ALMA MATER STUDIORUM-Università di Bologna

Corso di Laurea in Ingegneria Aerospaziale

A.A. 2019/2020

Elaborato di Tesi Finale

svolto presso:

Laboratorio di Radioscienza ed Esplorazione Planetaria dell'Università di Bologna

IMPLEMENTAZIONE DI UN MODELLO DI PRESSIONE DI RADIAZIONE SOLARE ATTRAVERSO UN ALGORITMO DI RAY-TRACING

Presentata da Federico Pagno

Relatore Prof. Paolo Tortora

Indice

LISTA FIGURE

LISTA TABELLE

ACRONIMI

NASA	National Aeronautics and Space Administration
GMAT	General Mission Analysis Tool
MONTE	Mission Analysis, Operations and Navigation Toolkit Environment
NAIF	Navigation and Ancilary Information Facility
S/C	Spacecraft
S.d.R.	Sistema di riferimento
C/S	CubeSat
PS	Pannello solare

Capitolo 1 : Introduzione

Nel processo di analisi e sviluppo di una missione spaziale, la progettazione e la determinazione della traiettoria che dovrà seguire il segmento spaziale al termine del lancio, è un passaggio di fondamentale importanza per il successo della missione stessa.

Per prevedere la traiettoria di un corpo nello spazio, è necessario avere una conoscenza precisa delle forze agenti sul corpo. L'errore che si commette nel calcolo della traiettoria dipende dall'accuratezza con cui vengono modellizzate tali forze.

In ambiente spaziale le forze di maggiore intensità (che in prima approssimazione determinano la dinamica del satellite) sono le forze gravitazionali esercitate da corpi celesti dotati di massa non trascurabile.

Tuttavia, per prevedere con accuratezza la dinamica della traiettoria del satellite, è necessario considerare anche le forze di origine non gravitazionale, che tendono a perturbare la traiettoria, a volte anche in modo significativo.

Un esempio di perturbazione di origine non gravitazionale, di notevole importanza, è la Solar Radiation Pressure (SRP) ovvero la pressione di radiazione solare. Tale forza ha origine quando una sorgente luminosa (ad esempio il Sole) illumina le superfici del satellite.

Nel Capitolo 2 verrà approfondito il concetto di SRP e, nei capitoli successivi, verranno proposte alcune strategie di modellazione analitica e numerica utilizzate nella propagazione di un'orbita.

La traiettoria di un corpo celeste nello spazio viene propagata a partire dal suo stato iniziale, ovvero la coppia di vettori $\vec{R}_0$ (raggio allo stato iniziale) e $\vec{V}_0$ (velocità allo stato iniziale) espressi in determinato sistema di riferimento (S.d.R.) e viene integrata tenendo conto delle forze (gravitazionali e non gravitazionali) agenti su di esso.

1.1 Software e strumenti utilizzati

Dal punto di vista operativo, il calcolo della traiettoria di un satellite nello spazio si effettua utilizzando un software d'integrazione numerica che propaga l'orbita sulla base dei modelli dinamici delle forze considerate e sulle condizioni iniziali fornite in ingresso.

Il software di propagazione orbitale utilizzato nel corso del lavoro di tesi è GMAT (General Mission Analysis Tool).

GMAT è un software open source per la progettazione e l'ottimizzazione di missioni spaziali che supporta lo sviluppo di missioni in vari regimi di volo, dall'orbita bassa terrestre alle missioni deep-space. GMAT è nato da una collaborazione della National Aeronautics and Space Administration (NASA) ed industrie private specializzate nel settore ed è uno strumento che viene utilizzato in varie fasi di analisi di missione reali, studi ingegneristici ed anche come strumento didattico [7].

GMAT permette di utilizzare modelli dinamici molto accurati e mette a disposizione dell'utente un'infrastruttura di scripting e programmazione integrata con una sintassi simile a quella di MATLAB. GMAT dispone inoltre di un sistema di visualizzazione 3D delle traiettorie ed un sistema di reportistica ed output dati personalizzabile (rapidamente importabili anche in MATLAB per un'analisi avanzata) [7].

In GMAT, l'accelerazione dovuta alla SRP può essere considerata nella propagazione utilizzando i seguenti modelli:

- Modello sferico standard (detto anche "Cannonball").
- ModelloSPAD (Solar Pressure and Aerodynamic Drag), dove i parametri di implementazione devono essere forniti dall'utente attraverso un file specifico.

Al fine di validare gli algoritmi utilizzati per l'implementazione del modello SPAD sono state effettuate delle comparazioni con delle traiettorie propagate con il software MONTE (Mission analysis, Operations, and Navigation Toolkit Environment).

MONTE è una libreria Python sviluppata nei laboratori del JPL (Jet Propulsion Laboratory) a Pasadena (California) con lo scopo di realizzare una piattaforma di

elaborazione astrodinamica che supporti tutte le fasi dello sviluppo della missione spaziale, dall'analisi di missione alla navigazione operativa. MONTE offre una piattaforma su cui gli utenti possono costruire i propri strumenti aerospaziali in modo personalizzato e fornisce inoltre un'infrastruttura astrodinamica di base: modelli di traiettoria, sistemi di riferimento, tempo ad alta precisione, ricerca di eventi astrodinamici, analisi di sensibilità, integratori numerici, ottimizzazione, ecc [3].

MONTE, per il calcolo di SRP e forza di attrito aerodinamica, consente di descrivere geometrie complesse per il satellite come composizione di pannelli elementari o come composizione di geometrie tridimensionali elementari (cilindri, coni, cubi o paraboloidi), aumentando quindi l'accuratezza rispetto ad un modello sferico standard.

La motivazione per cui sono stati confrontati i risultati dei due diversi software è quella di simulare una possibile situazione operativa dove, durante della progettazione di una missione spaziale, due team diversi (ad esempio il team di analisi di missione ed il team di navigazione), che utilizzano due sistemi diversi (ad esempio GMAT e MONTE), trovano risultati che non sono esattamente uguali fra loro.

Infatti, è necessario specificare che GMAT e MONTE sono due software significativamente diversi fra loro: GMAT si utilizza principalmente per l'analisi di missione mentre MONTE, che è uno strumento con diverse funzionalità in più, viene impiegato anche per la navigazione. Di conseguenza, è comprensibile che nel confronto dei risultati ottenuti con i due software sia presente un potenziale errore di modello che introduce quindi un'incertezza nella propagazione della traiettoria.

Per implementare al calcolatore le funzioni sviluppate in questa tesi per poter generare un file di modello SPAD è stato utilizzato il software MATLAB. In particolare, per implementare la funzione *SPADcomplex* (Files C), che verrà poi descritta nel Capitolo 4 , è stato utilizzato uno script di ray-tracing semplificato, basato sull'algoritmo di Möller-Trumbore (MT), ottenuto dal MATLAB Files Exchange Center [9].

In generale, il ray-tracing è una tecnica di geometria ottica che si basa sul calcolo del percorso fatto da un raggio di luce e della sua interazione con le superfici.

Nel caso specifico di questa tesi, l'algoritmo (descritto al paragrafo 4.1) è stato utilizzato per trovare l'intersezione tra un raggio di luce e la superfice esterna del satellite considerato.

MATLAB è stato inoltre utilizzato come ambiente di analisi, elaborazione e confronto delle traiettorie propagate da GMAT e MONTE utilizzando le librerie SPICE (NAIF) dedicate.

"SPICE" è uno strumento sviluppato dal Navigation and Ancilary Information Facility (NAIF) della NASA per assistere gli scienziati nella pianificazione e nell'interpretazione delle osservazioni scientifiche da strumenti spaziali e per assistere gli ingegneri nella modellazione, pianificazione ed esecuzione delle attività necessarie per condurre missioni di esplorazione planetaria. L'uso di SPICE si estende solitamente dallo sviluppo del concept di missione fino alla fase di analisi dei dati post-missione [2].

1.2 Motivazione ed obiettivi

La motivazione principale di questo lavoro di tesi si basa sull'utilizzo del software GMAT per la propagazione accurata di traiettorie spaziali considerando anche il disturbo relativo alla SRP utilizzando il modello SPAD.

L'obiettivo si è quindi concretizzato con l'implementazione di un algoritmo in MATLAB in grado di generare un file di modello SPAD, sia per casi elementari come il modello sferico che per satelliti di geometria complessa.

In particolare, al fine di sperimentare gli strumenti realizzati su una missione reale, è stato esaminato il caso di un CubeSat da sei unità (6U) che percorre una traiettoria particolarmente sensibile alla SRP.

Capitolo 2 : Introduzione alla SRP

La pressione della radiazione solare (SRP) è la pressione prodotta su di un corpo da parte di una radiazione elettromagnetica che incide su di esso. La SRP è dovuta al trasferimento di quantità di moto che avviene tra l'onda elettromagnetica e la superficie su cui essa incide.

Considerando ad esempio il Sole come sorgente luminosa, si ha che quando i raggi di luce dai lui prodotti intercettano, ad esempio, la superficie di un satellite, si sviluppa un'accelerazione che produce quindi una perturbazione sulla traiettoria di quest'ultimo.

Nel caso di un satellite, tale disturbo è generalmente di ordine inferiore rispetto alle forze gravitazionali, tuttavia per progettare accuratamente una traiettoria è necessario considerarlo.

In questo capitolo verrà introdotto il modello fisico della SRP (paragrafo 2.1), si tratterà la semplificazione del modello sferico (paragrafo 2.2) ed infine verrà presentato un confronto tra le propagazioni di due orbite Geostazionarie effettuate su GMAT (paragrafo 2.3).

Nello specifico, il confronto delle due orbite ha come obiettivo quello di dimostrare l'importanza del contributo dovuto alla SRP nella propagazione di una traiettoria.

2.1 Modellazione matematica della SRP

Il flusso della radiazione emessa dal Sole genera una pressione che può essere valutata come

$$P_{SRP} = \frac{K\phi}{c\ {R_{AU}}^2} \tag{2.1}$$

dove K è la frazione del disco solare visibile dal satellite, ϕ è il valore del flusso solare alla distanza di 1 AU, c è la velocità della luce e R_{AU} è la distanza sole-satellite in AU [8].

Quando la radiazione descritta giunge sulle superfici del satellite, essa viene in parte assorbita ed in parte riflessa, dove la riflessione può avvenire sia in modo speculare (Figura 2.2 (a)) che diffuso (Figura 2.2 (b)).

La componente della forza prodotta dall'assorbimento (rappresentata in Figura 2.1) può essere valutata come

$$F_a = -P_{SRP}\, C_a\, A \cos\theta\, \hat{S} \qquad \textbf{(2.2)}$$

dove C_a corrisponde alla frazione di luce assorbita dalla superficie del satellite, $A\cos\theta$ è la porzione di superficie del satellite visibile dal punto di vista del Sole, θ è l'angolo tra la normale alla superficie $\hat{N}$ e la direzione dei raggi incidenti $\hat{S}$ [8].

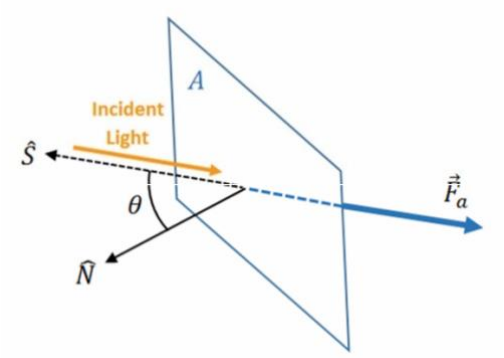

Figura 2.1 Radiazione solare incidente su di una lastra piana generica [8].

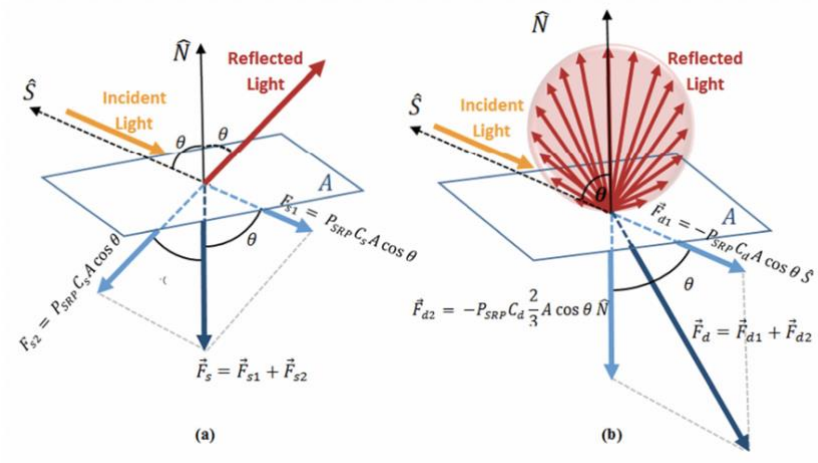

Figura 2.2 (a) Luce incidente riflessa in modo speculare e (b) Luce incidente riflessa in modo diffuso [8].

La componente della forza dovuta alla riflessione speculare (rappresentata in Figura 2.2 (a)) si può valutare come

$$\overrightarrow{F_s} = -2P_{SRP}\, C_s A\, (cos\,\theta)^2\, \hat{N} \qquad \textbf{(2.3)}$$

dove C_s è la frazione di luce riflessa in modo speculare, mentre nel caso di riflessione diffusa (rappresentata in Figura 2.2 (b)) essa si può calcolare come

$$\overrightarrow{F_d} = -P_{SRP}\, C_d A \cos\theta (\hat{S} + \frac{2}{3}\hat{N}) \qquad \textbf{(2.4)}$$

dove C_d è la frazione di luce riflessa in modo diffuso [8].

Considerando che deve valere

$$Cs + Cd + Ca = 1 \qquad \textbf{(2.5)}$$

(2.2) può essere riscritta come

$$\overrightarrow{F_a} = -P_{SRP}\,(1 - Cs - Cd) A \cos\theta\, \hat{S} \qquad \textbf{(2.6)}$$

Sommando infine i tre contributi descritti sopra è possibile trovare il valore complessivo della forza dovuta alla SRP agente su di una lastra piana generica [8] come

$$\overrightarrow{F_{SRP}} = \overrightarrow{F_s} + \overrightarrow{F_d} + \overrightarrow{F_a} =$$

$$= -P_{SRP}\, A\, \cos\theta \left\{(1 - Cs)\hat{S} + 2(Cs\cos\theta + \tfrac{1}{3}Cd)\hat{N}\right\} \qquad \textbf{(2.7)}$$

2.2 Il modello "Cannonball"

Il calcolo della forza dovuta alla SRP sul satellite può essere semplificato utilizzando quello che viene chiamato "modello sferico" o "Cannonball", in tale modello, la forma del satellite viene approssimata con quella di una sfera la cui superficie esterna è equivalente a quella complessiva della geometria reale del satellite.

Alla superficie della sfera viene quindi assegnato un coefficiente di riflettività complessivo definito come

$$C_R = 1 + C_s + {}^{2}/_{3}\, C_d \tag{2.8}$$

A questo punto, è possibile semplificare la (2.7) nel modo seguente

$$\overrightarrow{F_{SRP}} = -P_{SRP}\, A_{cb} C_R \hat{S} \tag{2.9}$$

dove A_{cb} è l'area della sezione frontale della sfera che approssima la geometria del satellite [8].

2.3 Propagazione con modello sferico Cannonball

Come è già stato affermato nell'introduzione del capitolo, GMAT permette di considerare il contributo dovuto alla SRP utilizzando un modello sferico predefinito, l'accelerazione dovuta alla SRP sul satellite è calcolata dal software come segue

$$\vec{a}_{SRP} = \nu P_S r_{AU}^2 \frac{C_R A}{m} \frac{\boldsymbol{r_{vs}}}{{r_{vs}}^3} \tag{2.10}$$

dove ν è il fattore di eclisse che indica la frazione di superficie del disco solare che è visibile dal satellite, P_S è il flusso solare alla distanza di 1 AU, r_{AU} è la distanza media della terra dal Sole espressa in km, C_R è il coefficiente di riflettività, la sua formulazione è espressa in (2.8), A è l'area frontale della superfice sferica, m è la massa del satellite, $\boldsymbol{r_{vs}}$ è il vettore che esprime la posizione dello s/c rispetto al Sole ed r_{vs} è il suo modulo [7].

L'effetto della sua attivazione nella propagazione di un'orbita Geostazionaria è stato quindi analizzato per comprendere la dinamica del contributo dato dall'accelerazione dovuta alla SRP.

Nella Tabella 2.1 sono riportate le impostazioni utilizzate per la propagazione dell'orbita.

In assenza di disturbi dovuti ad accelerazioni non gravitazionali, il periodo di un'orbita (T) dipende esclusivamente dal semiasse maggiore dell'orbita stessa (a).

In particolare, in un'orbita circolare attorno alla Terra è possibile calcolare il periodo come segue:

$$T = 2\pi \times \sqrt{a^3/\mu_{Terra}}$$

dove μ_{Terra} è la costante planetaria della Terra ed equivale al prodotto tra la massa della Terra e la costante di gravitazione universale G.

Ponendo $T = T_{sidereo}$ (dove $T_{sidereo}$ è pari a $23\ h\ 56\ min\ 4\ sec$, che equivale ad una rotazione completa della Terra attorno all'asse dei poli) e risolvendo l'equazione per a, si trova che il semiasse maggiore che garantisce un periodo orbitale pari a quello di rotazione terrestre è di circa $42168\ km$.

Dunque, l'orbita Geostazionaria è una particolare orbita circolare che ha velocità angolare media uguale a quella terrestre. Ponendo il satellite in un'orbita di questo tipo, esso risulterà fermo rispetto ad un osservatore solidale con la superficie Terrestre. Tale caratteristica può essere utilizzata ad esempio per effettuare telecomunicazioni satellitari con una antenna fissa a Terra ed assicurare inoltre una copertura continua.

Tabella 2.1 Variabili di propagazione inserite su GMAT per un'orbita Geostazionaria con modello Cannonball attivo.

CONDIZIONI INIZIALI		**PARAMETRI DI INTEGRAZIONE**	
Formato Epoca	UTCGregorian	Tipo di integratore	RungeKutta89
Epoca Iniziale	01 Jan 200012:00:28	Passo Minimo	60 sec
Sistema di Riferimento	EarthMJ2000	Passo Massimo	2700 sec
Formato Stato Iniziale	Keplerian	Accuratezza	1e-8
Semiasse Maggiore	42168 km	Tempo di Propagazione	10 giorni
Eccentricità	0	Modello di gravità esteso	No
Longitudine del Nodo Ascendente	0°	Punti di Massa	Terra, Luna
Argomento del Pericentro	0°	Forze non gravitazionali	SRP (modello Cannonball)
Inclinazione	0°		
Anomalia	0°		
Massa del satellite	850 km		
$\boldsymbol{C_R}$	1.789		
$\boldsymbol{A_{cb}}$	5 m^2		

In Figura 2.3 è rappresentato l'errore che si commette propagando la traiettoria senza considerare la SRP rispetto al caso in cui la traiettoria è propagata calcolando la SRP con il modello sferico predefinito in GMAT. Dall'andamento dei dati rappresentati è quindi possibile osservare come la propagazione che non considera l'accelerazione dovuta alla SRP presenti un errore che si propaga nel tempo in modo non trascurabile.

Si può quindi concludere che un modello che non considera l'accelerazione dovuta alla SRP non è adatto a rappresentare in modo accurato la dinamica del sistema.

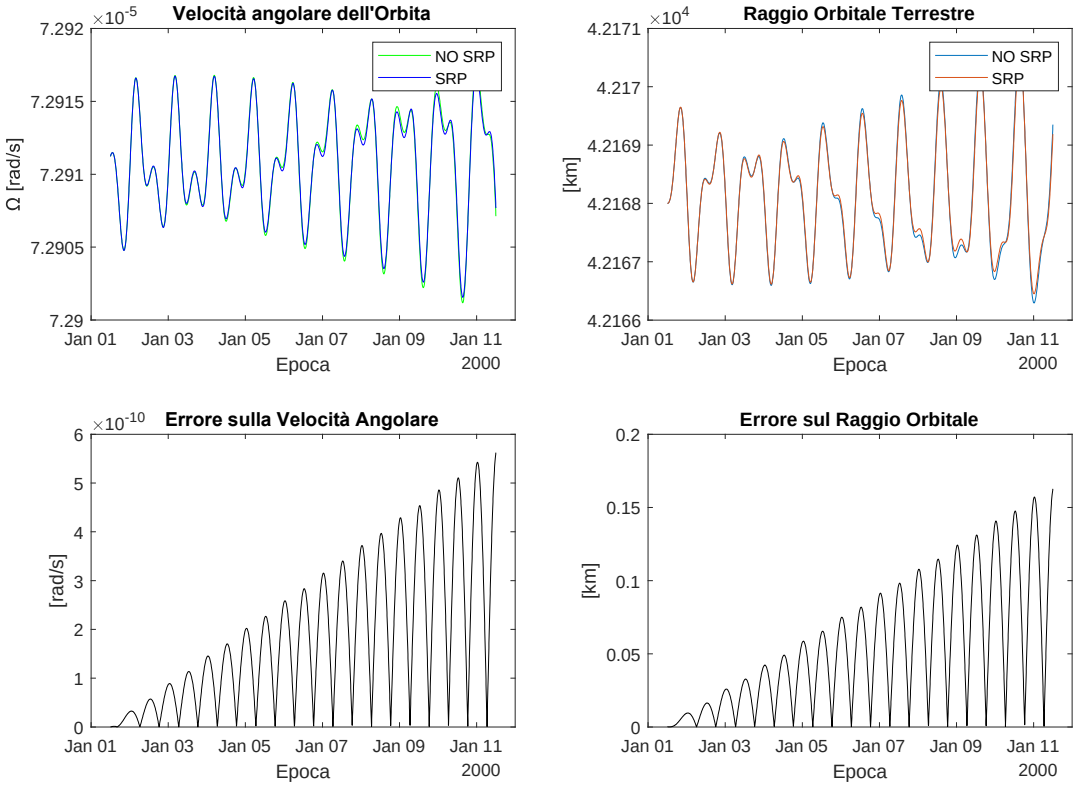

Figura 2.3 Raggio e velocità orbitale con e senza modello sferico SRP e relativo errore.

Capitolo 3 : Il modello SPAD

Come già accennato precedentemente, il software GMAT dispone di due modelli per valutare l'accelerazione dovuta alla SRP nella propagazione di un'orbita: il modello sferico integrato (approfondito al paragrafo 2.3) ed il modello SPAD.

Nel corso di questo capitolo si procederà innanzitutto con un approfondimento del concetto di modello SPAD, enunciando le equazioni su cui si basa (paragrafo 3.1) e approfondendo la struttura del file descrittore (paragrafo 3.2) che viene poi fornito a GMAT per la propagazione orbitale.

Inoltre, sarà analizzata la semplificazione sferica del modello SPAD, effettuando quindi un confronto con il modello sferico integrato di GMAT (paragrafo 3.3) per valutare le differenze ed i vantaggi e svantaggi associati.

Al fine di validare la funzione *SPADspherical* sarà anche mostrato un confronto fra la traiettoria di un CubeSat 6U propagata con GMAT prima con il modello sferico integrato e poi con il modello SPAD, e la traiettoria propagata con il modello sferico predefinito del software MONTE.

Per effettuare questo confronto è stato utilizzato il Toolkit SPICE [2] con MATLAB.

Infine, verrà discussa anche l'implementazione di una funzione MATLAB che genera un file SPAD per satelliti di forma cubica sempre utilizzando un modello analitico (paragrafo 3.4).

3.1 Teoria del modello SPAD

Ai fini della comprensione del modello SPAD, si immagini uno spazio tridimensionale dove esistono esclusivamente il satellite ed una sorgente luminosa (in questo caso la sorgente luminosa è il Sole).

La sorgente luminosa viene considerata come un corpo puntiforme posizionato all'infinito, tale per cui i raggi di luce proiettati che incidono sulla superficie del satellite possono essere considerati paralleli fra loro.

Ci si ponga quindi in un sistema di riferimento solidale con il satellite (Body-Fixed) con l'origine definita nel punto **C** e la sorgente luminosa posizionata nel punto **S** che ruota attorno all'origine al variare degli angoli di azimut ed elevazione, così come rappresentato in Figura 3.1.

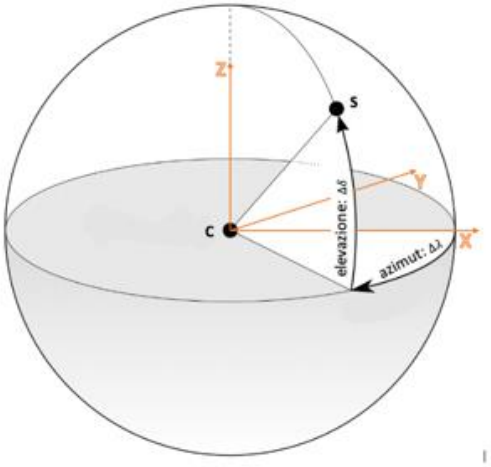

Figura 3.1 Rappresentazione geometrica dei riferimenti utilizzati per la definizione del modello SPAD: il satellite è posizionato nell'origine C del sistema di assi solidali al satellite (Body-Fixed), la sorgente di luce è posizionata nel punto S.

L'azimut può quindi variare tra -180° e +180° e l'elevazione tra -90° e +90°, non in modo continuo, ma con un passo di variazione definito in ingresso dall'utente.

Da (2.7) sappiamo che il modulo di $\overrightarrow{F_{SRP}}$ dipende dalla distanza dal Sole attraverso P_{SRP} , perciò il file SPAD dovrebbe tenerne conto. In realtà, la potenzialità del modello SPAD risiede nel fatto che, per semplificare l'implementazione del modello ed incrementarne la portabilità tra algoritmi di propagazione diversi, ciò che viene calcolato per una ipotetica superficie del satellite ad ogni angolazione è

$$\overrightarrow{F_{SRP}}\Big/_{P_{SRP}} = -A\ \cos\theta\left\{(1-Cs)\hat{S} + 2(Cs\cos\theta + \frac{1}{3}Cd)\hat{N}\right\}\ \ [m^2] \qquad \textbf{(3.1)}$$

che dipende esclusivamente dall'orientamento relativo dei corpi Sole-satellite e dalle caratteristiche superficiali delle facce illuminate

Infatti, il versore $\hat{S}$ è univocamente determinato per ogni combinazione di angoli di azimut ed elevazione, è sufficiente conoscere la direzione della normale alla superficie $\widehat{N}$ e le caratteristiche ottiche della superficie (*Cs* e *Cd*) ed è possibile calcolare $\overrightarrow{F_{SRP}}/P_{SRP}$.

Azimut ed elevazione variano in modo discreto, il passo dei due angoli è un parametro indipendente che deve essere scelto considerando che più è piccolo, più il risultato della propagazione dell'orbita sarà accurato.

Infatti, GMAT approssima $\overrightarrow{F_{SRP}}$ negli angoli intermedi (quindi non presenti nel file SPAD) attraverso un processo di interpolazione polinomiale fra i due estremi aumentando inevitabilmente l'incertezza.

In fase di propagazione della traiettoria, GMAT calcola l'accelerazione dovuta alla SRP moltiplicando i valori dei coefficienti (3.1) (riportati nella tabella principale all'interno del file SPAD) per P_{SRP} , dove quest'ultima viene calcolata ad ogni passo di integrazione sulla base della distanza del satellite dal Sole.

3.2 Modello SPAD per una geometria generica

Per considerare il satellite con la sua forma reale, la strategia con la quale si procede per implementare il modello è quella di considerare la geometria della superficie esterna del satellite come una composizione di lastre piane (triangoli o quadrilateri) elementari.

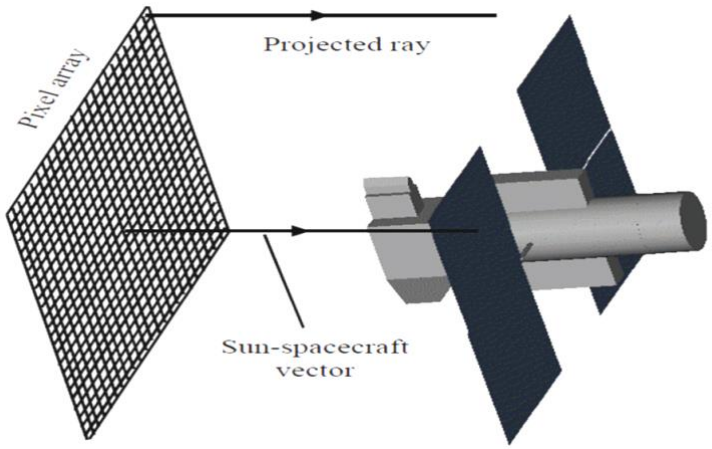

Figura 3.2 Immagine che chiarisce il procedimento di calcolo di $\overrightarrow{F_{SRP}}$ / P_{SRP} nel caso di un satellite di forma generica per un certo azimut ed elevazione.

Dal punto di vista operativo, la sorgente di luce che (deve proiettare raggi all'incirca paralleli fra loro) viene approssimata ad una sorgente bidimensionale posta ad una distanza unitaria dal satellite.

Tale sorgente viene quindi suddivisa in una matrice di pixel luminosi (indicizzati con *i* per le righe e *j* per le colonne), ognuno di essi emetterà un raggio che potrà intercettare o meno la superficie del satellite come in Figura 3.2.

Per l'*i*-esimo, *j*-esimo pixel della matrice vale dunque

$$\left(\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}\right)_{i,j} = -A_{pixel}\left\{(1-Cs)\hat{S} + 2\left(Cs\cos\theta + \frac{1}{3}Cd\right)\hat{N}\right\} \qquad \textbf{(3.2)}$$

dove A_{pixel} è l'area del pixel, è l'angolo di inclinazione fra il raggio proiettato dal pixel ($\hat{S}$, che risulta essere uguale per ogni pixel dal momento che si sta considerando una sorgente bidimensionale) e la normale uscente dalla porzione di superfice del satellite intercettata $\hat{N}$, *Cs* e *Cd* sono i coefficienti che esprimono la frazione di luce riflessa in modo speculare e diffuso da parte della lastra piana intercettata.

Dunque, proiettando ogni raggio e sommando tutti i contributi *i,j* dei singoli pixel si trova un'approssimazione del valore totale reale di $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$, dove più è alto il numero di pixel utilizzati nella matrice, minore sarà l'errore commesso nell'approssimazione.

3.3 Struttura di un file SPAD

Il file SPAD è un file di testo ASCII che contiene una tabella che riporta i valori del vettore $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ espresso nel S.d.R. Body-Fixed per ogni combinazione di angoli di azimut ed elevazione della sorgente luminosa rispetto al satellite.

Il file per essere compatibile con GMAT necessita di una corretta intestazione, un esempio è riportato nel File 3.1.

```
1.  Version            : 4
2.  System             : 1
3.  Analysis Type      : Area
4.  Pixel Size         : 0.2035 [m^2]
5.  Spacecraft Size    : ?
6.  Pressure           : 0
7.  Center of Mass     : [0 0 0]
8.  Current time       : 27/01/2021
9.
10. Motion             : 1
11.  Name               : Azimuth
12.  Method             : Step
13.  Minimum            : -180
14.  Maximum            : +180
15.  Step               : 5
16. Motion             : 2
17.  Name               : Elevation
18.  Method             : Step
19.  Minimum            : -90
20.  Maximum            : +90
21.  Step               : 5
22. :END
23.
24. Record count       : 2701
25.
26.   Azimuth    Elevation     Force(X)     Force(Y)      Force(Z)
27.   degrees    degrees         m^2          m^2           m^2

28.   -------    ---------     --------     ---------     --------
```

File 3.1 Esempio di intestazione di un file SPAD.

Nell'intestazione sono immagazzinate le seguenti informazioni:

1. *version*: la versione del file implementato.
2. *system*: il nome del satellite di cui si scrive il file.
3. *analysis type*: "Area" nel caso di analisi sulla SRP o "Drag" nel caso di analisi sulla resistenza aerodinamica.
4. *pixel size*: area del pixel della sorgente bidimensionale di cui si è parlato nel paragrafo 3.2.
5. *spacecraft size*: volume complessivo del satellite in m^3.
6. *pressure*: valore di riferimento impostato a 1 atm in ogni file
7. *center of mass*: l'origine del S.d.R. Body-Fixed.
8. *current time*: l'epoca in cui viene scritto il file SPAD.

Dalla riga 10 alla riga 22 del File **3.1** vengono definiti i due movimenti (*motion*) che la sorgente di luce esegue attorno all'origine nel S.d.R. Body-Fixed, ovvero azimut che varia da un minimo (*minimum)* di -180° ad un massimo (*maximum*) di +180° con

passo (*step*) pari a 5° ed elevazione che varia da un minimo di -90° ad un massimo di +90° con passo pari a 5°.

Alla riga 24 *record count* riporta il numero di combinazioni di azimut ed elevazione che sono state valutate nel file, in questo caso

$$\{[180-(-180)]\div 5\}_{azimut}\cdot\{[90-(-90)]\div 5\}_{elevzione}=2701$$

Dalla riga 24 alla riga 28 viene inizializzata la struttura che dovrà avere il corpo del file, ovvero in ordine: angolo di azimut (*azimut*), angolo di elevazione (*elevation*) e successivamente le tre componenti *Force(X)*, *Force(Y)* e *Force(Z)* calcolate in m^2 di $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ nel S.d.R Body-Fixed.

Una traccia del corpo di un file SPAD è riportata nel File **3.2**.

```
1.  …
2.   -180.000000      75.000000     0.022867425821946     -0.000000000468304     -0.084632078214970
3.   -180.000000      80.000000     0.016349640141298     -0.000000000468304     -0.086077036512620
4.   -180.000000      85.000000     0.006065902538907     -0.000000000535204     -0.064083819729980
5.   -180.000000      90.000000     0.000000000000000     -0.000000000468304     -0.032140919404659
6.   -175.000000     -90.000000     0.000000000000000     0.000000000512904     0.035201959347959
7.   -175.000000     -85.000000     0.006403006710916     0.000936461278855     0.069680195362214
8.   -175.000000     -80.000000     0.018378462298357     0.002271179206631     0.093126454852672
9.   -175.000000     -75.000000     0.034024570467676     0.004016306521411     0.119196884345507
10.  -175.000000     -70.000000     0.055095676521940     0.006146790797029     0.141173635459400
11.  -175.000000     -65.000000     0.077566335170758     0.008488988823225     0.156823841441023
12.  -175.000000     -60.000000     0.103720088547535     0.010687876987799     0.172420879216792
13.  -175.000000     -55.000000     0.133803902500696     0.013007428966112     0.181546370058518
14.  -175.000000     -50.000000     0.162599937031512     0.015526755542059     0.187633676030133
15.  -175.000000     -45.000000     0.190981439439397     0.017321104336707     0.183995996629481
16.  …
```

File 3.2 Esempio di traccia del corpo di un file SPAD.

Come si può notare, il corpo del file segue la struttura inizializzata nell'intestazione.

3.4 Semplificazione del modello sferico

Il modello SPAD può essere semplificato come nel caso del modello Cannonball, considerando quindi il satellite strutturato con una geometria sferica e calcolando il coefficiente $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ a partire da (2.9) come segue

$$\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}=-A_{CB}C_R\,\hat{S} \qquad \textbf{(3.3)}$$

Tale semplificazione facilita i calcoli per ogni azimut ed elevazione rispetto a (3.2) e inoltre fornisce un risultato esatto in quanto derivato da una soluzione analitica.

Con l'intento di comprendere la metodologia con cui deve essere strutturata una funzione MATLAB per poter generare un file SPAD è stata sviluppata la funzione di *SPADspherical* (Files A).

Tale funzione è stata costruita sulla base delle nozioni contenute nel paragrafo 2.1, considerando dunque la sorgente di luce puntiforme che ruota attorno al sistema di riferimento Body-Fixed variando azimut ed elevazione ad ogni iterazione del ciclo for relativo, utilizzando il passo fornito in ingresso dall'utente.

SPADspherical acquisisce come valori di ingresso i seguenti parametri:

- C_s: coefficiente che descrive la frazione di luce riflessa in modo speculare.
- C_d : coefficiente che descrive la frazione di luce riflessa in modo diffuso.
- A_{cb}: area della sezione della sfera di raggio R ($A_{cb} = \pi R^2$).
- $\Delta\lambda$: passo di variazione dell'angolo di azimut.
- $\Delta\delta$: passo di variazione dell'angolo di elevazione.
- $header$: struttura che contiene le informazioni necessarie a scrivere l'intestazione del file.
- $filename$: nome del file SPAD dove vengono salvati i risultati.

E restituisce in output il file SPAD completo di intestazione.

Ad ogni passo del ciclo iterativo vengono calcolate le componenti del versore $\hat{S}$ che descrive la direzione della retta congiungente il punto S con il punto C (Figura 3.1).

Il coefficiente che esprime il rapporto fra la forza e l'intensità della SRP viene calcolato come mostrato in (3.3) ed il risultato viene scomposto lungo i tre assi (X, Y e Z) del sistema di riferimento Body-Fixed.

Nell'ultima parte della funzione, viene scritta l'intestazione sulla base della struttura fornita nella variabile *header* e viene poi scritta nel file.

Infine, all'interno di un ciclo for viene scritta ogni riga del file che sarà poi automaticamente salvato con il nome fornito in ingresso dall'utente.

3.5 Validazione della funzione MATLAB per il modello SPAD

Per verificare la correttezza del modello implementato attraverso la funzione descritta nel paragrafo 3.4, il file SPAD è stato quindi confrontato con il modello sferico predefinito di GMAT analizzato al paragrafo 2.2.

I parametri inseriti in ingresso alla funzione *SPADspherical* sono contenuti in Tabella 3.2. ed i parametri di propagazione inseriti su GMAT sono riportati in Tabella 3.2.

Tabella 3.1 Parametri forniti in ingresso alla funzione MATLAB per generare un file SPAD che descriva un satellite la cui struttura è equivalente a quella di una sfera.

PARAMETRO	VALORE	DESCRIZIONE
A_{cb}	5 m^2	Pixel Size
$\Delta\lambda$	5°	Passo Azimut
$\Delta\delta$	5°	Passo Elevazione
C_s	0.789	Coefficiente Riflessione
C_d	0	Coefficiente Diffusione

Tabella 3.2 Parametri di propagazione di GMAT per un'orbita geostazionaria ai fini di un confronto fra modello sferico predefinito in GMAT ed il modello SPAD.

CONFIGURAZIONE PROPAGAZIONE		PARAMETRI DI INTEGRAZIONE	
Formato Epoca	UTCGregorian	Tipo di integratore	RungeKutta89
Epoca Iniziale	01 Jan 2000 12:00:28	Passo Minimo	60 sec
Sistema di Riferimento	EarthMJ2000	Passo Massimo	2700 sec
Formato Stato Iniziale	Keplerian	Accuratezza	1e-8
Semiasse Maggiore	42168 km	Tempo di Propagazione	10 giorni
Eccentricità	0	Modello di gravità esteso	No
Longitudine del Nodo Ascendente	0°	Punti di Massa	Terra, Luna
Argomento del Pericentro	0°	Forze non gravitazionali	SRP (modello CannonBall). SRP (SPAD file).
Inclinazione	0°		
Anomalia	0°		

Massa del satellite	850 kg		
$\boldsymbol{C_R}$	1.789		
$\boldsymbol{A_{cb}}$	5 m^2		

Nella Figura 3.3 sono stati riportati gli errori commessi su RMAG (modulo del raggio) e VMAG (modulo della velocità) propagando la traiettoria con il modello sferico SPAD (i cui parametri sono definiti nel file generato nel paragrafo precedente e riportati in Tabella 4.1 piuttosto che con il modello sferico predefinito di GMAT.

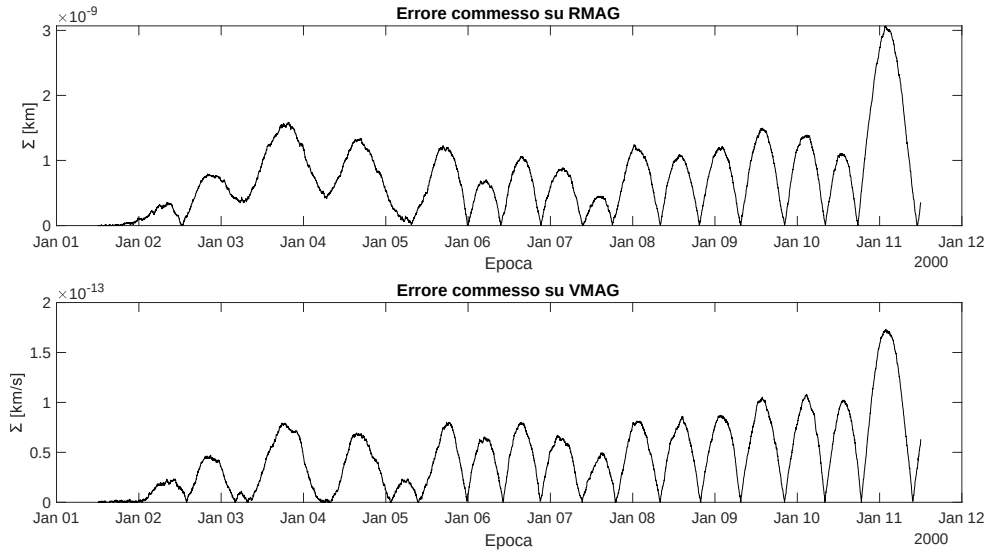

Figura 3.3 Errore commesso su RMAG e VMAG propagando un'orbita geostazionaria con il modello sferico SPAD piuttosto che con il modello sferico predefinito in GMAT.

Nei risultati riportati si può quindi osservare la presenza di un errore fra i due modelli, che rimane contenuta entro i $3 \times 10^{-9}\, km$ per il raggio e circa $2 \times 10^{-13}\, km/_s$ per la velocità, durante la propagazione della traiettoria per una durata totale di dieci giorni.

Per poter minimizzare l'errore descritto, sono state eseguite delle modifiche ai parametri dell'integrazione e si è infine trovato che in questo caso specifico è possibile annullare l'errore, come dimostrato in Figura 3.4, utilizzando l'integratore PrinceDormand78 con accuratezza impostata a 10^{-14}, passo minimo 10^{-8} sec e passo massimo di 60 *sec*.

Sulla base delle considerazioni effettuate, si può quindi assumere che l'errore commesso è principalmente di origine numerica. Si può quindi concludere che il file per il modello sferico SPAD, generato con la funzione MATLAB come descritto nel precedente paragrafo, permette di ottenere dei risultati equiparabili a quelli del modello sferico predefinito di GMAT ed è quindi utilizzabile per calcolare l'accelerazione dovuta a SRP.

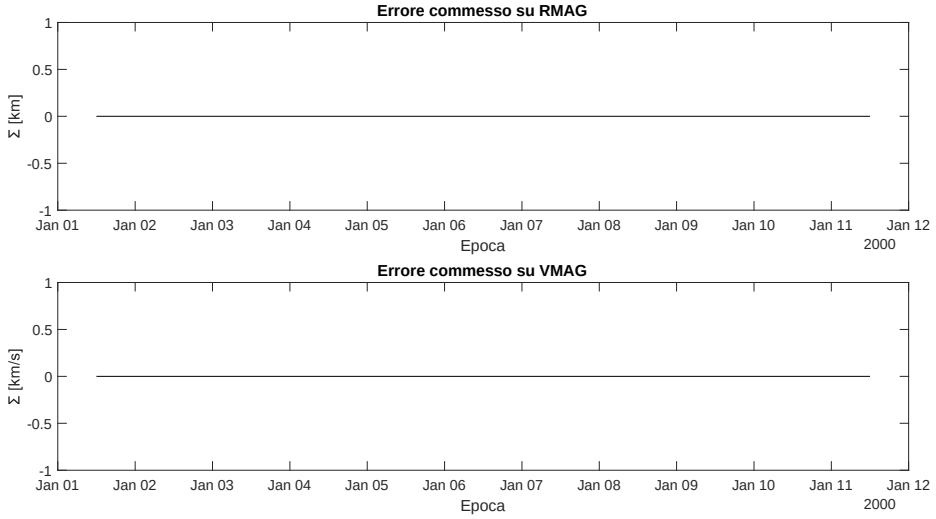

Figura 3.4 Errore commesso su RMAG e VMAG propagando un'orbita geostazionaria con il modello sferico SPAD rispetto al modello sferico integrato in GMAT con un metodo di integrazione PrinceDormand78 (accuratezza $\mathbf{10^{-14}}$, passo minimo di $\mathbf{10^{-8}}$ sec e passo massimo di 60 sec).

3.6 Confronto propagazione GMAT/ MONTE con semplificazione a geometria sferica

Con il fine di applicare *SPADspherical* ad un caso reale è stato effettuato un confronto tra la traiettoria di un CubeSat propagata su GMAT e la stessa traiettoria propagata con il software MONTE (fornita in ingresso in formato SPICE kernel a questa analisi).

Per effettuare tale confronto si è deciso di propagare l'orbita dello spacecraft per una durata di 10 giorni a partire dallo stato iniziale.

La traiettoria analizzata segue un percorso cislunare che porta il CubeSat ad eseguire un fly-by della Luna ed un successivo inserimento in un'orbita geocentrica ad elevata eccentricità.

La traiettoria di riferimento fornita in ingresso all'analisi è stata propagata con il software MONTE considerando:

- esclusivamente il GM della Terra e il GM della Luna nel modello gravitazionale
- il modello sferico predefinito del software per la valutazione della SRP

Utilizzando le librerie SPICE per MATLAB sono stati realizzati i grafici in Figura 3.5 e Figura 3.6, dove è possibile osservare l'orbita descritta proiettata sul piano XY ed XZ del S.d.R. ECLIPJ2000 (S.d.R. il cui piano XY che giace sul piano dell'eclittica e l'asse Z normale ad esso).

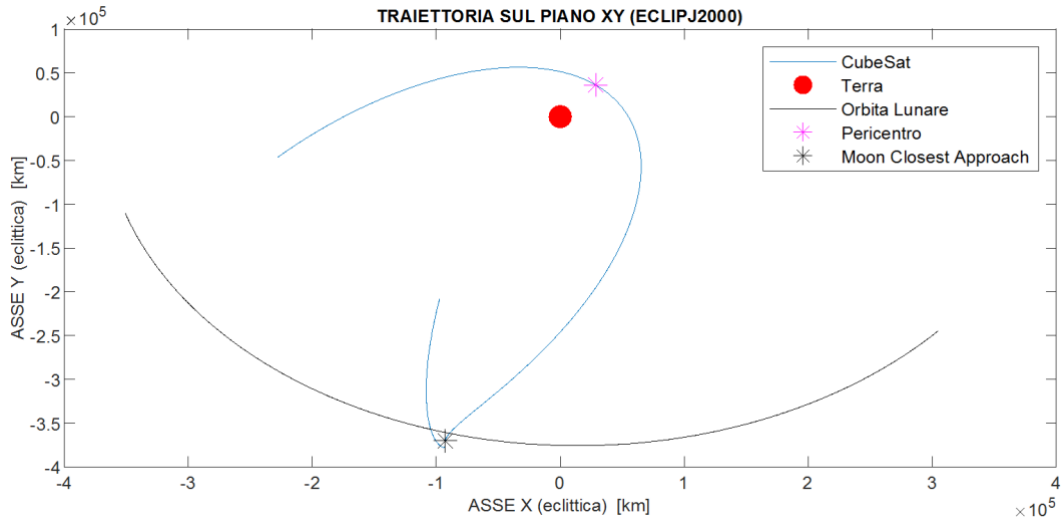

Figura 3.5 Traiettoria del C/S sul piano XY del Frame ECLIPJ2000.

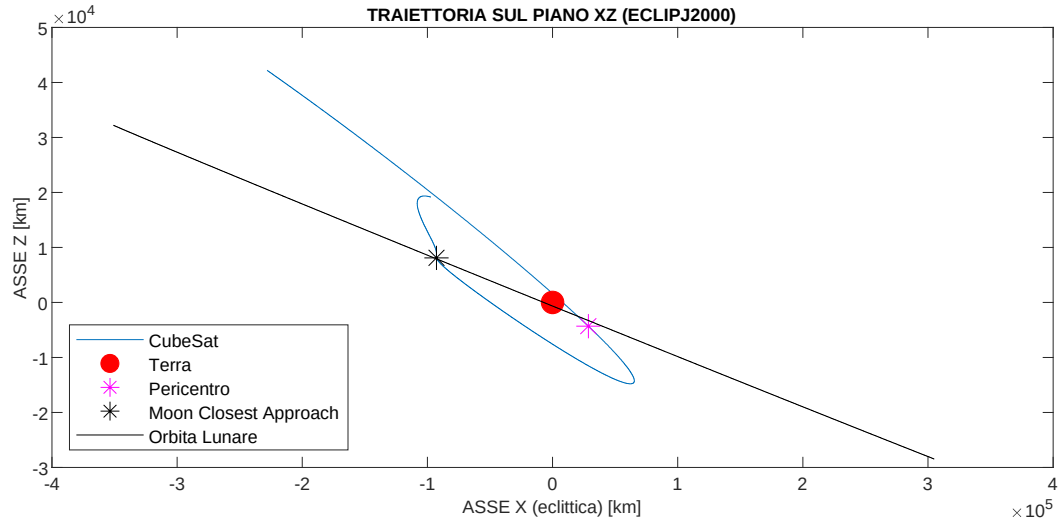

Figura 3.6 Traiettoria del C/S sul piano XZ del Frame ECLIPJ2000.

Sfruttando la funzione approfondita nel paragrafo 3.4, è stato generato un file SPAD per modello sferico per poter descrivere la forma del CubeSat e propagare quindi la sua traiettoria con GMAT.

In Tabella 3.3 vengono quindi elencati i parametri di ingresso utilizzati nella funzione MATLAB riportata in Files A, in Tabella 3.4 sono riportate le variabili di propagazione inserite su GMAT.

Tabella 3.3 Variabili di ingresso per la generazione di uno SPAD file per il C/S.

PARAMETRO	**VALORE**	**DESCRIZIONE**
A_{cb}	0.27 m^2	Pixel Size
$\Delta\lambda$	5°	Passo Azimut
$\Delta\delta$	5°	Passo Elevazione
C_s	0	Coefficiente Riflessione
C_d	0	Coefficiente Diffusione

Tabella 3.4 Variabili di propagazione del C/S su GMAT con il modello SPAD.

CONFIGURAZIONE PROPAGAZIONE		PARAMETRI DI INTEGRAZIONE	
Formato Epoca	TDBGregorian	Tipo di integratore	PrinceDormand78
Epoca Iniziale	29 Jun 2020 02:08:00.000	Passo Minimo	1e-8 s
Sistema di Riferimento	EarthICRF	Passo Massimo	60 s
Formato Stato Iniziale	Cartesiano	Accuratezza	1e-14
X	-97287.3004044117 m	Modello di gravità esteso	No
Y	-198295.227919193 m	Punti di Massa	Terra, Luna
Z	-65102.9940065872 m	Forze non gravitazionali	SRP (SPAD file)
Vx	-0.21948287944461 m/s	Tempo di Propagazione	10 giorni
Vy	-1.09956368105769 m/s		
Vz	-0.46046879825726 m/s		
Massa del satellite	13.3669 kg		
C_R	1		
A_{cb}	0.27 m^2		

Prima di passare all'analisi comparativa dei software GMAT e MONTE sono state confrontate le traiettorie del satellite propagate con il modello sferico predefinito e con lo SPAD file generato.

Gli errori sul modulo del raggio e sul modulo della velocità orbitale originati dal confronto indicato sono quindi riportati in Figura 3.7.

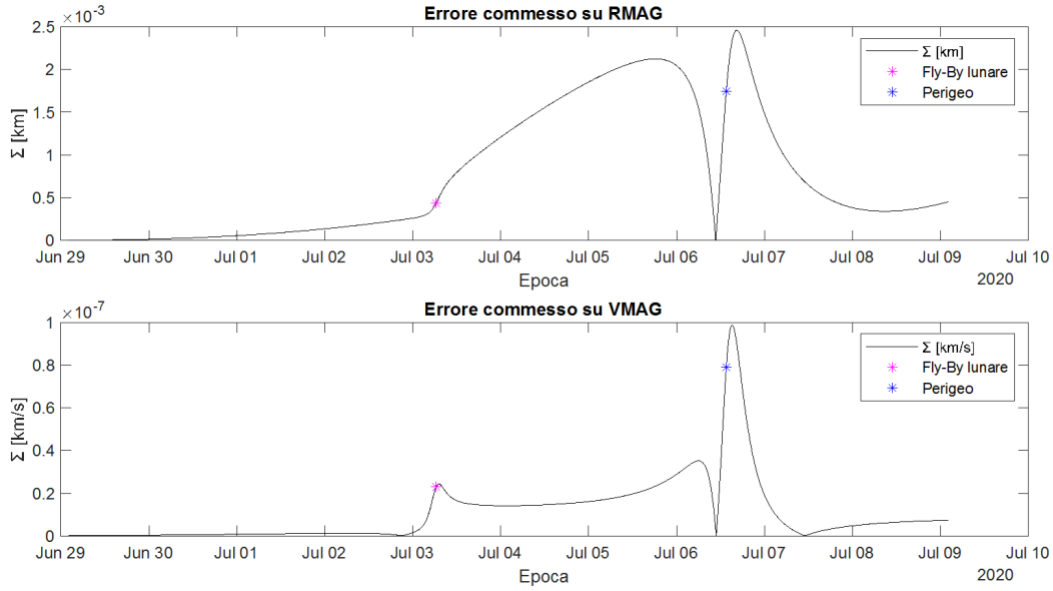

Figura 3.7 Errore sul modulo del raggio e modulo della velocità del C/S propagando la traiettoria con il modello sferico SPAD piuttosto che quello predefinito in GMAT per il calcolo dell'accelerazione dovuta alla SRP.

L'errore allo stato iniziale è molto piccolo ma non nullo. Si amplifica notevolmente in corrispondenza del fly-by lunare e successivamente si propaga aumentando ancora quando il satellite si avvicina al perigeo della sua orbita geocentrica.

Esaminando l'errore sul modulo del raggio RMAG si può osservare che esso rimane contenuto entro circa 2.5 m e l'errore su VMAG non supera $1 \times 10^{-7}\ km/s$.

Dati questi risultati è possibile affermare che la traiettoria propagata con il modello SPAD è sufficientemente accurata per poter essere confrontata con quella propagata dal software MONTE.

Per effettuare il confronto fra i due software sono state utilizzate variabili di integrazione analoghe a quelle in Tabella 3.4 ma sono stati modificati alcuni parametri

del modello dinamico gravitazionale in GMAT per poterlo rendere coerente con quello con cui è stata propagata la traiettoria in MONTE.

I parametri del modello che sono stati cambiati per l'analisi di questo paragrafo sono dunque i seguenti:

- Le effemeridi planetarie sono state cambiate da DE405 (predefinite di GMAT) a quelle più aggiornate DE430[1] poiché la traiettoria del CubeSat propagata in MONTE utilizzava quest'ultime nel modello dinamico.
- Il GM[2] della Terra è stato modificato utilizzando un valore più accurato[1]: μ = 398600.435436 km^3/s^2
- Il GM della Luna è stato modificato utilizzando un valore più accurato[1]: μ = 4902.8000066 km^3/s^2

Per quanto riguarda il confronto tra le traiettorie propagate utilizzando il modello sferico di GMAT e quello di MONTE, i risultati sono riportati in Figura 3.8, come si può vedere l'errore commesso è inizialmente non nullo e nel corso dei 10 giorni di simulazione si propaga raggiungendo un valore massimo di 0.6 km sul raggio e 1.4 $\times 10^{-5}$ km/s sulla velocità.

Successivamente, si è proceduto con il confronto tra le traiettorie del CubeSat propagate in GMAT con il modello SPAD e in MONTE con il modello sferico predefinito, i risultati sono riportati in Figura 3.9.

Osservando l'andamento dell'errore nei due casi (Figura 3.8 e Figura 3.9), si può notare che i valori trovati risultano molto simili tra loro.

Questi risultati si possono considerare come un'ulteriore conferma che il file SPAD generato è corretto e che l'errore commesso dal modello è quindi comparabile con quello ottenuto nella propagazione con il modello sferico predefinito di GMAT.

[1] È possibile trovare i file con le effemeridi planetarie DE405 e DE430 e i valori accurati dei GM di Terra e Luna nel sito riportato in [1]
[2] GM: costante di gravitazione planetaria, è il prodotto di G: costante di gravitazione universale e M: massa del corpo celeste.

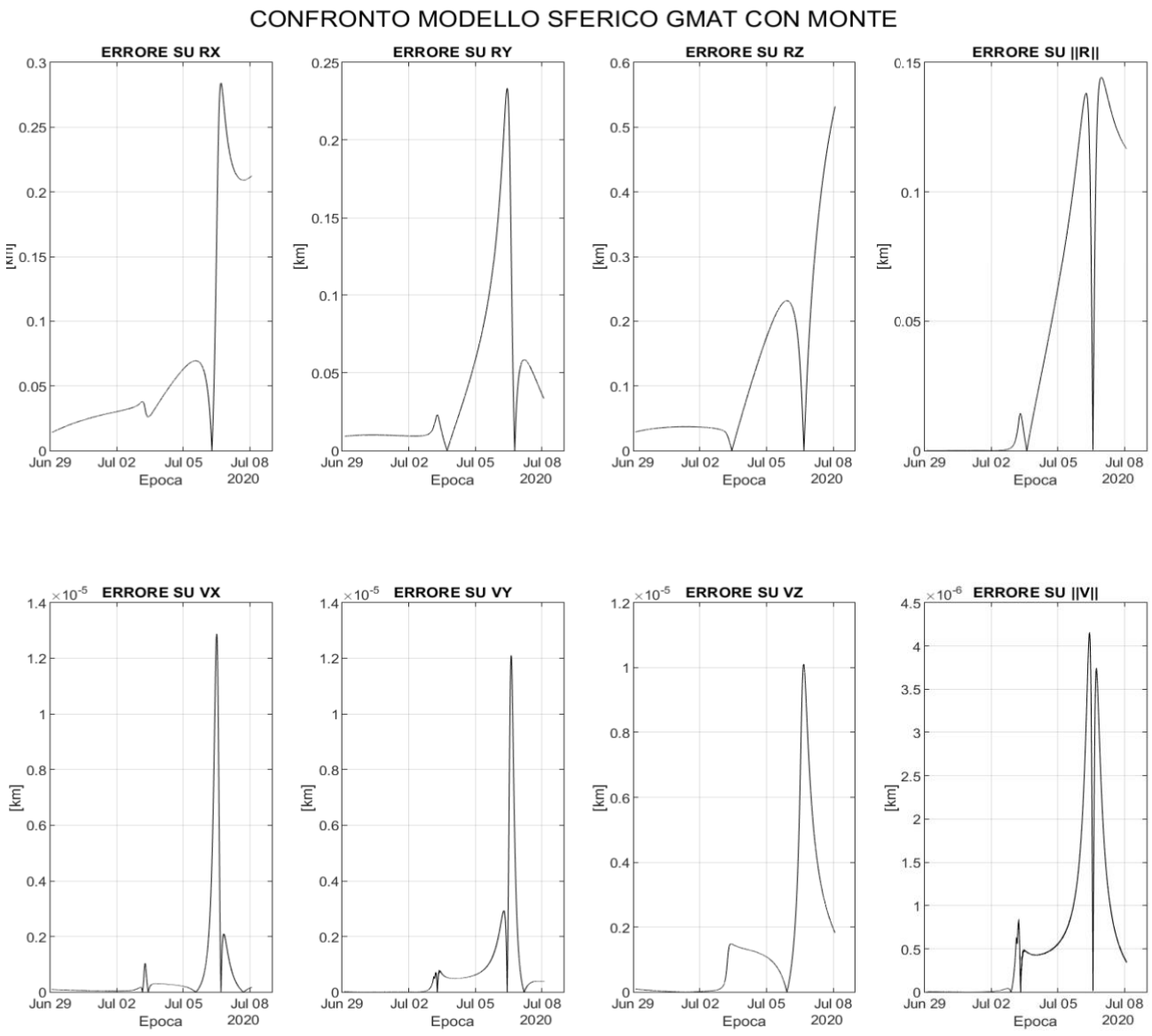

Figura 3.8 Andamento dell'errore su raggio e velocità fra le traiettorie propagate con GMAT e MONTE sfruttando i modelli sferici predefiniti.

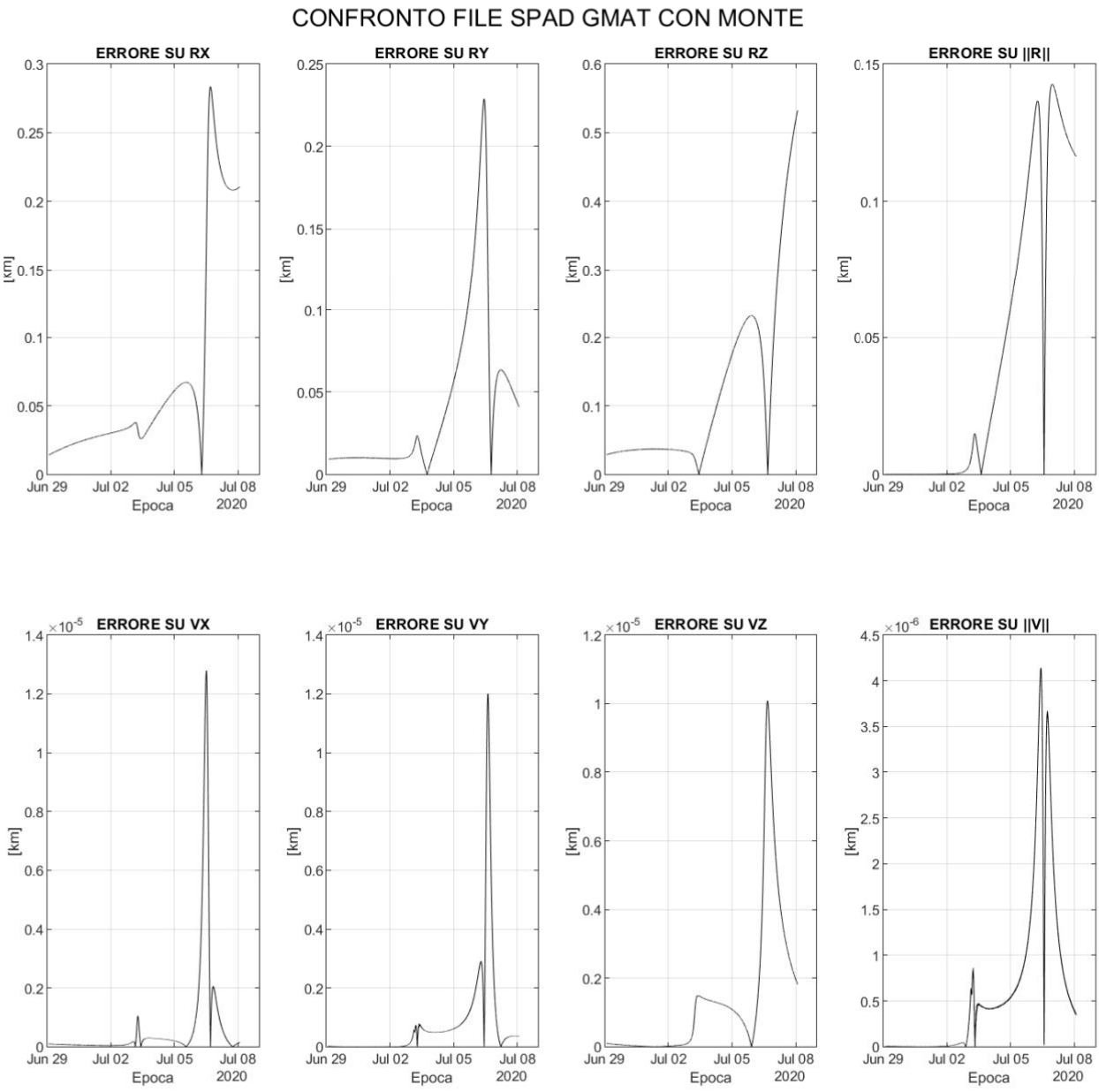

Figura 3.9 Andamento dell'errore su raggio e velocità fra le traiettorie propagate su GMAT sfruttando il modello SPAD e su MONTE sfruttando il modello sferico integrato.

3.7 Modello SPAD per un satellite di forma cubica

Nel caso in cui un satellite abbia geometria cubica (come, ad esempio, quella di un CubeSat da una unità, 1U), è possibile implementare una funzione che generi file SPAD con un calcolo analitico (Files B).

Infatti, è possibile considerare ogni faccia del cubo come una lastra piana e calcolare di conseguenza $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ come spiegato nel Capitolo 2, ovvero

$$\overrightarrow{F_{SRP}}\Big/_{P_{SRP}} = -A_{face}\ \cos\theta\left\{(1-Cs)\hat{S} + 2(Cs\cos\theta + \frac{1}{3}Cd)\widehat{N}\right\} \qquad \textbf{(3.4)}$$

dove A_{face} è l'area della faccia del cubo, θ è l'angolo tra il raggio di luce e la normale alla faccia considerata, *Cs* e *Cd* sono i coefficienti di riflessione speculare e diffusa della faccia considerata ed $\widehat{N}$ è la normale uscente della faccia considerata.

Dunque, la funzione *SPADcubical* realizzata in MATLAB acquisisce in ingresso i seguenti parametri:

- *Cs*: un vettore di sei elementi che riporta il valore del coefficiente di luce riflessa speculare per ogni faccia del cubo
- *Cd*: un vettore di sei elementi che riporta il valore del coefficiente di luce riflessa diffusa per ogni faccia del cubo
- A_{face}: area di una faccia del cubo in m^2
- $step\_el$: passo di variazione dell'elevazione
- $step\_az$: passo di variazione dell'azimut
- $header$: struttura che contiene i dati per scrivere l'intestazione
- $filename$: nome del file SPAD dove verranno salvati i risultati della funzione.

e restituisce in output il file SPAD completo di intestazione.

Nella prima parte della funzione vengono inizializzati i vettori $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$, $\hat{S}$ (direzione della sorgente di luce espressa nel S.d.R. Body-Fixed), $\widehat{N}_i$ (le normali uscenti di ogni faccia), θ_i (gli angoli tra $\widehat{N}_i$ ed $\hat{S}$), lambda e delta (rispettivamente angoli di azimut da -180° a 180° e angoli di elevazione da -90° a 90° con passo di variazione definito in ingresso).

All'interno di due cicli for annidati variano azimut ed elevazione e per ogni iterazione vengono calcolati $\hat{S}$ e θ_i.

Per ogni faccia del cubo di possono verificare due casi:

- θ minore di 90°, dove la faccia risulta illuminata dalla sorgente.
- θ maggiore di 90°, dove la faccia risulta in ombra rispetto alla sorgente.

Se la faccia è illuminata, viene calcolato $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ come riportato in (3.4).

Se la faccia è in ombra, $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$ riporta un valore pari a zero.

Eseguiti i calcoli per ogni azimut ed elevazione, si procede a scrivere la stringa di intestazione sulla base della struttura nella variabile *header* (fornito in ingresso) e tramite un ciclo for si scrive ogni riga del file con la struttura mostrata nel File 3.2.

I risultati della funzione *SPADcubical* saranno sfruttati al fine di effettuare una analisi comparativa con i risultati della funzione *SPADcomplex* trattata nel prossimo capitolo al paragrafo 4.1.2.

Capitolo 4 : Modello SPAD per geometria complessa con algoritmo di ray-tracing

In questo capitolo verrà approfondito nel dettaglio l'algoritmo realizzato per generare un file SPAD che tiene conto la geometria reale del satellite e non una semplificazione ad una sfera o ad un cubo.

In particolare, nel paragrafo 4.1 verrà descritta ogni parte dell'algoritmo di ray-tracing utlizzato, insieme alla relativa funzione *SPADcomplex*, implementata sfruttando le nozioni dell'algoritmo (riportata nell'appendice al Files C).

Nel paragrafo 4.2 verrà discusso il problema della risoluzione della matrice dei pixel con la quale viene approssimata la sorgente di luce nell'algoritmo e di come influisce sui tempi di calcolo.

Nel paragrafo 4.3 è riportato un confronto fra il file SPAD generato con tale funzione ed il file SPAD generato dalla funzione analitica riportata al paragrafo 3.7 per un satellite di forma cubica.

Nel paragrafo 4.4 la funzione verrà applicata al caso del CubeSat 6U (la cui traiettoria è riportata in Figura 3.5 e Figura 3.6) e si procederà con una analisi comparativa fra i risultati della propagazione su GMAT e su MONTE con la seguente strategia:

- Confronto del file generato dalla funzione SPAD ray-tracing (Files C) ed il file SPAD estrapolato da MONTE
- Confronto tra le traiettorie del C/S propagate dai due software: la traiettoria propagata in GMAT sfrutta il file SPAD generato con l'algoritmo di ray-tracing, mentre quella propagata in MONTE valuta la SRP attraverso il modello a pannelli elementari integrato nel software.

4.1 Algoritmo di ray-tracing semplificato

Per descrivere il funzionamento dell'algoritmo di ray-tracing si procede analizzandone ogni fase:

- Innanzitutto, si deve disporre di un modello 3D della superficie esterna del satellite, la cui geometria deve essere espressa come composizione di triangoli.
- Viene generata una matrice di pixel che rappresenta l'approssimazione della sorgente di luce bidimensionale dalla quale verranno proiettati i raggi.

La sorgente deve essere abbastanza grande da contenere la superficie del satellite da ogni angolazione, ma non troppo grande da richiedere un numero eccessivo di pixel per calcolare in modo accurato $\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}$, aumentando così troppo i tempi di calcolo (il problema della dimensione della sorgente è trattato più in dettaglio in 4.1.2).

- La sorgente viene ruotata in azimut ed elevazione a seconda dell'angolazione che si sta considerando.
- I raggi, la cui origine è il centro di ogni pixel, vengono proiettati uno ad uno sulla superficie del satellite.
- Per ogni raggio si calcola l'intersezione con i triangoli di cui è composta superficie esterna utilizzando l'algoritmo di Möller-Trumbore (MT).

4.1.1 Algoritmo d'intersezione di Möller-Trumbore (MT)

Il raggio di luce viene parametrizzato come

$$R(t) = O + D \times t \tag{4.1}$$

dove O è l'origine del raggio nel S.d.R. scelto (in questo caso Body-Fixed), D è il versore che esprime la direzione del raggio e t è il parametro attraverso il quale il raggio viene propagato nello spazio [6].

Il triangolo, rispetto al quale si desidera calcolare l'intersezione è univocamente definito dai suoi tre vertici: $V_0,\ V_1\ e\ V_2$ come in Figura 4.1.

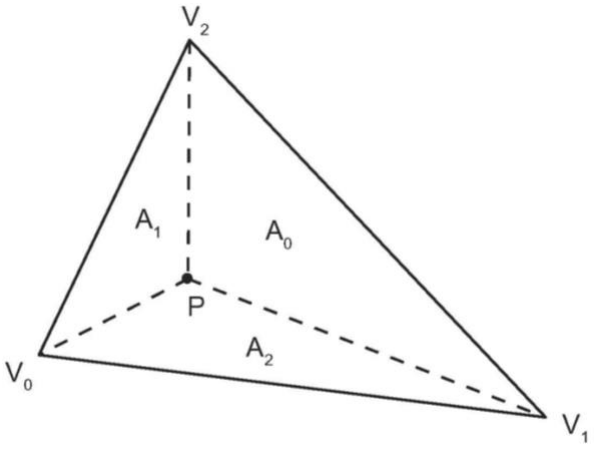

Figura 4.1 Punto espresso nelle coordinate baricentriche di un triangolo.

Tutti i punti appartenenti al piano (P) possono essere espressi utilizzando una base ortonormale di coordinate baricentriche w, u, v come segue

$$P = wV_0 + uV_1 + vV_2 \quad \textbf{(4.2)}$$

dove $w = {A_0}/{A_{TOT}}$, $u = {A_1}/{A_{TOT}}$ e $v = {A_2}/{A_{TOT}}$ con $A_{TOT} = A_1 + A_2 + A_0$ [6].

Le coordinate baricentriche hanno la particolarità che se il punto *P* è posizionato all'interno del triangolo sono tutte positive, mentre se *P* non appartiene al triangolo almeno una fra *u*, *v* o *w* risulta negativa.

Considerando che per la proprietà di qualsiasi base ortonormale vale

$$w + u + v = 1 \quad \textbf{(4.3)}$$

si può scrivere

$$P = (1 - u - v)V_0 + uV_1 + vV_2 \quad \textbf{(4.4)}$$

A questo punto è possibile imporre la relazione di intersezione tra il raggio e il piano a cui appartiene il triangolo [6].

$$P = R(t) \quad \textbf{(4.5)}$$

che, elaborando le due equazioni, porta a

$$[-D \quad V_1 - V_0 \quad V_2 - V_0] \times \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0 \qquad \textbf{(4.6)}$$

Se la soluzione di tale sistema lineare esiste ed è unica si trovano i valori *t, u,* e *v* che soddisfano la relazione (4.5) [6].

Partendo da *u* e *v*, si può trovare *w* come in (4.3).

Se una delle 3 coordinate baricentriche risulta essere negativa allora l'intersezione avviene al di fuori del triangolo, altrimenti avviene al suo interno.

Dunque, MT applicato ad ogni raggio proiettato dalla sorgente fornisce l'informazione su quali raggi intercettano quali triangoli della superficie del satellite nella particolare angolazione in cui si sta lavorando

Per ogni raggio:

- Se non intercetta nessun triangolo, si pone $\left(\overrightarrow{F_{SRP}}/P_{SRP}\right)_{i,j} = 0$ per il pixel *ij*.
- Se intercetta uno o più triangoli, viene considerato solo il triangolo il cui baricentro è il più vicino all'origine del raggio ed è possibile calcolare $\overrightarrow{F_{SRP}}/P_{SRP}$ come riportato in (3.2), ovvero

$$\left(\overrightarrow{F_{SRP}}/P_{SRP}\right)_{i,j} = -A_{pixel}\left\{(1-Cs)\hat{S} + 2(Cs\cos\theta + \frac{1}{3}Cd)\hat{N}\right\}$$

4.1.2 Implementazione della funzione MATLAB

In questo paragrafo è spiegata la struttura della funzione per il modello SPAD che tiene conto della reale geometria del satellite, implementata sulla base delle nozioni apprese al paragrafo 4.1.

La funzione *SPADcomplex* prende in ingresso:

- *objname*: il nome del file OBJ (con il relativo file MTL) che contiene il modello 3D del satellite di cui si vuole scrivere il file SPAD, la superficie

del satellite, per essere compatibile con le operazioni svolte dall'algoritmo di ray-tracing, deve essere implementata come composizione di triangoli (un esempio è riportato in Figura 4.2).

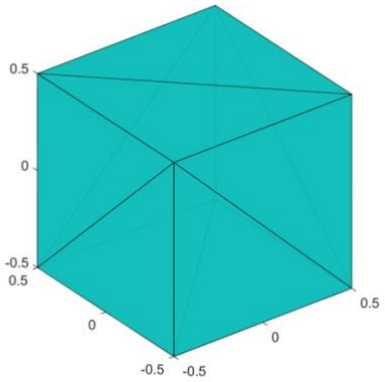

Figura 4.2 Esempio di modello 3D da immettere in ingresso alla funzione

- *step_el* e *step_az*: il passo di variazione di elevazione ed azimut di cui si è parlato al paragrafo 3.1.
- *header*: l'intestazione del file SPAD di cui si è parlato nel Capitolo 3
- *filename*: il nome del file SPAD dove verranno salvati i risultati

e restituisce in output il file SPAD completo di intestazione.

Il modello 3D del satellite viene importato in MATLAB con il formato OBJ.

Il formato OBJ (con il relativo MTL) è una estensione del formato Wavefront.

La motivazione che ha portato alla scelta di Wavefront è che generalmente, nella progettazione di una missione spaziale, la modellazione delle componenti hardware del satellite è eseguita tramite un software CAD e sfruttando un software di modellazione 3D (come 3DSmax o Blender) risulta immediato passare dal modello CAD al modello Wavefront a facce triangolari della superficie esterna del satellite.

La prima operazione che viene svolta nella funzione è leggere il file OBJ (con il relativo file MTL) tramite la funzione *read_wobj* [4], che restituisce in output una struttura contenente tutte le informazioni riguardanti la geometria e i materiali utilizzati (nel caso specifico di questa funzione non è rilevante il tipo di materiale che

viene utilizzato ma è fondamentale conoscere i coefficienti *Ca*, *Cs* e *Cd* relativi ai materiali di cui sono composte le facce), un esempio è riportato in Figura **4.3**.

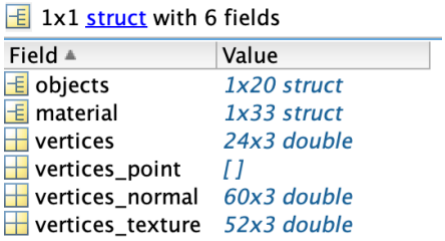

Figura 4.3 Struttura MATLAB dove sono salvati i dati relativi alla geometria e ai materiali del modello 3D.

Generata la struttura, si procede estraendo le informazioni relative alla geometria del modello 3D ed immagazzinando i dati in variabili con nomi più intuitivi (da riga 14 a 25 del codice).

I coefficienti *Cs* e *Cd* presenti nella sottostruttura "*material*" (Figura **4.3**) vengono immagazzinati in due vettori indicizzati in modo concorde agli indici dei triangoli delle facce, tale per cui ad ogni triangolo corrisponda il relativo coefficiente.

Il passo successivo è definire la dimensione della sorgente di luce bidimensionale (di cui si è già parlato al paragrafo 4.1). Come già accennato, la sorgente deve essere abbastanza grande da contenere tutta la superficie esposta del satellite, da qualunque angolazione, ma non troppo grande da dover richiedere un numero di pixel troppo elevato per ottenere un risultato accurato. Infatti, si è visto che più è piccola la dimensione del satellite rispetto alla sorgente, più pixel saranno necessari per ottenere una rappresentazione accurata.

Sulla base di alcune prove eseguite, si è dunque deciso di porre il lato della sorgente uguale a due volte la distanza del vertice più lontano dal baricentro del modello caricato.

Una volta definita la dimensione ottimale dell'array, si procede costruendo una matrice di pixel (il cui numero massimo viene fornito in input alla funzione) dove il centro di ognuno di essi viene espresso nel S.d.R solidale al modello.

Dopo aver inizializzato le variabili *Fsrp* (ovvero il coefficiente $\overrightarrow{F_{SRP}}/P_{SRP}$) e *ang* (la combinazione di angolo di azimut ed elevazione) vengono introdotti due cicli for annidati per l'azimut (che varierà da -180° a +180°) e per elevazione (che varierà da -90° a +90°), dove il passo è stato definito in ingresso dall'utente.

Per ridurre al minimo i tempi di calcolo si è deciso di parallelizzare il ciclo for relativo all'angolo di azimut.

Infatti, è possibile inizializzare un *parpool* di MATLAB che, in base al numero di core della CPU del calcolatore sul quale si sta lavorando che si desiderano mettere a disposizione, suddivide il ciclo for in un numero di cicli pari al numero di core configurati e valuta contemporaneamente ognuno di essi, riducendo notevolmente i tempi di calcolo.

Ad ogni iterazione del ciclo viene calcolato il versore $\hat{S}$ (uguale per ogni pixel essendo la sorgente bidimensionale).

Le coordinate dei pixel vengono quindi espresse in un S.d.R. ruotato in azimut ed elevazione rispetto al sistema di riferimento solidale al corpo.

Successivamente, attraverso due ulteriori cicli for annidati all'interno dei due cicli per azimut ed elevazione viene ricercata la potenziale intersezione tra il raggio proiettato dal centro dal centro del pixel *ij* e la superficie del modello.

Per valutare la presenza, o meno, di una intersezione, è stata utilizzato MT implementato su MATLAB attraverso la funzione *TriangleRayIntersection* [9].

La funzione acquisisce in ingresso l'origine del raggio (in questo caso il centro del pixel), la direzione del raggio (-$\hat{S}$ in questo caso) e i tre vertici del triangolo rispetto al quale si desidera calcolare la potenziale intersezione.

È possibile applicare la funzione a più triangoli contemporaneamente se in ingresso vengono fornite tre matrici di dimensione Nx3 (dove N è il numero di triangoli) contenenti le coordinate dei vertici.

Da quanto appena affermato è dunque evidente la ragione per la quale il modello 3D in ingresso deve essere definito come una composizione di triangoli.

La funzione *TriangleRayIntersection* restituisce in output un vettore logico (di 0 e 1) di lunghezza pari ad N che indica quali sono stati intercettati (1) e quali no (0).

Per chiarire il risultato di *TriangleRayIntersection* viene mostrata la Figura 4.4 dove si possono osservare in prospettiva i raggi dell'array di pixel proiettati su un satellite quando l'azimut è 30° e l'elevazione è 60° nel caso di 1000 pixel.

I raggi che intercettano lo s/c sono colorati magenta, i raggi che passano senza intersecare la superficie sono colorati di rosso.

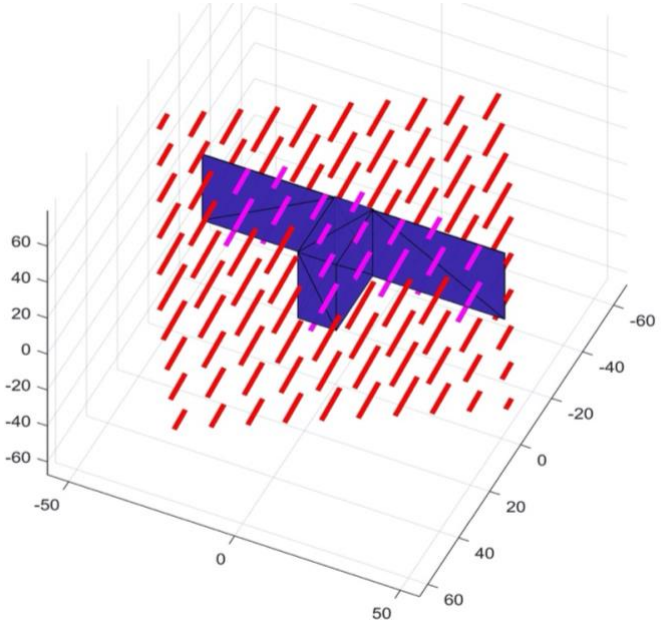

Figura 4.4 Immagine 3D del satellite con i raggi proiettati dalla sorgente di luce da azimut = 30° ed elevazione = 60°, i raggi che intercettano la superficie sono di colore magenta, i raggi che passano senza intersecare il satellite sono rosso.

Per simulare un caso reale, nel quale alcuni triangoli, seppur intercettati dal raggio, sono in ombra rispetto alla sorgente di luce, (come già accennato in 4.1) vengono esclusi dal vettore logico tutti i valori che corrispondono a un triangolo il cui baricentro ha distanza maggiore rispetto alla distanza del baricentro del triangolo più vicino all'origine del raggio.

A questo punto si possono verificare due casi distinti:

1. il triangolo intercettato è unico,
2. i triangoli intercettati sono più di uno (infatti è possibile che, essendo le facce del modello espresse come composizione di triangoli, un raggio intercetti uno spigolo comune a 2 o più triangoli e che la loro distanza sia uguale rispetto all'origine del raggio)

Nel caso 1, è immediato calcolare coefficiente $\overrightarrow{F_{SRP}}/P_{SRP}$ tramite (3.2), estrapolando il *Cs* e il *Cd* del triangolo e calcolando l'angolo fra la normale uscente del triangolo e direzione del raggio (θ).

Nel caso 2, θ viene calcolato rispetto alla media delle normali uscenti di tutti i triangoli e *Cs* e *Cd* diventano la media dei *Cs* e dei *Cd* dei triangoli intercettati.

Per chiarire come opera l'algoritmo ad ogni iterazione è possibile generare una matrice di pixel che simuli l'osservazione del satellite dal punto di vista della sorgente luminosa.

Ogni elemento della matrice riporta il modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$ del raggio del relativo pixel.

È possibile renderizzare la matrice in una immagine digitale con gli stessi pixel del numero di elementi della matrice, nella quale ogni pixel viene colorato in base al valore numerico dell'elemento corrispondente.

Un esempio del risultato per un CubeSat 6U nel caso di azimut uguale a 120° ed elevazione pari a 60° è riportato in Figura 4.5.

All'immagine è stata aggiunta una colorbar che associa il colore del pixel al valore del coefficiente $\overrightarrow{F_{SRP}}/P_{SRP}$.

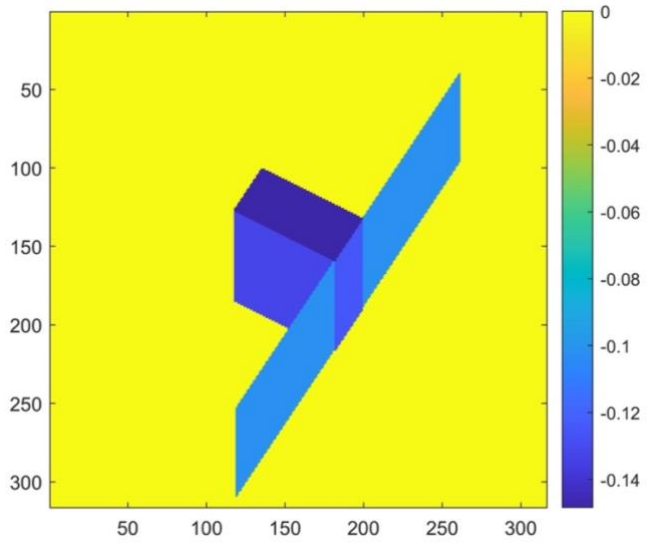

Figura 4.5 Immagine renderizzata del satellite per azimut = 120° ed elevazione = 60°, l'intensità di SRP aumenta dal giallo verso il blu.

Successivamente, nella funzione, le variabili *Fsrp* e *ang* calcolate per ogni azimut ed elevazione vengono redistribuite in due array più facili da richiamare durante la scrittura del file.

Infine, viene aperto il file con il nome fornito in ingresso e sulla base dell'*header* fornito viene generata la stringa di intestazione e all'interno di un ciclo for viene scritta ogni riga del file.

4.2 Numero di pixel ottimale

Al fine di trovare il numero ottimale di pixel da fornire in ingresso alla funzione, trovando un compromesso fra l'accuratezza del risultato finale e i tempi di calcolo dell'algoritmo, sono state eseguite alcune prove con un numero di pixel da 100.000 a 1.000.000.

1 milione è il limite di pixel considerato, oltre il quale i tempi di calcolo aumentano a tal punto da non essere compatibili con lo scopo di questo lavoro di tesi.

Per valutare come varia l'accuratezza sul risultato finale in base al numero di pixel si è deciso di calcolare il massimo errore sulla norma commesso dai vari file SPAD, rispetto al file generato con 1 milione di pixel generati per il CubeSat.

I risultati di questa analisi sono riportati in Figura 4.6.

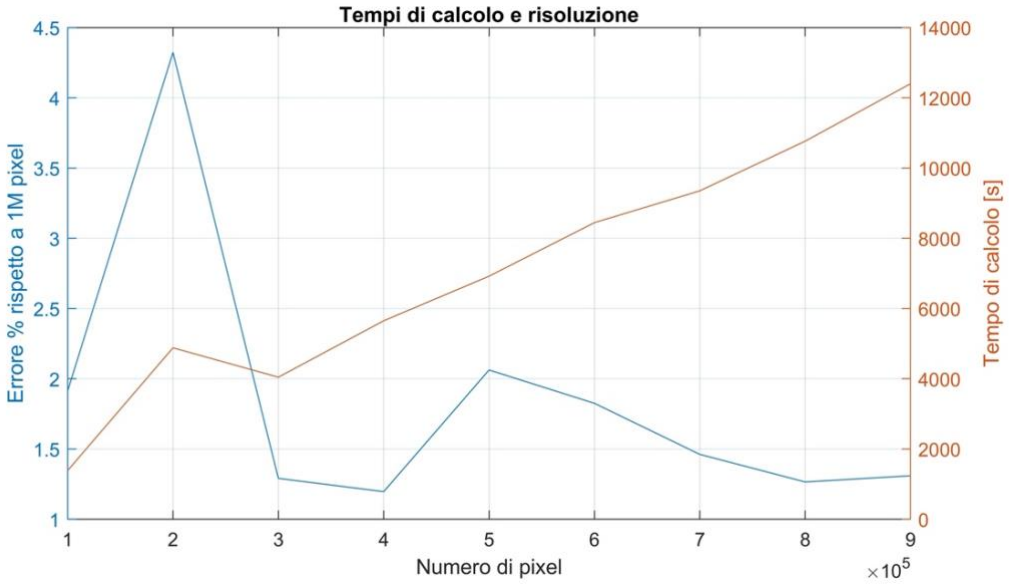

Figura 4.6 Rapporto tra tempi di calcolo dell'algoritmo ed errore percentuale commesso rispetto a 1 milione di pixel.

Come si può notare, un numero di pixel pari a 300.000 garantisce una soluzione con un errore massimo sulla norma inferiore all'1.5% e un tempo di calcolo totale pari a circa 40000 secondi (1*h* 36 *min* 6 *sec*).

4.3 Confronto su un satellite di forma cubica

Per effettuare la prima validazione della funzione per il modello SPAD complesso si è deciso di confrontare il file generato nel caso di un modello 3D cubico con il file generato dalla funzione analitica per satelliti di forma cubica trattata nel paragrafo 3.7.

I dati forniti in ingresso alla funzione per il file SPAD complesso sono contenuti in Tabella 4.1, mentre i dati forniti in ingresso a *SPADcubical* sono elencati in Tabella 4.2.

Tabella 4.1 Parametri inseriti in ingresso alla funzione per il modello SPAD complesso nel caso di un modello 3D cubico

PARAMETRO	VALORE	DESCRIZIONE
objname	"cube1.obj"	Modello 3D del cubo costruito come composizione di triangoli con area frontale di 1 m^2.
relativo file MTL	"cube1.mtl"	*Cs*= 0, *Cd*= 0 per ogni triangolo di ogni faccia.
$\boldsymbol{pixels}$	300.000	Numero di pixel in cui viene divisa la sorgente di luce bidimensionale
$\Delta\boldsymbol{\delta}$	5°	Passo Elevazione
$\Delta\boldsymbol{\lambda}$	5°	Passo Azimut
$\boldsymbol{filename}$	"$Cube\_SPADp300k.txt$"	Nome del file SPAD dove vengono salvati i risultati dell'algoritmo

Tabella 4.2 Parametri inseriti in ingresso alla funzione SPADcubical.

PARAMETRO	VALORE	DESCRIZIONE
Cs	[0 0 0 0 0 0]	Valori di *Cs* per ogni faccia
Cd	[0 0 0 0 0 0]	Valori di *Cd* per ogni faccia
$\boldsymbol{\Delta\delta}$	5°	Passo Elevazione
$\Delta\boldsymbol{\lambda}$	5°	Passo Azimut
Aface	1 m^2	Area frontale del cubo
$\boldsymbol{filename}$	"$Cube\_SPADanalytical.txt$"	Nome del file dove vengono salvati i risultati

Avendo scelto un numero di pixel pari a 300.000, per le considerazioni fatte al paragrafo 4.2, è comprensibile che tra i due file sia presente un errore.

Per effettuare il confronto si è deciso di calcolare l'errore sulla componente X, Y, Z e sul modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$ espressa nel S.d.R. solidale con il corpo.

Per poterli analizzare meglio, i risultati del confronto sono stati disposti in 4 matrici dove per ogni riga varia l'elevazione e per ogni colonna varia l'azimut.

È possibile renderizzare le matrici in 4 immagi digitali che hanno un numero di pixel pari al numero di elementi della matrice, ogni pixel viene colorato con una intensità che dipende dal valore numerico dell'errore dell'elemento corrispondente (Figura 4.7), accanto a ogni immagine è stata inserita una colorbar che chiarisce quale sia l'intensità relativa ad ogni colore.

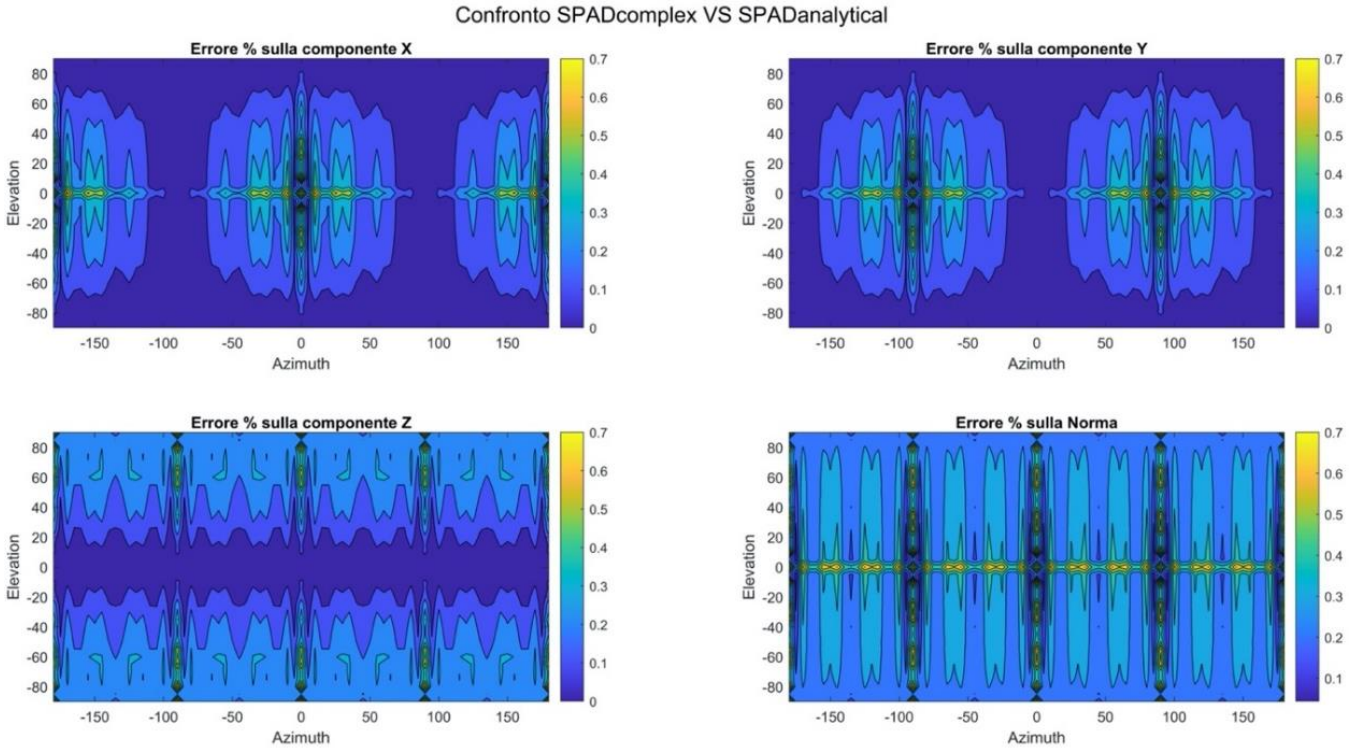

Figura 4.7 Errore percentuale commesso sulla componente X, Y, Z di $\overrightarrow{\boldsymbol{F_{SRP}}}/\ \boldsymbol{P_{SRP}}$ e sulla norma del file di SPADcomplex con il file di SPADanalytical.

Come si può notare l'errore è complessivamente molto basso, nel caso della norma non supera lo 0.7 % e solo in particolari angolazioni.

L'origine di tale errore è probabilmente dovuta al fatto che utilizzando l'algoritmo di ray-tracing si ottiene, inevitabilmente, un risultato approssimato dato che si tratta di

un modello numerico, mentre con la funzione *SPADcubical* il risultato è calcolato in modo analitico, dunque esatto.

4.4 Confronto per un CubeSat 6U

Al fine di effettuare una ulteriore validazione della funzione implementata per il modello SPAD (Files C) nel caso di un satellite reale si è deciso di considerare ancora una volta il caso del C/S trattato in precedenza (paragrafo 3.6).

Per prima cosa, sono stati confrontati il file generato dalla funzione con il file SPAD estrapolato dal software MONTE con lo stesso procedimento utilizzato per il confronto sul satellite cubico.

Al fine di scegliere il numero ottimale di pixel da immettere in ingresso alla funzione per ottenere il file SPAD che verrà elaborato da GMAT per ottenere il contributo di SRP sulla traiettoria si è deciso di calcolare l'errore massimo in percentuale sulla norma di $\overrightarrow{F_{SRP}}/P_{SRP}$ dei file SPAD generati con un numero di pixel da 100.000 a 1.000.000 rispetto al file SPAD elaborato da MONTE per il satellite.

I risultati sono esposti in Figura 4.8.

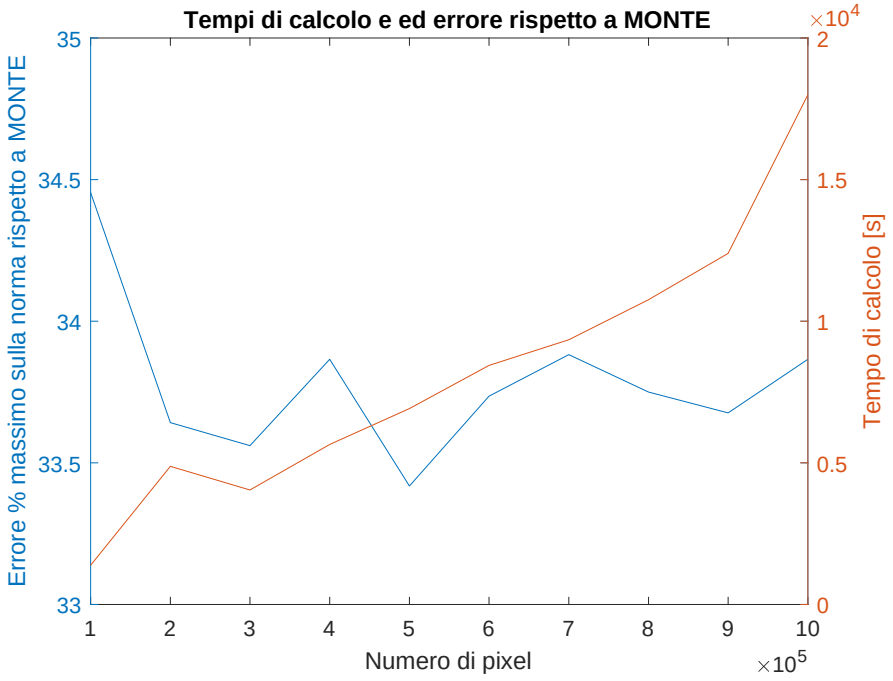

Figura 4.8 Tempi di calcolo dell'algoritmo ed errore dei file SPAD generati rispetto al file SPAD estrapolato da MONTE.

L'origine di un errore così elevato sarà approfondita in seguito, per ora è sufficiente notare che per un numero di pixel maggiore di 300.000 l'errore commesso rispetto al file di MONTE tende a oscillare attorno al 33.75%.

Dunque, trovando un compromesso fra i tempi di calcolo e l'accuratezza del risultato finale per le successive analisi si è deciso di scegliere il valore di 400.000 pixel.

I parametri inseriti in ingresso alla funzione *SPADcomplex* sono riportati in Tabella 4.3.

La Figura **4.9** espone i risultati sull'errore ottenuto confrontando i due file SPAD sulla componente X, Y, Z e sulla norma di $\overrightarrow{F_{SRP}}/P_{SRP}$ per ogni angolazione di azimut ed elevazione.

Tabella 4.3 Parametri inseriti in ingresso alla funzione per il modello 3D del CubeSat 6U.

PARAMETRO	VALORE	DESCRIZIONE
objname	"CubeSat.obj"	Modello 3D del CubeSat
relativo file MTL	"CubeSat.mtl"	per le facce frontali dei pannelli solari: *Cs* = 0.23, *Cd* = 0; per le facce posteriori dei pannelli solari: *Cs* = 0; *Cd* = 0.75, per il corpo del C/S: *Cs* = 0, *Cd* = 0.75.
$\boldsymbol{pixels}$	400.000	Numero di pixel in cui viene divisa la sorgente di luce bidimensionale
$\Delta\boldsymbol{\delta}$	5°	Passo Elevazione
$\Delta\boldsymbol{\lambda}$	5°	Passo Azimut
$\boldsymbol{filename}$	"$SPAD\_CubeSat.txt$"	Nome del file SPAD dove vengono salvati i risultati dell'algoritmo

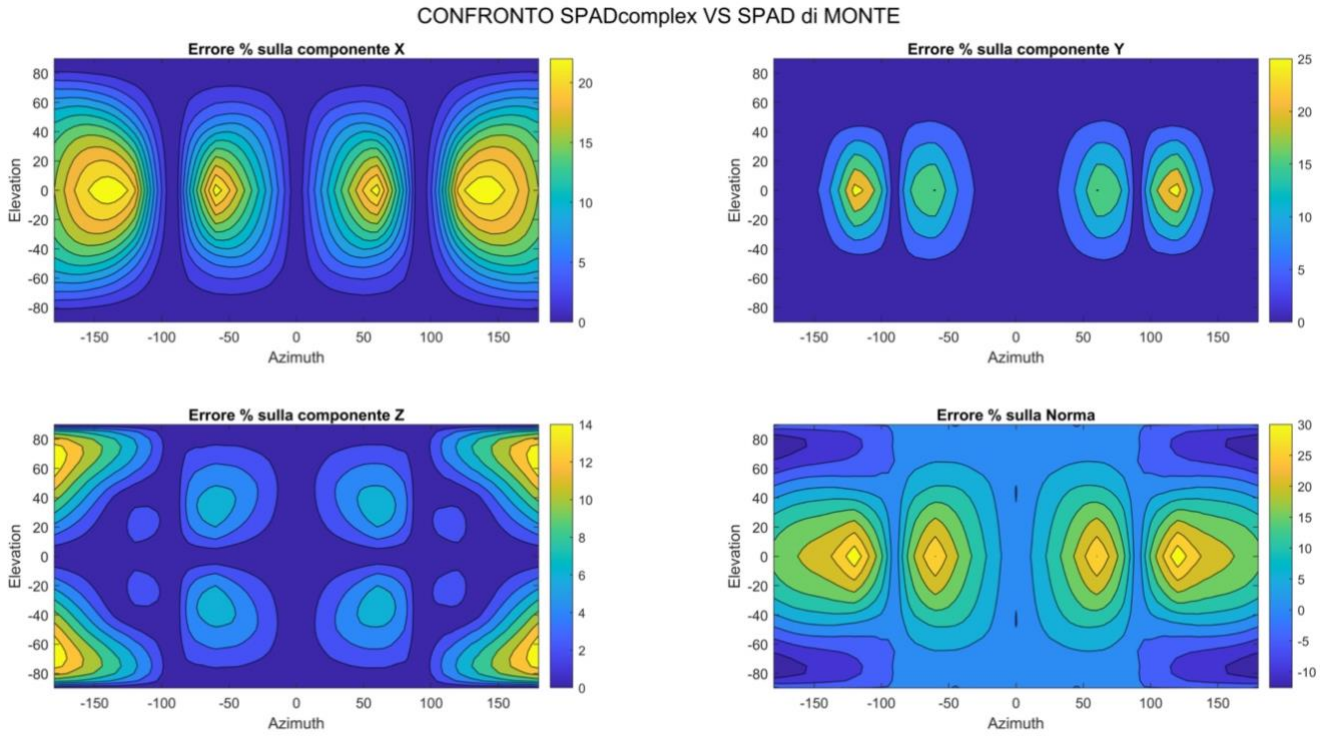

Figura 4.9 Errore percentuale commesso sulla componente X, Y, Z di $\overrightarrow{F_{SRP}}$ / P_{SRP} e sulla norma del file di SPADcomplex con il file SPAD di MONTE**.**

L'errore percentuale commesso è rilevante (in alcune zone addirittura superiore al 30%).

La ragione è imputabile al fatto che il file SPAD generato da MONTE si basa su di un modello a panelli che non considera il fenomeno dello *shadowing* tra le varie parti del satellite. Il fenomeno introdotto verifica sostanzialmente quando una parte dello s/c ne oscura un'altra.

Infatti, nel calcolo di $\overrightarrow{F_{SRP}}/P_{SRP}$, MONTE considera anche il contributo delle facce nascoste rispetto alla sorgente di luce mentre la funzione *SPADcomplex* no.

Per dimostrare che gran parte dell'errore fra i due modelli è causato dallo *shadowing* è stata considerata una zona dove l'errore è molto elevato.

È stato preso in esame il caso in cui azimut = 120° ed elevazione = 0° (errore: 32.5%).

Ciò che vede un osservatore solidale con la sorgente luminosa da questa particolare angolazione è riportato in Figura 4.10.

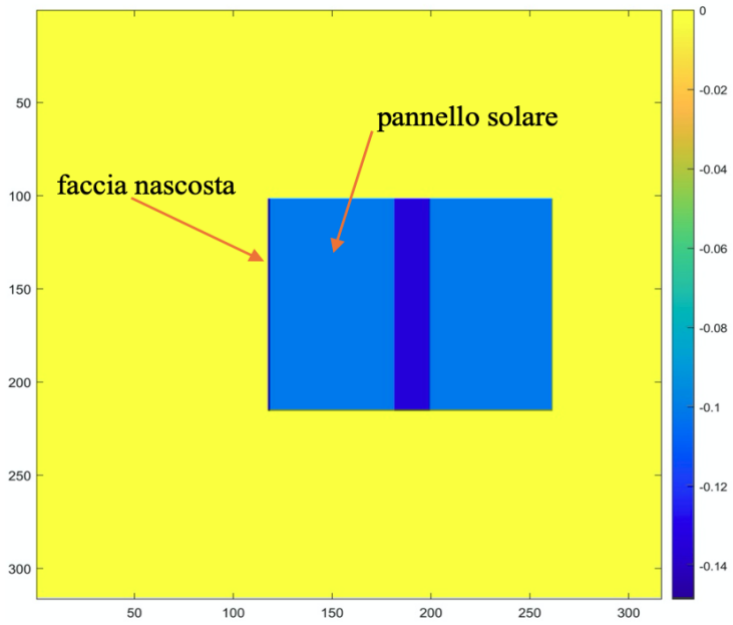

Figura 4.10 Immagine renderizzata del CubeSat 6U per azimut = 120° ed elevazione = 0°, l'intensità di SRP aumenta dal giallo verso il blu.

Come si può notare, una delle facce del corpo dello s/c è quasi completamente coperta dal pannello solare (PS).

Per questa angolazione il modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$ calcolato dall'algoritmo è

$$\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{(az=120°,el=0°)} = 0.193613\ [m^2] \qquad \textbf{(4.7)}$$

Nel file di MONTE il modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$ è

$$\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{MONTE} = 0.264642\ [m^2] \qquad \textbf{(4.8)}$$

Il contributo della faccia coperta può essere calcolato con lo stesso metodo utilizzato in (3.4)per calcolare $\overrightarrow{F_{SRP}}/P_{SRP}$ per una faccia del cubo, ovvero

$$\overrightarrow{F_{SRP}}/P_{SRP} = -A_{face}\ \cos\theta\left\{(1-Cs)\hat{S} + 2(Cs\cos\theta + \frac{1}{3}Cd)\hat{N}\right\}$$

dove A_{face} è l'area della faccia nascosta ($0.2 \times 0.3\ m^2$), θ è l'angolo tra la normale uscente della faccia nascosta ($\widehat{N} = [0\ 1\ 0]$) e $\widehat{S}$ ($= [-0.5\ 0.866\ 0]$), il vettore che definisce la direzione dell'origine dell'array di pixel rispetto al centro del S.d.R. Body-Fixed, $Cs\ (= 0)$ e $Cd\ (=\ 0.75)$ sono i coefficienti di luce riflessa speculare e diffusa della faccia nascosta.

Tenendo conto che

$$\cos\theta = \widehat{S} \times \widehat{N}$$

dove × è il prodotto scalare tra vettori, si possono sostituire i valori come in (4.9)

$$\overrightarrow{F_{SRP}}\Big/_{P_{SRP}} = 0.06 \cdot \left\{\begin{bmatrix}1\\0\\0\end{bmatrix} \times \begin{bmatrix}-0.5\\0.866\\0\end{bmatrix}\right\} \cdot \left\{(1-0) \cdot \begin{bmatrix}-0.5\\0.866\\0\end{bmatrix} + 2 \cdot (0 + {}^{1}/_{3} \times 0.75) \cdot \begin{bmatrix}1\\0\\0\end{bmatrix}\right\} \quad \textbf{(4.9)}$$

per trovare il vettore risultante riportato in (4.10)

$$\overrightarrow{F_{SRP}}\Big/_{P_{SRP}} = \begin{bmatrix}0.02598\\0.0709794\\0\end{bmatrix} \quad \textbf{(4.10)}$$

il cui modulo è

$$\left|\overrightarrow{F_{SRP}}\Big/_{P_{SRP}}\right|_{(faccia\ nascosta)} = 0.078585\ [m^2] \quad \textbf{(4.11)}$$

Dunque, l'errore percentuale che si commette non considerando lo *shadowing* si può calcolare come

$$\mathcal{E}_{\%}^{\ Shadowing} = \frac{0.078585}{0.264642} \times\ 100 = 29.694833\ \%. \quad \textbf{(4.12)}$$

Tale valore è relativamente simile all'errore commesso confrontando direttamente i due file in questa particolare angolazione (32.5%, ricavato dalla Figura **4.9**).

Tuttavia, per effettuare una analisi accurata, è necessario considerare che, come si vede in Figura 4.10, non tutta la faccia del corpo del satellite è coperta dal PS.

La parte di faccia non oscurata offre un contributo pari a

$$\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{(parte\ di\ faccia\ esposta)} = 0.001692[m^2] \qquad \textbf{(4.13)}$$

Sommando il modulo calcolato dalla funzione (4.7) con il contributo della faccia nascosta (4.11) e togliendo il contributo della parte di faccia esposta (4.13) si trova

$$\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{(no\ shadowing)} = 0.270509\ [m^2] \qquad \textbf{(4.14)}$$

Dunque, sommando al risultato di *SPADcomplex* il contributo della faccia nascosta, l'errore percentuale relativo sulla norma, per la particolare angolazione considerata (azimut = 0°, elevazione = 0°), risulta

$$100 \times \frac{\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{(no\ shadowing)} - \left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{MONTE}}{\left|\overrightarrow{F_{SRP}}/P_{SRP}\right|_{MONTE}} = 0.22\ \% \qquad \textbf{(4.15)}$$

Sulla base delle valutazioni eseguite per analizzare l'origine dell'errore commesso confrontando i due file SPAD , i cui risultati sono riportati in (4.12) e in(4.15 , si può dire che la funzione *SPADcomplex* fornisce un risultato più accurato del modello a pannelli elementari di MONTE dal momento che esso non considera il fenomeno dello *shadowing.*

4.5 Confronto tra le traiettorie su MATLAB utilizzando SPICE

Visti i risultati del paragrafo precedente è possibile confrontare la traiettoria propagata da GMAT con il modello SPAD con la traiettoria propagata da MONTE utilizzando il modello a pannelli elementari.

La traiettoria di MONTE è stata fornita in ingresso a questa analisi, mentre la traiettoria di GMAT è stata propagata con le variabili di integrazione riportate in Tabella 4.4.

Tabella 4.4 Parametri di propagazione della traiettoria del satellite inseriti su GMAT considerando il file SPAD generato dalla funzione *SPADcomplex*

CONFIGURAZIONE PROPAGAZIONE		PARAMETRI DI INTEGRAZIONE	
Formato Epoca	TDBGregorian	Tipo di integratore	PrinceDormand78
Epoca Iniziale	29 Jun 2020 00:00:00.000	Passo Minimo	1e-8 s
Sistema di Riferimento	EarthMJ2000Eq	Passo Massimo	60 s
Formato Stato Iniziale	Cartesiano	Accuratezza	1e-14
X	-95505.35424576626 m	Modello di gravità esteso	No
Y	-189654.180051302 m	Punti di Massa	Terra, Luna
Z	-61502.33776439037 m	Forze non gravitazionali	SRP (SPAD file)
Vx	-0.2450115477240185 m/s	Tempo di Propagazione	10 giorni
Vy	-1.151377154772402 m/s		
Vz	-0.477385138430765 m/s		
Massa del satellite	13.3669 kg		

Come già fatto nel paragrafo 3.6, per fare sì che i due software sfruttino lo stesso modello gravitazione sono stati modificati i seguenti parametri di GMAT:

- Le effemeridi planetarie sono state cambiate da DE405 (predefinite di GMAT) a quelle più aggiornate DE430[3] poiché la traiettoria del C/S propagata in MONTE utilizzava quest'ultime nel modello dinamico.
- Il GM[4] della Terra è stato modificato utilizzando un valore più accurato[2]: μ = 398600.435436 km^3/s^2
- Il GM della Luna è stato modificato utilizzando un valore più accurato[2]: μ = 4902.8000066 km^3/s^2

I risultati del confronto sono riportati in Figura 4.11.

[3] È possibile trovare i file con le effemeridi planetarie DE405 e DE430 e i valori accurati dei GM di Terra e Luna nel sito riportato in [1].
[4] GM: costante di gravitazione planetaria, è il prodotto di G: costante di gravitazione universale e M: massa del corpo celeste.

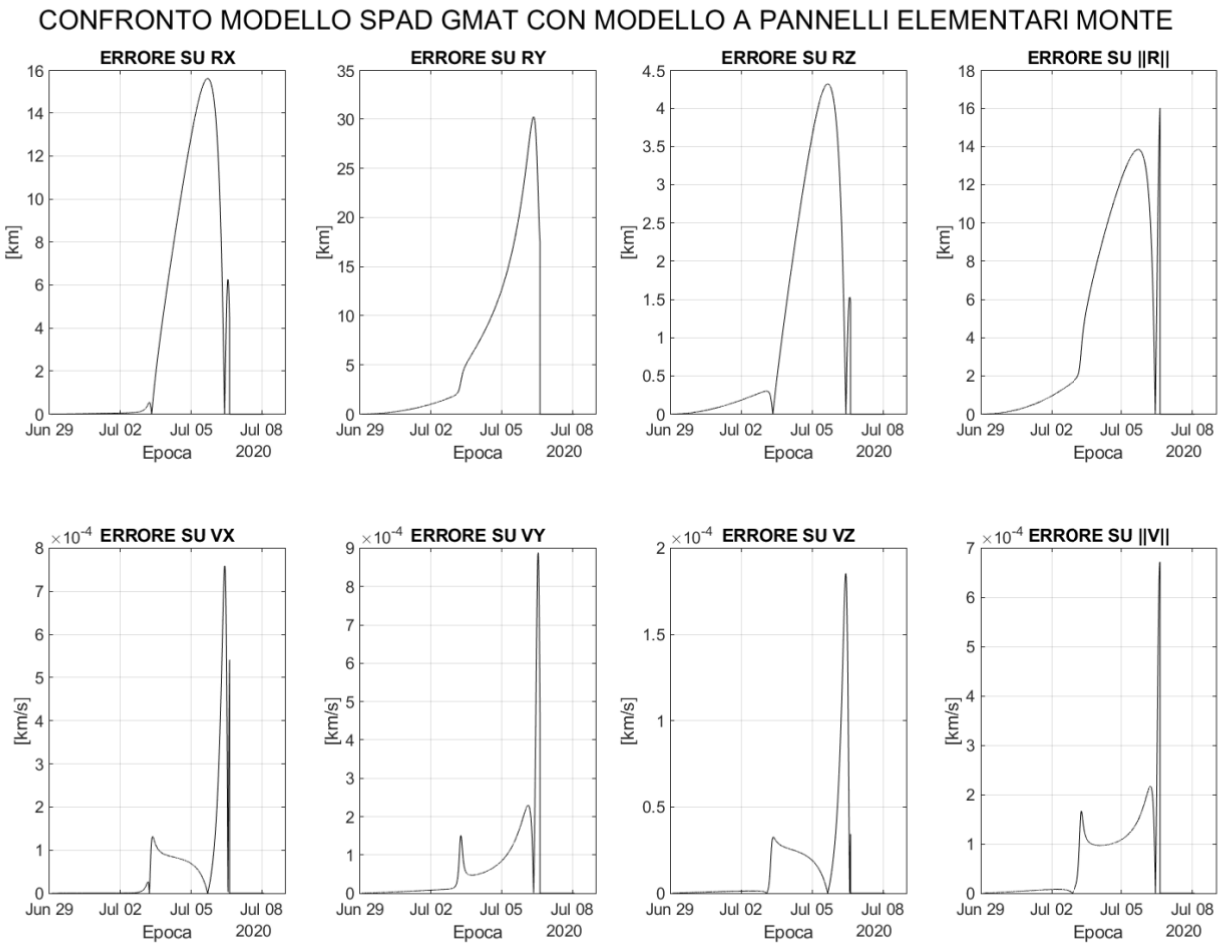

Figura 4.11 Risultati del confronto tra la traiettoria del C/S propagata su GMAT con il file SPAD della funzione *SPADcomplex* e la traiettoria propagata da MONTE con il modello a pannelli elementari.

L'errore allo stato iniziale è nullo, ma si amplifica fino a raggiungere il valore massimo di 16 km sul modulo del raggio e circa $7 \times 10^{-4}\ km/s$ nel modulo della velocità orbitale.

Sulla base delle valutazioni fatte al paragrafo 4.4, è possibile affermare che parte di questo errore è commesso dal modello a pannelli elementari di MONTE, il quale, come è già stato affermato, a differenza dell'algoritmo di ray-tracing, non tiene conto dello *shadowing*, necessario da considerare in un caso reale ai fini del calcolo accurato del modulo e della direzione di $\overrightarrow{F_{SRP}}$.

Capitolo 5 : Conclusioni

L'obiettivo del lavoro di tesi, dichiarato nel Capitolo 1, è stato implementare delle funzioni MATLAB per il modello SPAD che consentano di valutare con il massimo grado di accuratezza possibile il contributo del disturbo relativo a SRP nella propagazione di una traiettoria sul software GMAT.

Le funzioni *SPADspherical* (Files A) e *SPADcubical* (Files B) implementate, generano file SPAD calcolando $\overrightarrow{F_{SRP}}/P_{SRP}$ in modo analitico, i loro principali punti di forza sono:

- seguire una logica semplice, dunque essere relativamente facili da implementare,
- richiedere tempi di calcolo estremamente bassi rispetto alla funzione *SPADcomplex*,
- fornire in output un risultato esatto (eseguendo un calcolo analitico).

Tuttavia, presentano lo svantaggio di fornire un risultato esatto solo per satelliti di forme elementari, ovvero: sfere (*SPADspherical*) o cubi (*SPADcubical*).

Implementare una funzione del modello SPAD analitica per un satellite con una geometria più complessa, significherebbe togliere il vantaggio della semplicità e, ad ogni modo, sarebbe difficile generalizzare la funzione per una geometria qualsiasi.

SPADcomplex (Files C) è stata implementata per generare file SPAD calcolando $\overrightarrow{F_{SRP}}/P_{SRP}$ con un metodo numerico che si basa su un algoritmo di ray-tracing, questo comporta i seguenti svantaggi:

- fornire in output un risultato approssimato rispetto a un calcolo analitico
- l'accuratezza del risultato finale dipende dalla risoluzione scelta e come si è visto in Figura 4.6 i tempi di calcolo aumentano notevolmente all'aumentare del numero di pixel scelto
- i tempi di calcolo sono in ogni caso molto maggiori rispetto a quelli delle funzioni analitiche

Tuttavia, a differenza delle due funzioni analitiche, *SPADcomplex* può essere utilizzata per un satellite di geometria generica, complessa a piacere.

Nel caso di un satellite che non presenta una forma elementare (sfera o cubo), l'utilizzo di *SPADcomplex* fornisce un risultato più accurato di *SPADspherical* e *SPADcubical*, dal momento che è in grado di considerare la reale geometria del satellite.

Inoltre, è opportuno ricordare che utilizzando l'algoritmo di ray-tracing è possibile tenere conto del fenomeno dello *shadowing*.

5.1 Conclusioni tecniche

I risultati della funzione *SPADspherical* sono stati confrontati con quelli del modello sferico predefinito di GMAT, ottenendo:

- un errore nullo nella propagazione di un'orbita geostazionaria (Figura 3.4Figura 3.8)
- un errore molto piccolo nella propagazione della traiettoria del C/S 6U (Figura 3.7)

Dunque, è possibile affermare che essa produce un risultato accurato nel caso di un satellite di geometria sferica.

Il file SPAD generato dalla funzione *SPADcomplex* è stato confrontato con il modello a pannelli elementari trovando un errore che si aggira attorno al 30% sulla norma del coefficiente $\overrightarrow{F_{SRP}}/P_{SRP}$ per un range di angolazioni piuttosto ampio (come si può notare in Figura **4.9**).

In particolare, il modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$ calcolato dal modello a pannelli elementari di MONTE risulta del 30% maggiore rispetto quello calcolato utilizzando l'algoritmo di ray-tracing.

Tale differenza è dovuta principalmente allo *shadowing* e per dimostrarlo, nel paragrafo 4.4, è stato calcolato il contributo, dato da una faccia nascosta, al modulo complessivo di $\overrightarrow{F_{SRP}}/P_{SRP}$ (per una particolare angolazione della sorgente di luce) e di

conseguenza è stato valutato l'errore che si commette considerando che la faccia sia illuminata nonostante sia coperta dal PS:

$$\mathcal{E}_{\%}^{\ Shadowing} = 29.694833\ \%.$$

Dunque, la maggior parte dell'errore fra i due file SPAD è dovuto al fatto che il modello a pannelli elementari di MONTE non considera il fenomeno dello *shadowing* e quindi sovrastima il modulo di $\overrightarrow{F_{SRP}}/P_{SRP}$.

Sulla base di quanto detto fin qui, si può affermare che l'algoritmo di ray-tracing produce un risultato più accurato del modello a pannelli di MONTE.

Nel lavoro presentato in questo documento, *SPADcomplex* è stata applicata a un CubeSat 6U, che presenta una geometria relativamente semplice, per fare si che i risultati siano facilmente interpretabili, ma è importante evidenziare che attraverso la funzione sarebbe teoricamente possibile scrivere il file SPAD di un satellite con una geometria anche molto articolata, come, ad esempio, quella del Mars Reconnaissance Orbiter [5] (Figura 5.1).

Figura 5.1 Fotografia del Mars Reconnaissance Orbiter [5].

Capitolo 6 : Appendice

Files A funzione *SPADspherical*.

```
1. function
   SPADspherical(Cs,Cd,Apixel,step_az,step_el,header,filename)
2.
3. % calcolo del coefficiebte CR
4. Cr = 1 + Cs + 2/3*Cd;
5.
6. format long
7. % Inizializzazione delle variabili
8. lambda = -180 : step_az : 180;
9. delta = -90 : step_el : 90;
10. Fsrp = zeros(length(lambda)*length(delta),3);
11. N    = zeros(length(lambda)*length(delta),3);
12. ang  = zeros(length(lambda)*length(delta),2);
13. FMAG = -Apixel * Cr;
14.
15. % Iterazione per azimut ed elevazione
16. t = 0;
17. for k1 = lambda
18.     for k2 = delta
19.         % Aumento dell'indice
20.         t = t+1;
21.         % Calcolo della direzione del Sole
22.         N(t,:) = [cosd(k2)*cosd(k1), cosd(k2)*sind(k1),
    sind(k2) ];
23.         % Calcolo del coefficiente
24.         Fsrp(t,:) = FMAG*( N(t,:));
25.         % Vettore degli angoli
26.         ang(t,:) = [k1 k2];
27.     end
28. end
29.
30. % Creazione di un nuovo indice su cui iterare
31. t = 1: length(lambda)*length(delta);
32.
33. % Generazione della stringa di intestazione
34. file_header = "";
35. file_header = file_header + "Version            : " +
    header.version + " \n";
36. file_header = file_header + "System             : " +
    header.system + " \n";
```

```
37. file_header = file_header + "Analysis Type      : " +
    header.analysis + " \n";
38. file_header = file_header + "Pixel Size         : " +
    header.pixelsize + " \n";
39. file_header = file_header + "Spacecraft Size    : " +
    header.scsize + " \n";
40. file_header = file_header + "Pressure           : " +
    header.pressure + " \n";
41. file_header = file_header + "Center of Mass     : " +
    header.com + " \n";
42. file_header = file_header + "Current time       : " +
    header.date + " \n\n";
43. file_header = file_header + "Motion             : 1 \n";
44. file_header = file_header + " Name               : Azimuth
    \n";
45. file_header = file_header + " Method             : Step
    \n";
46. file_header = file_header + " Minimum            : -180
    \n";
47. file_header = file_header + " Maximum            : +180
    \n";
48. file_header = file_header + " Step               : " +
    step_az + " \n";
49. file_header = file_header + "Motion             : 2 \n";
50. file_header = file_header + " Name               :
    Elevation \n";
51. file_header = file_header + " Method             : Step
    \n";
52. file_header = file_header + " Minimum            : -90 \n";
53. file_header = file_header + " Maximum            : +90 \n";
54. file_header = file_header + " Step               : " +
    step_el + " \n";
55. file_header = file_header + ":END \n\n";
56. file_header = file_header + "Record count       : " +
    length(Fsrp) + " \n\n";
57. file_header = file_header + "  degrees         degrees
    m^2                  m^2                    m^2
    \n";
58. file_header = file_header + "-----------     ----------
    ------------------    ------------------    ---------------
    ---     \n";
59.
60. % Apertura del file
61. fileID = fopen(filename, 'w');
62. file_header_string = sprintf("%s", file_header);
63. fprintf(fileID, file_header_string, '-append');
64.
```

```
% Iterazione sull'indice per scrivere le righe del corpo
del file
for k = t
    str(k) = sprintf("%f",ang(k,1)) + '     ' +
sprintf("%10f",ang(k,2))  + '     '+
sprintf("%.15f",Fsrp(k,1)) + '     '
+sprintf("%.15f",Fsrp(k,2)) + '     '
+sprintf("%.15f",Fsrp(k,3)) + '\n' ;
    fprintf(fileID, str(k), '-append');
end

% Chiusura del file
fclose(fileID);

end

```

Files B funzione *SPADcubical*.

```
1.  function SPADcubical(Cs,Cd,Aface,step_az,step_el,filename)
2.
3.  % Mappatura dei coefficienti su ogni faccia
4.  Cs1 = Cs(1);
5.  Cs2 = Cs(2);
6.  Cs3 = Cs(3);
7.  Cs4 = Cs(4);
8.  Cs5 = Cs(5);
9.  Cs6 = Cs(6);
10.
11. Cd1 = Cd(1);
12. Cd2 = Cd(2);
13. Cd3 = Cd(3);
14. Cd4 = Cd(4);
15. Cd5 = Cd(5);
16. Cd6 = Cd(6);
17.
18. % Creazione dei vettori di azimut ed elevazione
19. lambda = -180 : step_az : 180;
20. delta = -90 : step_el : 90;
21.
22. % Inizializzazione delle componenti di forza su ogni
    faccia
23.
24. Fsrp = zeros(length(lambda)*length(delta),3);
25.
26. Fsrp1 = zeros(length(lambda)*length(delta),3);
27. Fsrp2 = zeros(length(lambda)*length(delta),3);
28. Fsrp3 = zeros(length(lambda)*length(delta),3);
29. Fsrp4 = zeros(length(lambda)*length(delta),3);
30. Fsrp5 = zeros(length(lambda)*length(delta),3);
31. Fsrp6 = zeros(length(lambda)*length(delta),3);
32.
33. % Inizializzazione del vettore della direzione del Sole
34.
35. S    = zeros(length(lambda)*length(delta),3);
36.
37. % Inizializzazone del vettore degli angoli
38. ang  = zeros(length(lambda)*length(delta),2);
39.
40. % Espressioni delle normali per ogni faccia nel
    S.d.R.Body-Fixed
41.
42. N1 = [0 -1 0];
```

```
N2 = [1 0 0];
N3 = [0 1 0];
N4 = [-1 0 0];
N5 = [0 0 1];
N6 = [0 0 -1];

%Inizializzazione degli angoli tra S ed N per ogni faccia

teta1 = zeros(length(lambda)*length(delta),1);
teta2 = zeros(length(lambda)*length(delta),1);
teta3 = zeros(length(lambda)*length(delta),1);
teta4 = zeros(length(lambda)*length(delta),1);
teta5 = zeros(length(lambda)*length(delta),1);
teta6 = zeros(length(lambda)*length(delta),1);

t = 0;
for k1 = lambda
    for k2 = delta
    % Incremento dell'indice
    t = t+1;

    % Posizione del Sole
    S(t,:) = [cosd(k2)*cosd(k1), cosd(k2)*sind(k1),
sind(k2) ];

    % Calcolo di Fsrp per la facccia 1
    teta1(t) = acosd(dot(N1,S(t,:)));
    if teta1(t) >= 90
        Fsrp1(t,:) = [0 0 0];
    else
        Fsrp1(t,:) = -Aface*cosd(teta1(t))*((1-Cs1)*S(t,:)
+ 2*(Cs1*cosd(teta1(t)) + 1/3*Cd1)*N1);
    end
    % Calcolo di Fsrp per la faccia 2
    teta2(t) = acosd(dot(N2,S(t,:)));
    if teta2(t) >= 90
        Fsrp2(t,:) = [0 0 0];
    else
        Fsrp2(t,:) = -Aface*cosd(teta2(t))*((1-Cs2)*S(t,:)
+ 2*(Cs2*cosd(teta2(t)) + 1/3*Cd2)*N2);
    end
    % Calcolo di Fsrp per la faccia 3
    teta3(t) = acosd(dot(N3,S(t,:)));
    if teta3(t) >= 90
        Fsrp3(t,:) = [0 0 0];
    else
        Fsrp3(t,:) = -Aface*cosd(teta3(t))*((1-Cs3)*S(t,:)
+ 2*(Cs3*cosd(teta3(t)) + 1/3*Cd3)*N3);
```

```
87.     end
88.     % Calcolo di Fsrp per la faccia 4
89.     teta4(t) = acosd(dot(N4,S(t,:)));
90.     if teta4(t) >= 90
91.         Fsrp4(t,:) = [0 0 0];
92.     else
93.         Fsrp4(t,:) = -Aface*cosd(teta4(t))*((1-Cs4)*S(t,:)
    + 2*(Cs4*cosd(teta4(t)) + 1/3*Cd4)*N4);
94.     end
95.     % Calcolo di Fsrp per la faccia 5
96.     teta5(t) = acosd(dot(N5,S(t,:)));
97.     if teta5(t) >= 90
98.         Fsrp5(t,:) = [0 0 0];
99.     else
100.             Fsrp5(t,:) = -Aface*cosd(teta5(t))*((1-
    Cs5)*S(t,:) + 2*(Cs5*cosd(teta5(t)) + 1/3*Cd5)*N5);
101.         end
102.         % Calcolo di Fsrp per la faccia 6
103.         teta6(t) = acosd(dot(N6,S(t,:)));
104.         if teta6(t) >= 90
105.             Fsrp6(t,:) = [0 0 0];
106.         else
107.             Fsrp6(t,:) = -Aface*cosd(teta6(t))*((1-
    Cs6)*S(t,:) + 2*(Cs6*cosd(teta6(t)) + 1/3*Cd6)*N6);
108.         end
109.         % Somma dei risultati
110.         Fsrp(t,:) = Fsrp1(t,:) + Fsrp2(t,:) +
    Fsrp3(t,:) + Fsrp4(t,:) + Fsrp5(t,:) + Fsrp6(t,:);
111.         % Angoli di azimut ed elevazione per ogni
    iterazione
112.         ang(t,:) = [k1 k2];
113.         end
114.     end
115.
116.     % Creazione del nuovo indice di iterazione
117.     t = 1: length(lambda)*length(delta);
118.
119.     % Generazione della stringa di intestazione sulla
    base di header
120.     file_header = "";
121.     file_header = file_header + "Version            : "
    + header.version + " \n";
122.     file_header = file_header + "System             : "
    + header.system + " \n";
123.     file_header = file_header + "Analysis Type      : "
    + header.analysis + " \n";
124.     file_header = file_header + "Pixel Size         : "
    + header.pixelsize + " \n";
```

```
125.      file_header = file_header + "Spacecraft Size    : "
   + header.scsize + " \n";
126.      file_header = file_header + "Pressure           : "
   + header.pressure + " \n";
127.      file_header = file_header + "Center of Mass     : "
   + header.com + " \n";
128.      file_header = file_header + "Current time       : "
   + header.date + " \n\n";
129.      file_header = file_header + "Motion             : 1
   \n";
130.      file_header = file_header + " Name               :
   Azimuth \n";
131.      file_header = file_header + " Method             :
   Step \n";
132.      file_header = file_header + " Minimum            :
   -180 \n";
133.      file_header = file_header + " Maximum            :
   +180 \n";
134.      file_header = file_header + " Step               :
   " + step_az + " \n";
135.      file_header = file_header + "Motion             : 2
   \n";
136.      file_header = file_header + " Name               :
   Elevation \n";
137.      file_header = file_header + " Method             :
   Step \n";
138.      file_header = file_header + " Minimum            :
   -90 \n";
139.      file_header = file_header + " Maximum            :
   +90 \n";
140.      file_header = file_header + " Step               :
   " + step_el + " \n";
141.      file_header = file_header + ":END \n\n";
142.      file_header = file_header + "Record count       : "
   + length(Fsrp) + " \n\n";
143.      file_header = file_header + "  Azimuth
   Elevation          Force(X)              Force(Y)
   Force(Z)          \n";
144.      file_header = file_header + "  degrees
   degrees             m^2                   m^2
   m^2             \n";
145.      file_header = file_header + "-----------     ------
   ----     ------------------    ------------------    -----
   -------------     \n";
146.
147.      % Apertura del file
148.      fileID = fopen(filename, 'w');
149.      file_header_string = sprintf("%s", file_header);
```

```
150.      fprintf(fileID, file_header_string, '-append');
151.
152.      % Generazione del corpo del file con processo
   iterativo
153.      for k = t
154.          str(k) = sprintf("%f",ang(k,1)) + '     ' +
   sprintf("%10f",ang(k,2))  + '     '+
   sprintf("%.15f",Fsrp(k,1)) + '     '
   +sprintf("%.15f",Fsrp(k,2)) + '     '
   +sprintf("%.15f",Fsrp(k,3)) + '\n' ;
155.          fprintf(fileID, str(k), '-append');
156.      end
157.
158.      % Chiusura del file
159.      fclose(fileID);
160.      end
```

Files C funzione *SPADcomplex*.

```
1.  function SPADcomplex(objname,pixels,step_az,step_el,header,filename)
2.
3.  % Lettura del file OBJ
4.
5.  fwmat = read_wobj(objname);
6.
7.
8.  % Numero dei campi della struttura generata
9.  % per leggere il modello 3D (costante)
10. nfieldmat = 11;
11. nfieldobj = 4;
12.
13. nobj = length(fwmat.objects)/nfieldobj;
14. nobjmat = length(fwmat.material)/nfieldmat;
15.
16. % Estrapolazione dei dati sulla geometria del modello 3D
17.
18. vertices = fwmat.vertices;
19. faces = [];
20. normals_ind = [];
21. normals = fwmat.vertices_normal;
22. double_normals(2:2:(length(normals(:,1))*2),:) = normals;
23. double_normals(1:2:(end-1),:) = normals;
24.
25. for yr = 1:nobj
26. faces = [faces; fwmat.objects(4*yr).data.vertices];
27. normals_ind = [normals_ind; fwmat.objects(4*yr).data.normal(:,1)];
28. end
29.
30. % Mappatura dei Cs e dei Cd su ogni faccia
31. hj = 0;
32. for o = 1:nobj
33. nfacesP(o) = length(fwmat.objects(4*o).data.vertices(:,1));
34. end
35.
36. hj = 0;
37.
38. for o1 = 1:length(fwmat.objects)
39.     asd1 = fwmat.objects(o1).data;
40.     if fix(o1/4) == o1/4
41.         asd1 = NaN;
42.     end
43.     for  o2 = 1:length(fwmat.material)
44.         asd2 = fwmat.material(o2).data;
45.             if length(asd1) ==length(asd2)
```

```
                if all(asd1==asd2)
                    hj = hj+1;
                    yu(2,hj) = o2;
                    yu(1,hj) = o1;
                end
            end
    end
end


yu(3,:) = nfacesP;

for ki = 1:nobj
    if ki == 1
        Cs(1:yu(3,ki)) = fwmat.material(yu(2,ki)+9).data(1);
        Cd(1:yu(3,ki)) = fwmat.material(yu(2,ki)+8).data(1);
    else
        Cs((1:yu(3,ki))+sum(yu(3,1:ki-1))) =
    fwmat.material(yu(2,ki)+7).data(1);
        Cd((1:yu(3,ki))+sum(yu(3,1:ki-1))) =
    fwmat.material(yu(2,ki)+8).data(1);
    end
end

% x = vertices(:,1);
% y = vertices(:,2);
% z = vertices(:,3);

% Specifico per AGM: rotazione del modello di -90 gradi
%per essere compatibile con il modello di MONTE

vert1 = vertices(faces(:,1),:);
vert2 = vertices(faces(:,2),:);
vert3 = vertices(faces(:,3),:);

%generazione di un array di pixel

max_L = 0;
for t1 = 1:length(vertices(:,1))
    dist = norm(vertices(t1,:));
    if dist > max_L
        max_L = dist;
    end
end
L = 2*norm(max_L);
R = L;

n_pix_coord = floor(sqrt(pixels));
```

```
92.
93. Apixel = L^2/pixels; %area pixel in m^2
94. l_pix = sqrt(Apixel); %lato pixel in m
95.
96. % Calcolo delle coordinate dei centri di ogni pixel
97. x_pix_array = linspace(-(L-l_pix)/2, (L-l_pix)/2,n_pix_coord);
98. y_pix_array = x_pix_array;
99. [x_pix_array,y_pix_array] = meshgrid(x_pix_array,y_pix_array);
100.
101.       % Inizializzazione delle variabili dei cicli for annidati in
    azimut ed
102.       %elevazione
103.
104.       % Possibilità di generare una matrice
105.       %Fsrp_mat = zeros(n_pix_coord);
106.
107.       %Inizializzazione di vettori degli angoli di azimut ed
    elevazione
108.       lambda = -180 : step_az : 180;
109.       delta = -90 : step_el : 90;
110.
111.       % Inizializzazione della variabile Fsrp
112.       Fsrp = zeros(length(lambda),length(delta));
113.       Fsrp = num2cell(Fsrp);
114.
115.       % Inizializzazione della variabile ang
116.       ang = zeros(length(lambda),length(delta));
117.       ang = num2cell(ang);
118.
119.       % Calcolo della lunghezza di lambda e delta
120.       l_az = length(lambda);
121.       l_el = length(delta);
122.
123.
124.       %Calcolo iterativo di Fsrp per ogni angolo di azimut ed
    elevazione
125.       parfor tui = 1:l_az
126.           for iop = 1:l_el
127.           %Calcolo di azimut ed elevazione
128.
129.           az = deg2rad(lambda(tui));
130.           el = deg2rad(delta(iop));
131.           % Calcolo del vettore che esprime la direzione del Sole
132.           S = [cos(az)*cos(el) sin(az)*cos(el) sin(el)];
133.           % Direzione dei raggi
134.           dir = -S*2*R;
135.           % Sistema di riferimento ruotato
136.           xr = S;
```

```
137.          yr = [-sin(az) cos(az) 0];
138.          zr = cross(xr,yr);
139.          Or = xr*R;
140.
141.              %Iterazione su ogni raggio di ogni pixel
142.              for g1 = 1:n_pix_coord
143.                  for g2 = 1:n_pix_coord
144.                  % Calcolo dell'origine di un pixel
145.                  orig = x_pix_array(g1,g2)*zr+
    y_pix_array(g1,g2)*yr+ Or;
146.                  % Utilizzo della funzione per calcolare
    l'intersezione fra un
147.                  % raggio e i triangoli di cui è composto il
    modello 3D
148.                  intersect = TriangleRayIntersection(orig, dir,
    vert1, vert2, vert3,'planeType','one sided','eps',1e-14);
149.                      % Esclusione di tutte i triangoli a distanza
    maggiore del
150.                      % triangolo a ditanza maggiore
151.                      if any(intersect)
152.
153.                      dist = zeros(1,length(intersect));
154.                          for i = 1:length(dist)
155.                              if intersect(i)
156.                              mtCG = vertices(faces(i,:),:);
157.                              CG = [sum(mtCG(:,1))/3
    sum(mtCG(:,2))/3 sum(mtCG(:,3))/3];
158.                              dist(i) = dot(orig-
    CG,abs(double_normals(i,:)));
159.                              else
160.
161.                                  dist(i) = NaN;
162.                              end
163.                          end
164.                          [min_dist, ~] = min(abs(dist));
165.                          intersect(abs(dist)>min_dist) = false;
166.                              % Calcolo di Fsrp nel caso di un
    triangolo
167.                              % intercettato
168.                              if sum(intersect) == 1
169.                              teta =
    acos(dot(normals(normals_ind(intersect),:),S));
170.
171.                              Fsrp(tui,iop) = {-Apixel*((1-
    Cs(intersect))*S +
    2*(Cs(intersect)*cos(teta)+1/3*Cd(intersect))*normals(normals_ind(in
    tersect),:)) + cell2mat(Fsrp(tui,iop))};
```

```
172.                                %Calcolo dell'elemento della matrcie
    che ripora il
173.                                %modulo della forza per ogni pixel
    nel caso di un
174.                                %triangolo intercettato
175.      %                          Fsrp_mat(g2,g1) = -Apixel*norm(((1-
    Cs(intersect))*S +
    2*(Cs(intersect)*cos(teta)+1/3*Cd(intersect))*normals(normals_ind(in
    tersect),:))) ;
176.                                else
177.                                % Calcolo di Fsrp nel caso di più
    triangoli intercettati
178.                                normals2 =
    sum(normals(normals_ind(intersect),:))/length(normals(:,1));
179.                                teta = acos(dot(normals2,S));
180.                                Fsrp(tui,iop) = {-Apixel*((1-
    min(Cs(intersect)))*S +
    2*(mean(Cs(intersect))*cos(teta)+1/3*mean(Cd(intersect)))*normals2)
    + cell2mat(Fsrp(tui,iop))};
181.                                %Calcolo dell'elemento della matrcie
    che ripora il
182.                                %modulo della forza per ogni pixel
    nel caso di più
183.                                %triangoli intercettati
184.
185.      %                          Fsrp_mat(g2,g1) = -Apixel*norm(((1-
    min(Cs(intersect)))*S +
    2*(min(Cs(intersect))*cos(teta)+1/3*min(Cd(intersect)))*normals2)) ;
186.
187.                                end
188.                        end
189.                    end
190.                end
191.                % Variabile che riporta gli angoli di azimut ed
    elevazione
192.                ang(tui,iop) = {[az el]};
193.            end
194.        end
195.
196.
197.        % Fsrp_mat = Fsrp_mat(end:-1:1,:);
198.
199.        % figure(2);
200.        % colormap('default')
201.        % imagesc(Fsrp_mat)
202.        % colorbar
203.        % axis('image')
204.
```

```
205.      % Creazione dei vettori per Fsrp e ang che vanno scritti nel
   file
206.      % ridisponendo i risultati del ciclo
207.
208.      t = 0;
209.      for t1 = 1:length(lambda)
210.          for t2 = 1:length(delta)
211.              t = t+1;
212.              FsrpS(t,:) = cell2mat(Fsrp(t1,t2))/1e4;
213.              angS(t,:) = rad2deg(cell2mat(ang(t1,t2)));
214.          end
215.      end
216.
217.      % Generazione di un indice di iterazione
218.      t = 1: length(lambda)*length(delta);
219.
220.      % Generazione della stringa di intestazione del file sulla
   base di header
221.      file_header = "";
222.      file_header = file_header + "Version            : " +
   header.version + " \n";
223.      file_header = file_header + "System             : " +
   header.system + " \n";
224.      file_header = file_header + "Analysis Type      : " +
   header.analysis + " \n";
225.      file_header = file_header + "Pixel Size         : " +
   header.pixelsize + " \n";
226.      file_header = file_header + "Spacecraft Size    : " +
   header.scsize + " \n";
227.      file_header = file_header + "Pressure           : " +
   header.pressure + " \n";
228.      file_header = file_header + "Center of Mass     : " +
   header.com + " \n";
229.      file_header = file_header + "Current time       : " +
   header.date + " \n\n";
230.      file_header = file_header + "Motion             : 1 \n";
231.      file_header = file_header + " Name               : Azimuth
   \n";
232.      file_header = file_header + " Method             : Step \n";
233.      file_header = file_header + " Minimum            : -180 \n";
234.      file_header = file_header + " Maximum            : +180 \n";
235.      file_header = file_header + " Step               : " +
   step_az + " \n";
236.      file_header = file_header + "Motion             : 2 \n";
237.      file_header = file_header + " Name               : Elevation
   \n";
238.      file_header = file_header + " Method             : Step \n";
239.      file_header = file_header + " Minimum            : -90 \n";
```

```
240.      file_header = file_header + " Maximum            : +90 \n";
241.      file_header = file_header + " Step               : " +
   step_el + " \n";
242.      file_header = file_header + ":END \n\n";
243.      file_header = file_header + "Record count       : " +
   length(Fsrp) + " \n\n";
244.      file_header = file_header + "  Azimuth        Elevation
   Force(X)              Force(Y)              Force(Z)          \n";
245.      file_header = file_header + "  degrees         degrees
   m^2                   m^2                   m^2             \n";
246.      file_header = file_header + "-----------     ----------     -
   -----------------    ------------------    ------------------
   \n";
247.
248.      % Apertura del file
249.      fileID = fopen(filename, 'w');
250.      file_header_string = sprintf("%s", file_header);
251.      fprintf(fileID, file_header_string, '-append');
252.
253.
254.
255.      % Iterazione sull'indice per scrivere ogni riga del corpo del
   file
256.      for k = t
257.          str(k) = sprintf("%f",angS(k,1)) + '     ' +
   sprintf("%10f",angS(k,2))  + '     '+ sprintf("%.15f",FsrpS(k,1)) +
   '     ' +sprintf("%.15f",FsrpS(k,2)) + '     '
   +sprintf("%.15f",FsrpS(k,3)) + '\n' ;
258.          fprintf(fileID, str(k), '-append');
259.      end
260.
261.      % Chiusura del file
262.      fclose(fileID);
263.
```

Capitolo 7 : Bibliografia

[1] Jet Propulsion Laboratory. (s.d.). Tratto da ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/

[2] Jet Propulsion Laboratory. (2020, 1 7). *The SPICE Concept*. Tratto da https://naif.jpl.nasa.gov/naif/spiceconcept.html

[3] Jet Propulsion Laboratory. (s.d.). *Monte Mission Analysis, Operations, and Navigation Toolkit Environment.* Tratto da https://montepy.jpl.nasa.gov

[4] Kroon, D.-J. (2011, 9 9). *MATLAB Central- Mathworks Files Exchange*. Tratto da Wavefront OBJ toolbox: https://www.mathworks.com/matlabcentral/fileexchange/27982-wavefront-obj-toolbox

[5] NASA. (s.d.). *mars.nasa.gov*. Tratto da Mars Reconnaissance Orbiter: https://mars.nasa.gov/mro/

[6] T. Möller, B. T. (1997). *Fast, Minimun Storage Ray/Triangle Intersection.*

[7] The GMAT Development Team. (2012, 5 18). *General Mission Analysis Tool (GMAT) User Guide* . Tratto da http://gmat.sourceforge.net/docs/R2012a/help-letter.pdf

[8] Trevor W. Williams, K. M. (2015). *Orbit Stability of OSIRIS-REx in the Vicinity of Bennu Using a High-Fidelity Solar Radiation Model.*

[9] Tuszynski, J. (2018, 5 18). *MATLAB Central-MathWorks Files Exchange*. Tratto da Triangle/Ray Intersection: https://www.mathworks.com/matlabcentral/fileexchange/33073-triangle-ray-intersection