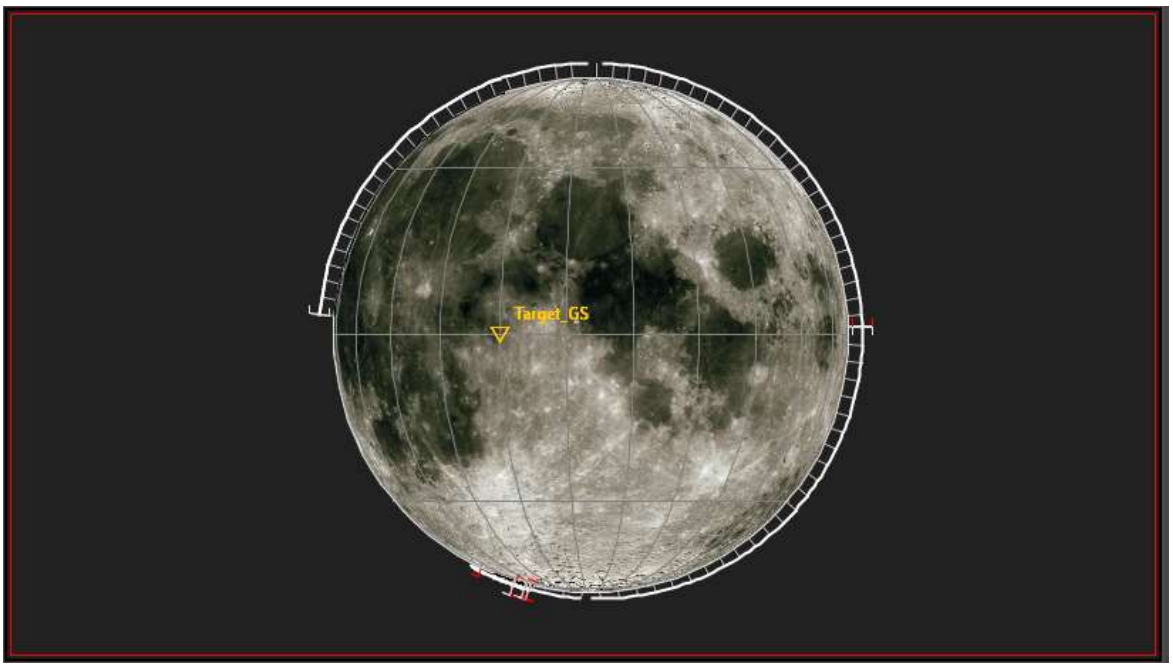

Figura 34 - Risultati Astos: 3D Map

- Ground Track:

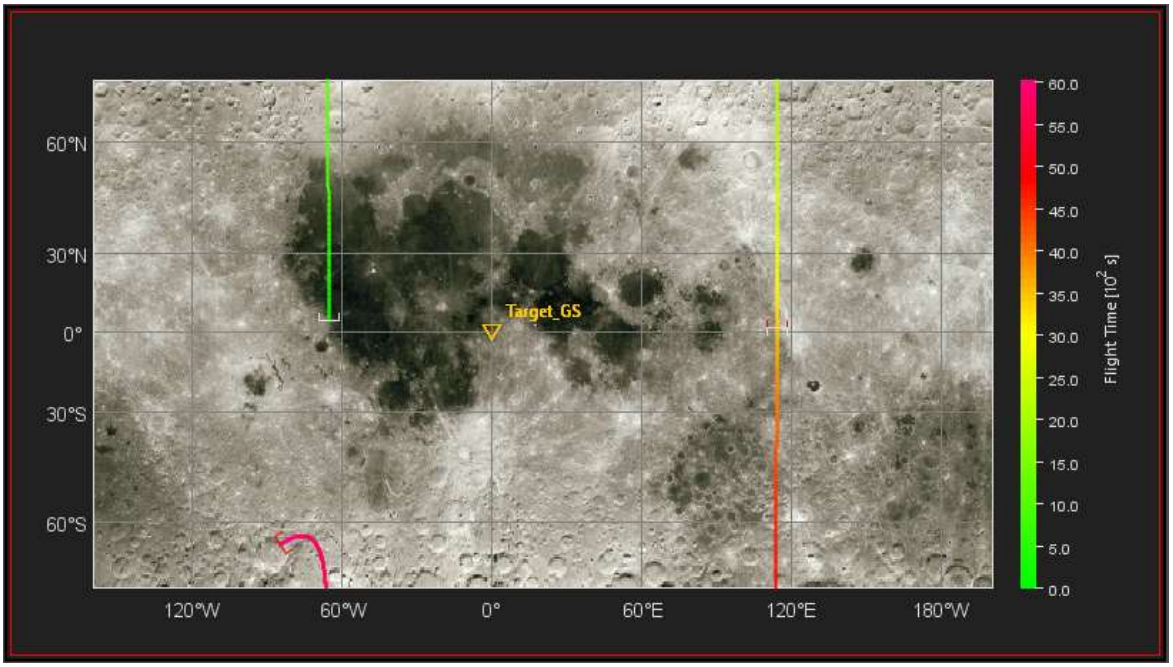

Figura 35 - Risultati Astos: Ground Track

- ΔV:

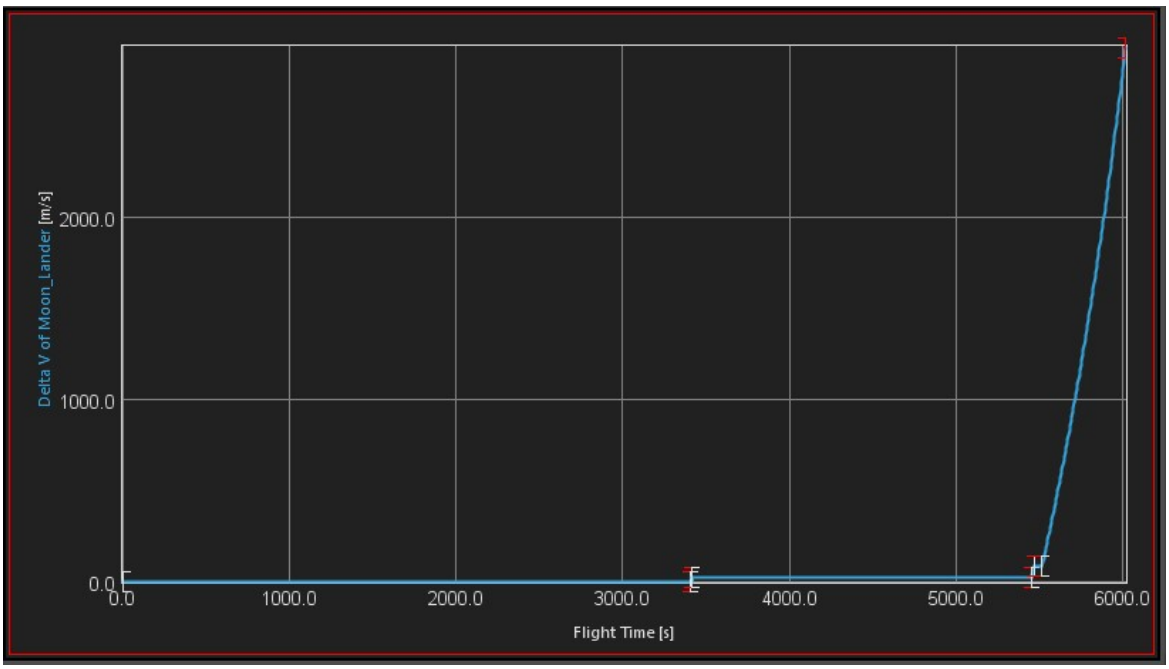

Figura 36 - Risultati Astos: DV

- Latitudine e Longitudine:

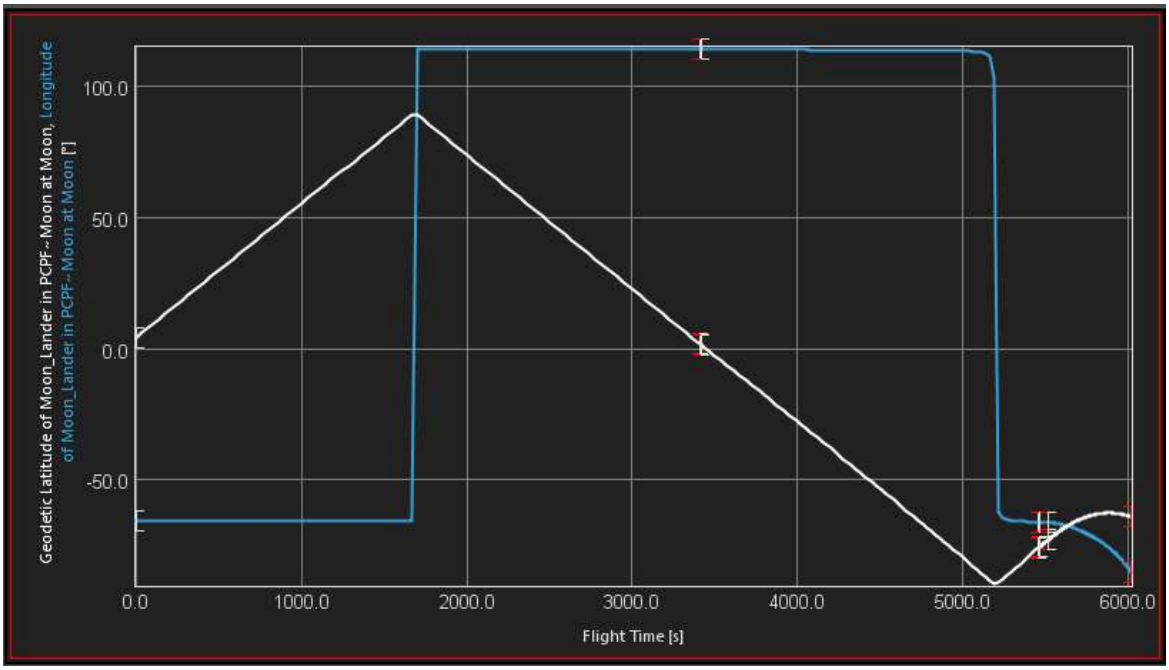

Figura 37 - Risultati Astos: Latitudine&Longitudine

- Eclipse Time:

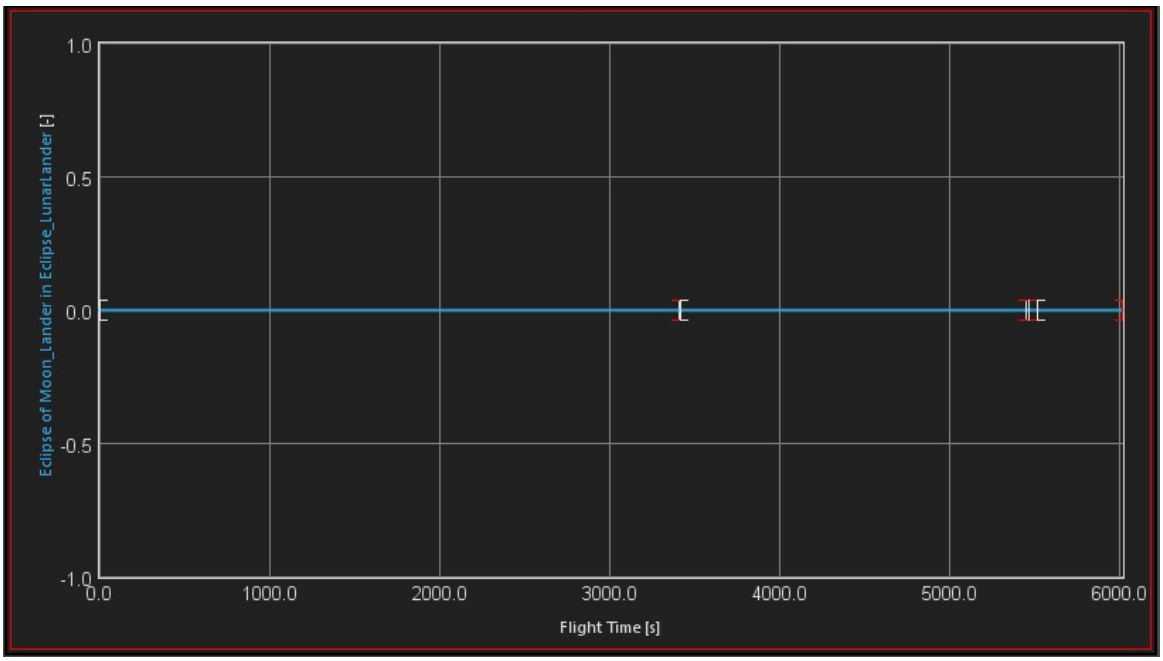

Figura 38 - Risultati Astos: Eclipse Time

I risultati mostrati appaiono coerenti con quanto atteso.

Per completezza di esposizione riportiamo inoltre alcuni frame della traiettoria di discesa sulla superficie lunare compiuta dal lander e visibile nella sezione di Astos Astroview:

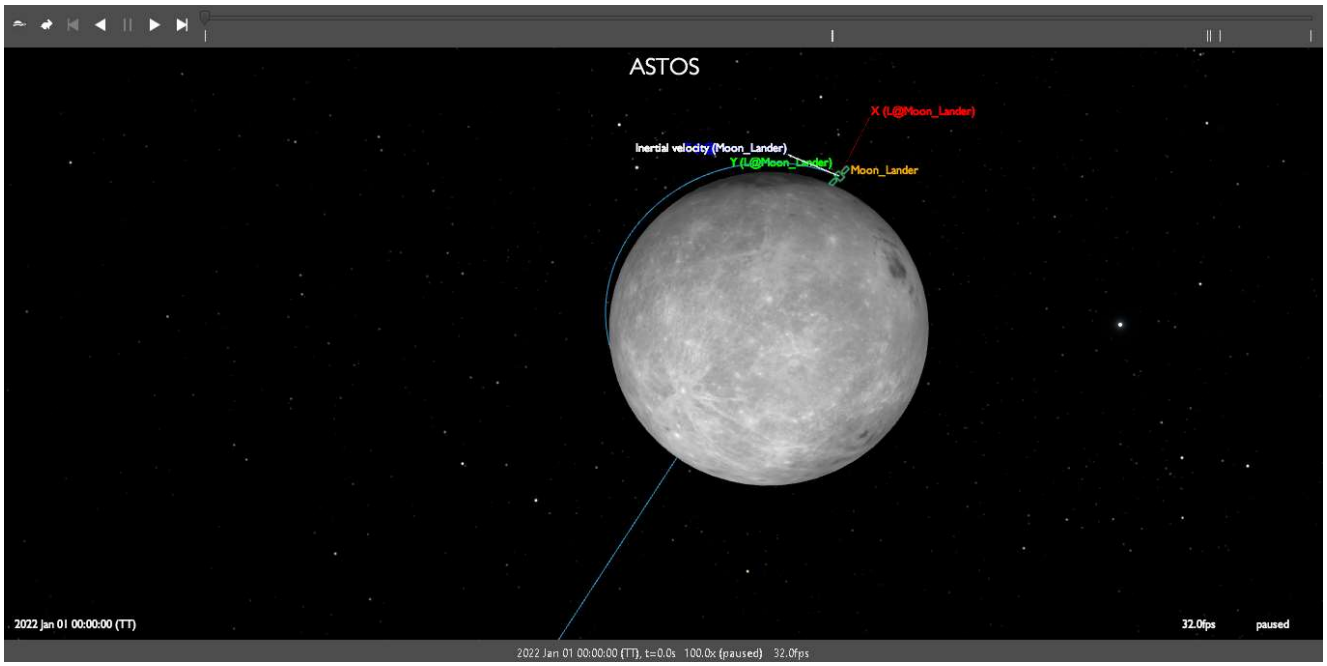

Figura 39 - Risultati Astos: Astroview (1)

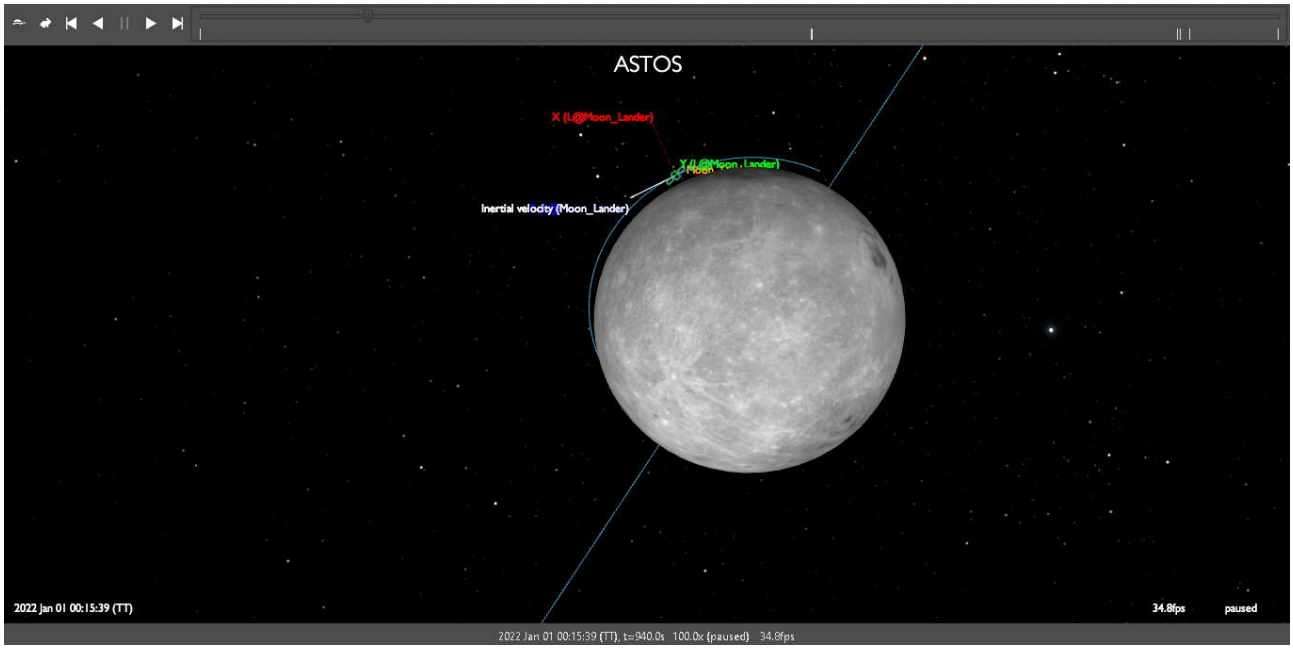

Figura 40 - Risultati Astos: Astroview (2)

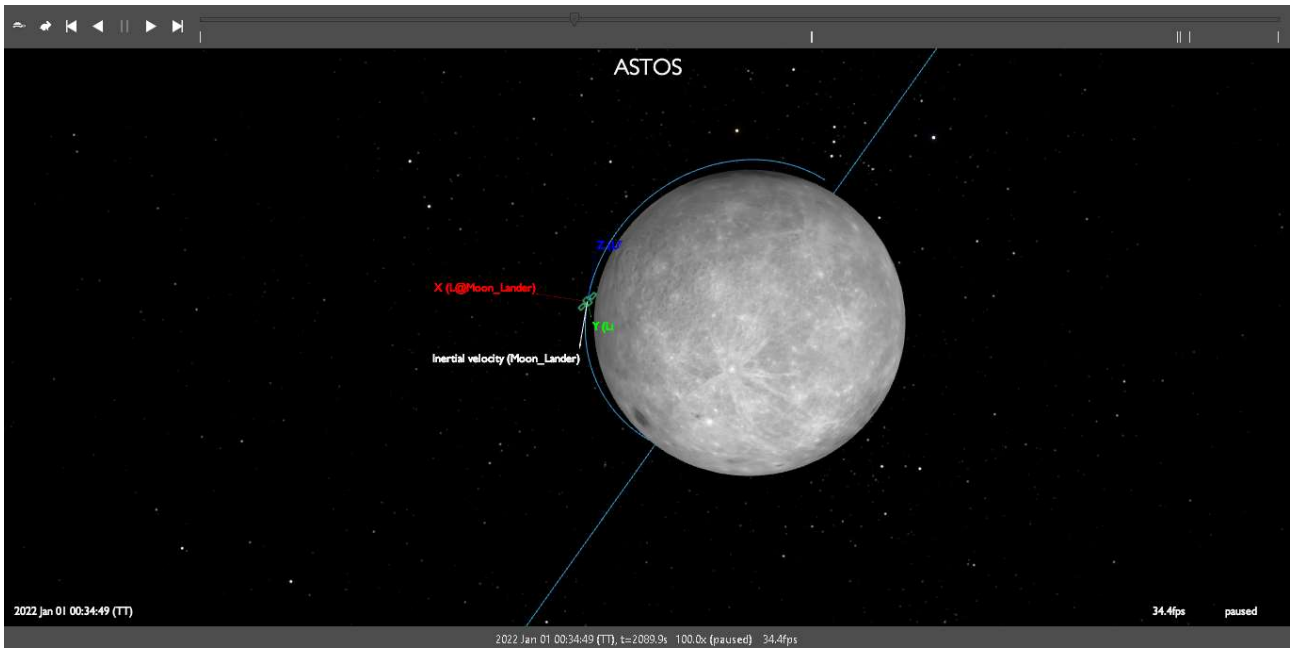

Figura 41 - Risultati Astos: Astroview (3)

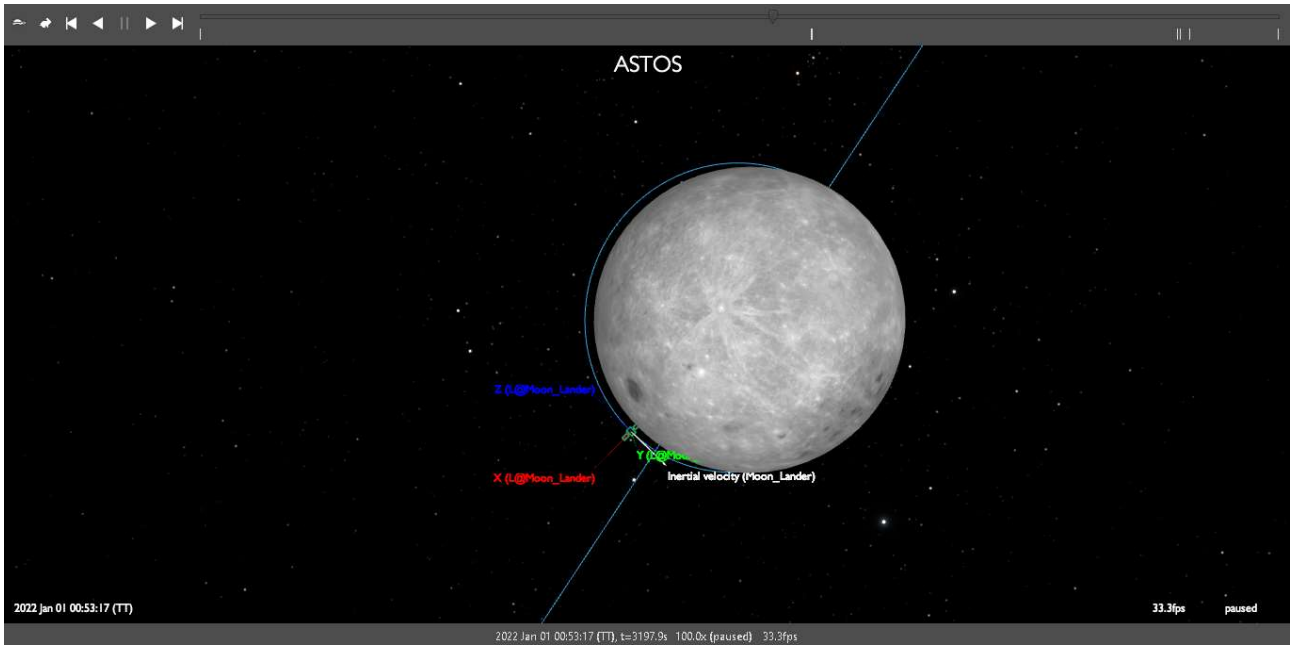

Figura 42 - Risultati Astos: Astroview (4)

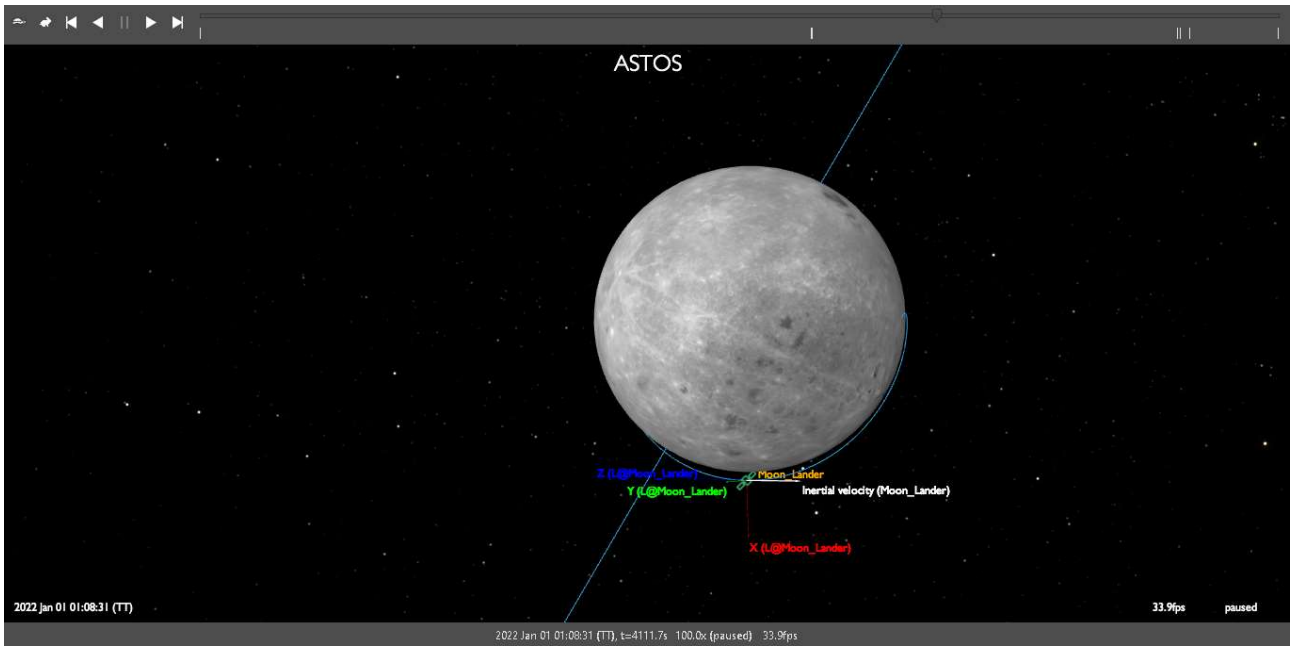

Figura 43 - Risultati Astos: Astroview (5)

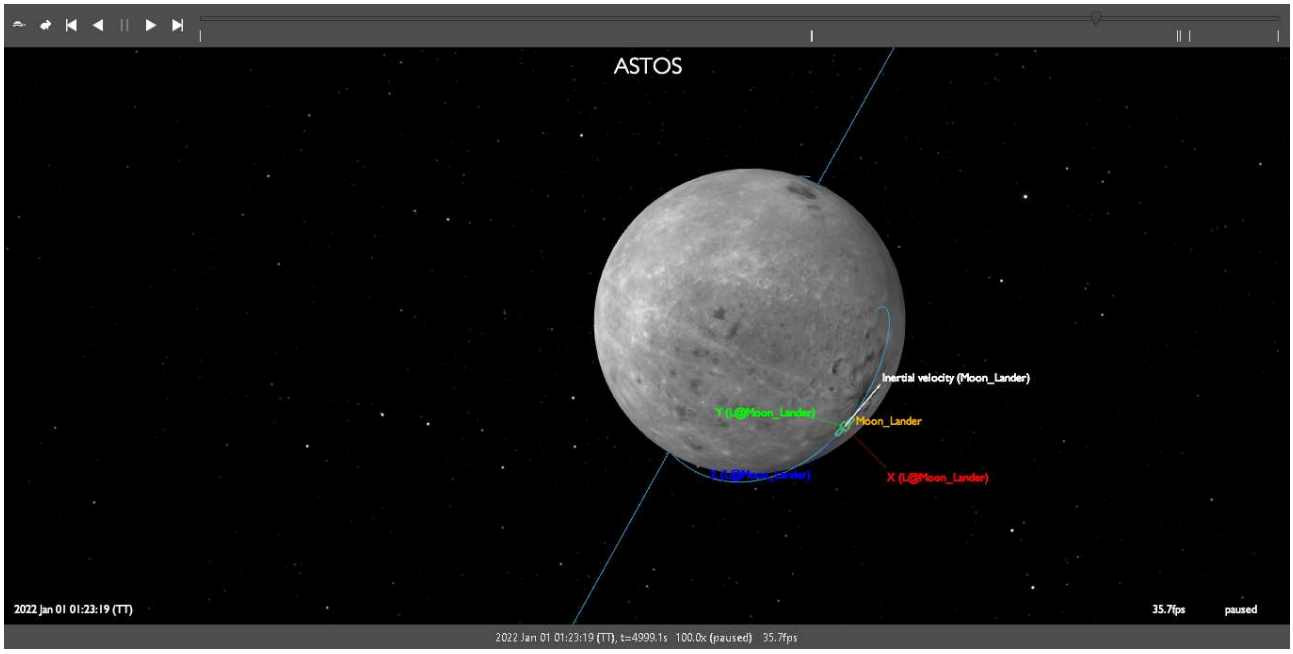

Figura 44 - Risultati Astos: Astroview (6)

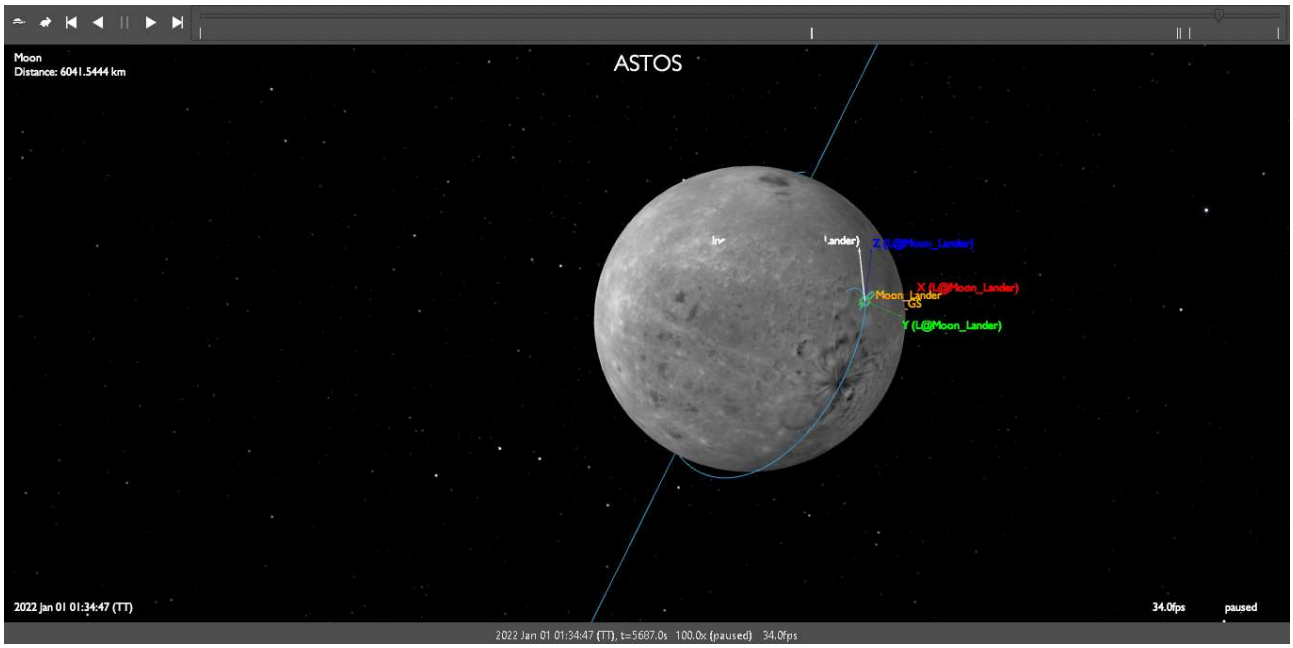

Figura 45 - Risultati Astos: Astroview (7)

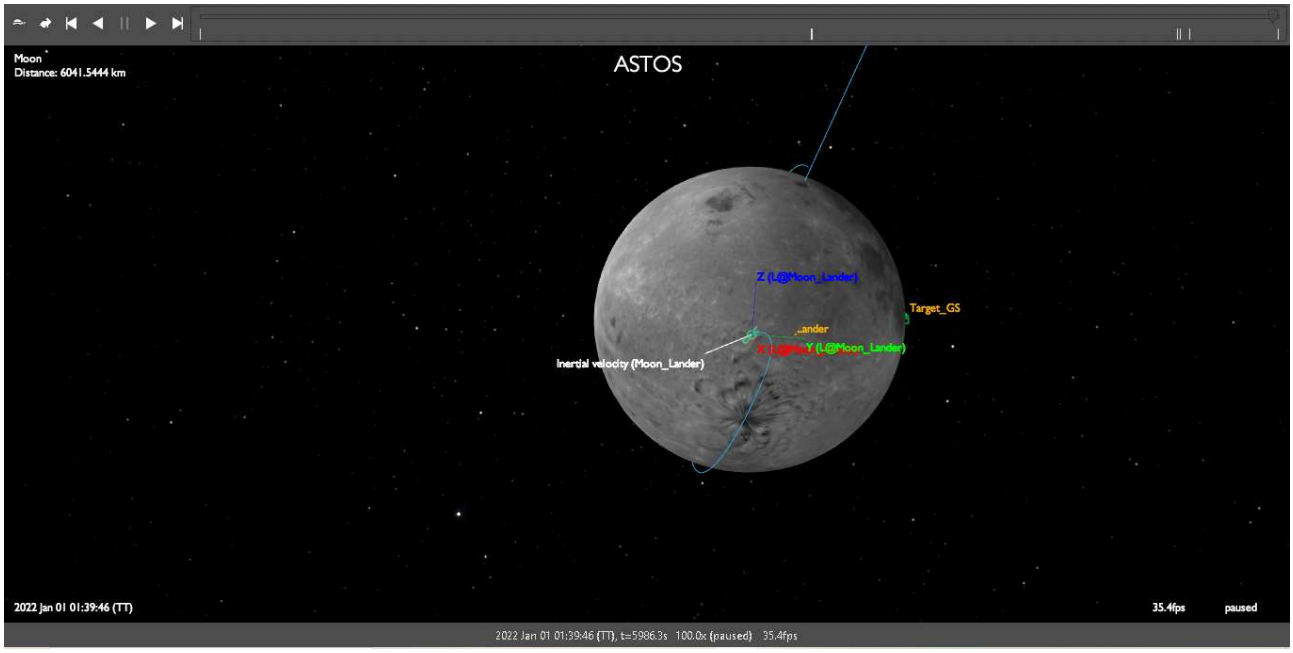

Figura 46 - Risultati Astos: Astroview (8)

6. CONCLUSIONI

Il panorama futuro dell'esplorazione spaziale è focalizzato sullo sfruttamento del nostro satellite naturale, come fonte di risorse grazie alle riserve di ghiaccio trovate ai suoi poli, e come punto intermedio per missioni più lunghe. Parte fondamentale di tale visione è la Lunar Orbital Platform-Gateway, la stazione spaziale orbitante in ambiente cislunare, un ponte tra l'orbita terrestre e la superficie lunare. In questo contesto diventa indispensabile la definizione di traiettorie ottimali per il trasferimento tra la NRHO, sulla quale orbita la LOP-G, e la Low Lunar Orbit da cui avverrà la discesa verso la superficie lunare. Lo scopo di questa tesi è stato quello di porre le basi per la generazione di un tool che permetta la valutazione preliminare della suddetta traiettoria di trasferimento. Dopo un excursus sulla storia e sulle prospettive attuali del campo spaziale, i capitoli 2 e 3 si sono posti l'obiettivo di sviscerare i fondamenti di meccanica orbitale e teoria dell'ottimizzazione utilizzati per l'esecuzione del progetto. Il capitolo successivo ha quindi analizzato lo svolgimento del lavoro di tesi. In primo luogo è stato quindi definito il problema, descrivendo le caratteristiche del sistema Terra-Luna nel quale si muove il nostro spacecraft e delle orbite di partenza e arrivo della manovra di trasferimento esaminata e modellando il problema di ottimizzazione con relativi initial guesses, constraints, bounds e gradienti. È stata perciò approfondita l'implementazione, con linguaggio di programmazione Python, del modello matematico definito nei paragrafi precedenti. Ne abbiamo ricavato innanzitutto il valore minimizzato del ΔV propulsivo, utilizzato poi per ricavare il propellente necessario ad effettuare il trasferimento. Abbiamo poi calcolato e tracciato la traiettoria corrispondente. Abbiamo potuto definire i parametri orbitali del punto di arrivo sull'orbita finale e infine gli angoli di pitch e yaw assunti dal lander al termine della manovra. I valori ottenuti sono stati validati tramite confronto con i valori tipici riportati dalla letteratura. Sono stati inoltre testati in una simulazione completa comprendente la fase di missione di discesa sulla superficie lunare svolta in Astos. Il progetto portato avanti, pur rappresentando un buon punto di partenza, presenta alcune limitazioni. In prima battuta è importante notare come il tool realizzato abbia l'obiettivo di poter essere riutilizzato per diversi input. L'accuratezza della funzione di ottimizzazione utilizzata però, pur presentando il vantaggio di essere molto immediata nell'utilizzo, è modesta. Le condizioni iniziali di ottimizzazione dovranno essere estremamente precise. Automatizzando invece il processo e non valutandole per ogni input selezionato, si rischia di abbassare ancora di più l'accuratezza dell'ottimizzazione. I limiti della funzione di ottimizzazione incidono anche sull'eccessivo ordine di grandezza dei vincoli di equality risultanti al termine

della ricerca del minimo. Per ovviare a tali problematiche ci si propone per i lavori futuri l'utilizzo di pacchetti matematici più complessi, ma maggiormente accurati, come Gekko o Mystic. Un'altra problematica che incide ulteriormente sull'accuratezza dei risultati è il passo di integrazione per la computazione della traiettoria. Per ottenere una maggiore precisione è necessario una aggiuntiva diminuzione dello stesso, rallentando però sproporzionatamente l'esecuzione del programma. Per risolvere questa problematica si potrebbero perciò ricercare metodologie di integrazione più raffinate.

7. LISTA DELLE FIGURE

8. LISTA DELLE TABELLE

9. BIBLIOGRAFIA

[1] *Narducci G., Chirulli D., Vincelli F., Ferretto D., Fusaro R., Technical note, 2021*

[2] *Curtis H. D., Orbital Mechanics for Engineering Students, Butterworth-Heinemann, 2021, fourth edition*

[3] *Whitley R., Martinez R., Options for Staging Orbits in Cis-Lunar Space, in: "NASA Technical Reports Server, Conference Paper, 2015*

[4] *Mastrodonato S., Rimozione di debris multipli in LEO: applicazione della teoria dei grafi ad un problema tempo – dipendente, Politecnico di Torino, 2020/2021, Rel. Casalino L.*

[5] *Betts J. T., Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Society for Industrial and Applied Mathematics, Philadelphia, 2010*

[6] *Betts J. T., Survey of Numerical Methods for Trajectory Optimization, in "Journal of Guidance, Control and Dynamics", Vol. 21, No. 2, March-April 1998*

[7] *Betts J. T., Huffman W. P., Path-Constrained Trajectory Optimization Using Sparse Sequential Quadratic Programming, in "Journal of Guidance, Control and Dynamics", Vol. 16, No. 1, January-February 1993*

[8] *Savaglia N., Design and Optimization of Low Thrust Maneuvers in Interplanetary Missions, Politecnico di Torino, March 2018, Rel. Casalino L.*

[9] *Corigliano N., Orbite e Parametri Orbitali, Progetto EduSAT, ITIS "G. Marconi" – Bari / ASI / IMT*

[10] *Beauregard L., Approche systématique de la conception d'un véhicule d'ascension operant entre la surface de la Lune et une station spatiale en orbite cis-lunaire, Université De Toulouse, July 2021, Rel. Morlier J. and Lizy-Destrez S.*

[11] *International Space Exploration Coordination Group. The Global Exploration Roadmap, January 2018. url: https://www.nasa.gov/sites/default/files/atoms/files/ger_2018_small_mobile.pdf*

[12] *Li S., Lucey P. G., Milliken R. E., Hayne P. O., Fisher E., Williams J., Hurley D. M., Elphic R. C., Direct evidence of surface exposed water ice in the lunar polar regions, in: "Proceedings of the National Academy of Sciences", 115.36 (2018), pp. 8907–8912*

[13] *Zini E. E., Precise orbit determination techniques for a lunar satellite navigation system, Politecnico di Torino, 2020/2021, Rel. Dovis F.*

[14] *Grammatico M., Fuel-Optimal Lander Trajectory for Lunar Soft-Precision Landing. Optimal Control Theory, Politecnico di Torino, 2020/2021, Rel. Casalino L.*

[15] *Feldman W. C., Maurice S., Binder A. B., Barraclough B. L., Elphic R. C., Lawrence D. J., Fluxes of fast and epithermal neutrons from Lunar Prospector: Evidence for water ice at the lunar poles, in: "Science", 281.5382 (1998), pp. 1496–1500*

[16] Schonfeldt M., Grenier A., DelEpaut A., Giordano P., Swinden R., Ventura-Traveset J., Across the Lunar Land-scape: Towards a Dedicated Lunar PNT System, in: "Inside GNSS", 2020

[17] Drake N., Howard J., A brief history of moon exploration, in: "www.nationalgeographic.co.uk", 20 July 2020. url: https://www.nationalgeographic.co.uk/space/2020/07/a-brief-history-of-moon-exploration

[18] Manned Spacecraft Center, Houston TX, NASA,National Aeronautics and Space Administration, Apollo 11 Fligh Plan, 1969. url: https://www.hq.nasa.gov/alsj/a11/a11fltpln_final_reformat.pdf

[19] Manned Spacecraft Center, Houston TX, NASA, National Aeronautics and Space Administration, Apollo Lunar Landing Mission Symposium, 1966. url: https://www.hq.nasa.gov/alsj/LunarLandingMIssionSymposium1966_1978075303.pdf

[20] Burke J. D., Chapter 56 - Planetary Exploration Missions, in "Encyclopedia of the Solar System", pages 1205–1222, Elsevier, Boston, third edition, 2014

[21] Crawford I. A., Joy K. H., Anand M., Chapter 25 - Lunar Exploration, in "Encyclopedia of the Solar System", pages 555–579, Elsevier, Boston, third edition, 2014

[22] ESA website - Exploration of the Moon. url: https://exploration.esa.int/web/moon

[23] NASA website – Apollo 15 Gallery. url: https://www.nasa.gov/feature/50-years-ago-apollo-15-on-the-moon-at-hadley-apennine

[24] Harland D., Exploring the Moon: The Apollo Expeditions, Springer Praxis Pub, New York, London, Chichester, UK, 2008

[25] Orloff R. W., Harland D., Apollo: The Definitive Sourcebook, Springer, New York, 2006

[26] Bignami L, Le prime immagini dal suolo lunare, 50 anni fa, in: "www.focus.it", 4 February 2016. url: https://www.focus.it/scienza/spazio/cinquanta-anni-fa-le-prime-immagini-dal-suolo-lunare

[27] NASA website – Apollo 11 Gallery. url: https://www.nasa.gov/apollo11-gallery

[28] Gualandi S., Metodi di Quasi Newton, Università di Pavia, Dipartimento di Matematica

[29] NASA website – Gateway. url: https://www.nasa.gov/gateway/overview

[30] Princi A., Deep learning techniques for micro-launchers branching trajectories optimization, Politecnico di Torino, 2020/2021, Rel. Viola N. and Rimani J.

[31] Curry A., Haskell B., The Method of Steepest Descent for Non-linear Minimization Problems, Quart. Appl. Math. 2

[32] Wikipedia website. url: https://en.wikipedia.org/wiki/Gradient_descent

[33] NASA website – Benson T., Ideal Rocket Equation. url: https://www.grc.nasa.gov/WWW/K-12/rocket/rktpow.html

[34] Bucchioni G., Innocenti M., Phasing Maneuver Analysis from a Low Lunar Orbit to a Near Rectilinear Halo Orbit, in: "Aerospace" 2021, 8, 70. url: https://www.mdpi.com/journal/aerospace

[35] ESA website – Enabling Support – Angelic halo orbit chosen for humankind's first lunar outpost, 18 July 2019. url: https://www.esa.int/Enabling_Support/Operations/Angelic_halo_orbit_chosen_for_humankind_s_first_lunar_outpost