universidad de león

Universidad de Ingenierías
Industrial, Informática y Aeroespacial

Grado en Ingeniería Aeroespacial

Trabajo de Fin de Grado

Simulador para análisis y estudio de misiones espaciales
Simulator for analysis and study of space missions

Autor: Álvarez Redondo, Mario
Tutor: García Gutiérrez, Adrián
Cotutor: Domínguez Fernández, Diego

Julio 2023

UNIVERSIDAD DE LEÓN

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA AEROESPACIAL
Trabajo de Fin de Grado

ALUMNO: Mario Álvarez Redondo

TUTOR: Adrián García Gutiérrez / Diego Domínguez Fernández

TÍTULO: Simulador para análisis y estudio de misiones espaciales

TITLE: Simulator for analysis and study of space missions

CONVOCATORIA: Julio, 2023

RESUMEN:

El presente trabajo consiste en el desarrollo de simulador numérico en el ámbito de la mecánica espacial. Se presenta como una librería de Python que tiene una interfaz para hacer uso de algunas de sus funciones. El objetivo es el análisis y estudio de los diferentes parámetros involucrados en las misiones espaciales. Se ha desarrollado un *propagador de órbitas* para diferentes situaciones tanto para el ámbito terrestre como dentro del sistema solar. A partir de la definición de la misión se implementa un modelo de *potencial gravitatorio* (EGM96) dentro del ámbito terrestre y del *campo magnético* (IGRF13). Se han desarrollado algoritmos de *determinación preliminar de órbitas*, así como *cambios de coordenadas*, *transferencias interplanetarias* (Problema de Lambert), etc. El simulador se ha realizado con el lenguaje de programación Python, y los resultados extraídos pueden ser exportados en un fichero *.csv* para su posterior uso en otras aplicaciones.

ABSTRACT:

This work consists of the development of a numerical simulator in the field of spatial mechanics. It is presented as a Python library with an interface to make use of some of its functions. The objective is the analysis and study of the different parameters involved in space missions. An *orbit propagator* has been developed for different situations both for the terrestrial environment and within the solar system. Based on the mission definition, a model of the *gravitational potential* (EGM96) within the terrestrial domain and of the *magnetic field* (IGRF13) are implemented.

Table 1 continued from previous page

<table>
<tr><td colspan="2">Algorithms have been developed for preliminary orbit determination, as well as coordinate changes, interplanetary transfers (Lambert problem), etc. The simulator has been developed with the Python programming language, and the extracted results can be exported in a .csv file for further use in other applications.</td></tr>
<tr><td colspan="2">Palabras clave: Espacial, simulador numérico, Python, Campo magnético, IGRF13,EGM96, propagador de órbitas, problema de Lambert</td></tr>
<tr><td>Firma del alumno:</td><td>VºBº Tutor/es:</td></tr>
</table>

Índice general

Índice de figuras

Índice de extractos de código

Capítulo 1

Introducción

La simulación numérica es uno de los sectores claves dentro de la ingeniería, científico e ingenieros hacen uso de este tipo de herramientas para desempeñar su profesión. El presente proyecto se encuadra dentro de este área, donde se centra en la rama de la mecánica espacial. Se presenta una librería en Python que consta de diferentes utilidades como la obtención del potencial terrestre o los valores del campo magnético dentro del ámbito terrestre para un determinado punto, además se han desarrollado una serie de propagadores de órbitas, cambio de ejes coordenados, etc. También es importante que los datos puedan ser extraídos de la aplicación con facilidad y en un formato adecuado, en nuestro caso *.csv* para que puedan ser usados posteriormente para otros proyectos e investigaciones.

1.1. OBJETIVOS

Este proyecto tiene una serie de objetivos que se pretenden alcanzar a lo largo de su desarrollo, En primer lugar, el objetivo principal es desarrollar una herramienta que permita realizar una serie de simulaciones dentro del ámbito de la mecánica espacial.

Esto se pretende conseguir con el desarrollo en Python de una serie de librerías que contienen las funciones más destacadas para la simulación de una misión espacial. Además, se pretende realizar una aplicación de consola que permita utilizar las diferentes funciones de la librería para una obtención de resultados más rápida y eficaz.

Entre las herramientas que se pretenden desarrollar, se necesita una "suite"de funciones que permitan obtener los vectores posición $\vec{r}_0$ y velocidad iniciales $\vec{v}_0$, esto se corresponde con la determinación preliminar de órbitas. Una vez se obtienen los vectores de inicio, es necesario propagar la órbita a lo largo del tiempo, por lo que será necesario desarrollar unos propagadores que sean capaces de realizar dicha función.

Otro de los objetivos que se tienen en el proyecto es la determinación del campo magnético y el potencial terrestre en cualquier punto, es por ello, que puesto que al propagar la órbita se trabaja en coordenadas inerciales ECI, para la obtención de dichos campos es necesario cambiar a coordenadas ECEF, se hace necesaria una nueva "suite"de funcionalidades que permitan cambiar de coordenadas de manera cómoda.

En el caso de que el tipo de misiones sean fuera del ámbito de la Tierra, no tendremos las funcionalidades de la obtención del campo magnético y potencial terrestre, pero el resto de funcionalidades se mantendrán funcionales.

1.2. ¿POR QUÉ PYTHON?

Python es uno de los lenguajes de programación más utilizados a día de hoy y en constante crecimiento, es necesario analizar el porqué de esta situación. En primer lugar, se trata de un lenguaje de programación de alto nivel, esto permite una mejor comprensión del código. Además, se considera un lenguaje multipropósito, se pueden hacer desde interfaces gráficas para escritorio como utilizarse para Data Science donde está actualmente muy extendido. Este lenguaje a su vez permite una programación estructurada, funcional y orientada a Objetos, lo cual es uno de sus principales puntos fuertes. Se pueden añadir la gran cantidad de librerías desarrolladas con soporte continuo que permiten abordar casi cualquier temática, en nuestro caso son de especial interés las librarías de desarrollo de GUI como tkinter y las de análisis numérico como Numpy o Scipy entre otras. Se trata de un lenguaje de código abierto, por lo que se puede tener un control total de lo que programamos y como se mencionó al principio, es uno de los lenguajes más usados, lo que significa que hay una comunidad muy amplia.

Por otro lado, hay que considerar algunas de sus desventajas frente a lenguajes alternativos, los lenguajes compilados. Una desventaja es el alto consumo de memoria que se produce a causa de la flexibilidad con los tipos de datos. Hay que añadir, que al tratarse de un lenguaje interpretado, este se ejecuta línea a línea al contrario que los lenguajes compilados, esto permite tener esa flexibilidad en los tipos de datos, pero penaliza el rendimiento general. Para paliar este problema del lenguaje, se pueden utilizar unas librerías que se encargan de precompilar el código (Lo que limita el tipo de programación se puede realizar) por lo que se pueden suplir las pérdidas de rendimiento (numba). Estos aceleradores de código son especialmente útiles en los bucles.

1.3. LIMITACIONES

Python es un lenguaje de programación muy popular y versátil, que se puede usar para diversas aplicaciones numéricas, como data science, machine learning, análisis estadístico, etc. Sin embargo, también tiene algunas limitaciones que pueden afectar su rendimiento o funcionalidad en ciertos casos. Algunas de estas limitaciones son:

- **Velocidad de ejecución:** Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea y no se compila previamente. Esto hace que Python sea más fácil de leer y escribir, pero también más lento que otros lenguajes compilados como C o Fortran. Además, Python tiene un mecanismo de gestión de memoria llamado recolector de basura, que libera el espacio ocupado por los objetos que ya no se usan. Esto puede causar pausas o retrasos en la ejecución del programa.
- **Tipado dinámico:** Python es un lenguaje de tipado dinámico, lo que significa

que el tipo de las variables se determina en tiempo de ejecución y no se declara explícitamente. Esto hace que Python sea más flexible y expresivo, pero también más propenso a errores o inconsistencias en el manejo de los datos.

- **Memoria:** Python usa una estructura de datos llamada arreglo de NumPy para almacenar y operar con datos numéricos. Los arreglos de NumPy son eficientes y ofrecen muchas funciones y operaciones matemáticas, pero también consumen mucha memoria. Cada elemento de un arreglo de NumPy tiene el mismo tipo y tamaño, lo que implica que se desperdicia espacio si los datos son heterogéneos o tienen diferentes precisiones.

Estas son algunas de las limitaciones de Python para su uso en aplicaciones numéricas de gran complejidad. Sin embargo, existen formas de mitigar o superar estas limitaciones, como usar bibliotecas externas, integrar código en otros lenguajes, optimizar el código o usar herramientas alternativas [1].

Capítulo 2

Estado del arte

En este apartado se van a estudiar las diferentes alternativas tanto de código abierto como comerciales que proponen herramientas de simulación numérica en diferentes campos de la mecánica espacial.

El desarrollo de la informática y sus diferentes ramas han permitido que a día de hoy ya haya algunas herramientas de simulación numérica de misiones espaciales. En este apartado se analizan algunas de las más destacadas y algunas de las cuales serán usadas para la validación de algunos aspectos de la herramienta que se ha desarrollado.

2.1. GMAT

Se trata de un software que tiene por objetivo el análisis de misiones espaciales, de ahí su nombre "General Mission Analysis Tool"del cual proviene su acrónimo GMAT. Se trata de una herramienta software libre que permite simular diversas misiones en distintas condiciones de vuelo, desde órbitas bajas, órbitas lunares e incluso misiones interplanetarias. Su desarrollo ha sido llevado a cabo por la NASA, diversas empresas privadas del sector aeroespacial y otros colaboradores tanto de carácter público como privado.

Sus principales usos son el desarrollo de misiones espaciales reales, estudios de ingeniería, como herramienta educativa, etc. Dado el gran rango de usuarios a los que va orientado este programa, contiene datos reales de ciertos objetos como naves espaciales, satélites y lanzadores, así como herramientas de propagación de órbitas, representaciones gráficas e informes de los datos obtenidos. Todas estas herramientas disponibles se usan en las distintas fases de una misión espacial, en las cuales los usuarios emplean una serie de comandos soportados por el sistema con el fin de modelar la misión y establecer su viabilidad. [2]

2.2. AGI-STK

Se trata de una solución de la compañía Ansys para el estudio de misiones y análisis de sistemas dentro de un entorno de modelado físico. Se usa para el análisis de plataformas y cargas de pago en un escenario de una misión realista. Entre sus múltiples funcionalidades tenemos el diseño de sistemas para misiones espaciales.

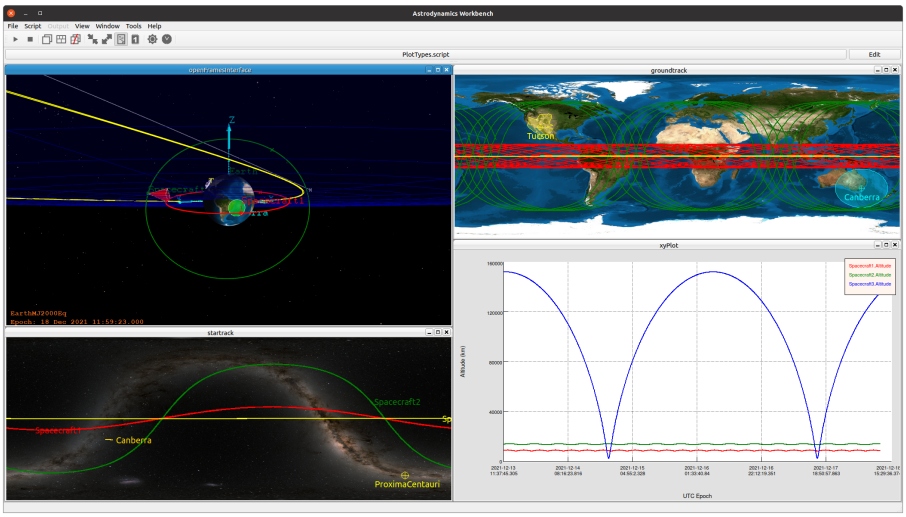


Figura 2.1: Captura de pantalla del programa GMAT

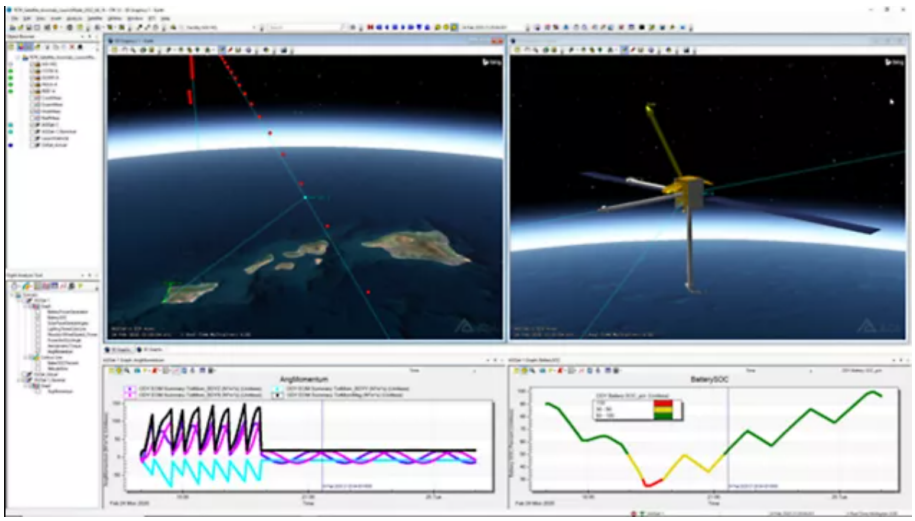


Figura 2.2: Ansys STK [3]

2.3. ORBITER

Se trata de un simulador desarrollado para la plataforma Microsoft Windows por Martin Schweiger. Se trata de un simulador realista de altas capacidad con una sencilla interfaz, que permite una visualización de manera gráfica. *Orbiter* puede usarse en instituciones de investigación para la visualización de misiones espaciales. Las aplicaciones van de crear vídeos de misiones a simulación de vuelo en plataformas fijas [4]. Una de las limitaciones del programa es que, si bien, tiene implementados diferentes modelos que le hacen ser un simulador realista, su interfaz gráfica está más orientada al desarrollo visual de misiones espaciales.

2.4. CELESTRACK

Se trata de un repositorio de datos y de código que permite tener información de los objetos celestes que orbitan la Tierra, ya sean satélites activos, inactivos, estaciones espaciales, etc. Estos datos que actualizan periódicamente y en nuestro caso se ha extraído información de la órbita del satélite militar español PAZ para la realización de algunas validaciones. El objetivo de esta web, es proporcionar herramientas de acceso libre a toda la comunidad de estudiantes, científicos, ingenieros y personas que necesiten este tipo de recursos dentro del ámbito de la mecanica espacial. Tal y como se cuenta en su propia web [5]:

"CelesTrak's mission remains focused on making data and other resources freely available to the space community to facilitate understanding of our orbital environment and how to use it safely and responsibly"

2.5. SIMU-CIC Y CELESTLAB

SIMU-CIC es una herramienta desarrollada por la CNES que permite mediante el uso de´l entorno Scilab - Celestlab la generación de ficheros de efemérides teniendo en cuenta los siguientes parámetros:

1. Parámetros orbitales.
2. Localización de estaciones terrestres.
3. Geometría de vehículo espacial.
4. Actitud.

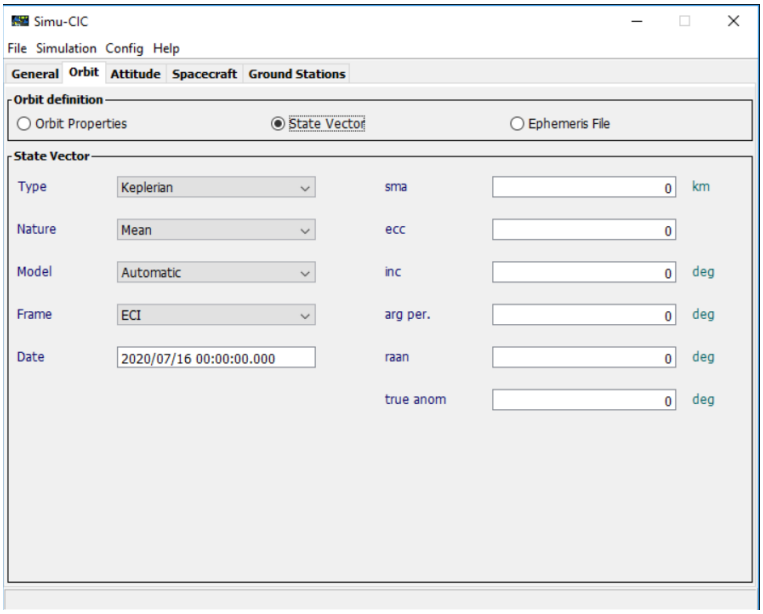


Figura 2.3: Interfaz de SIMU-CIC [6]

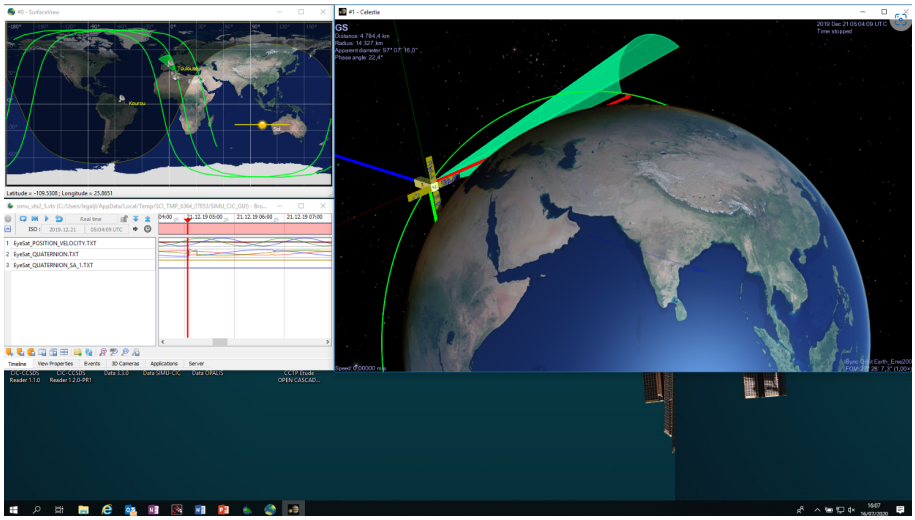


Figura 2.4: Visualización de resultados SIMU-CIC [6]

Una vez diseñada la misión, los datos generados por SIMU-CIC pueden ser visualizados en Celestia u otros software como VTS también desarrollados por la CNES. VTS es un software desarrollado también por la CNES para sincronizar y visualizar de manera ordenada los datos de una determinada misión espacial.

2.6. OTROS RECURSOS DE INTERÉS

Puesto que el software que se va a utilizar para la realización del proyecto es el lenguaje de programación Python, es necesario que veamos las librerías que pueden resultar de interés.

2.6.1. POLIASTRO

Se trata de una librería de Python open source que permite el estudio de astrodinámica y mecánica orbital. Ha sido desarrollada por Juan Cano y su equipo [7]. Algunas de sus características que lo hacen ser un motor potente son sus métodos de propagación de órbitas tanto analíticos como numéricos, conversión entre parámetros orbitales y vectores de posición y velocidad, cambios de sistemas de referencia, maniobras de transferencias orbitales...

Figura 2.5: Librería Poliastro [7]

2.6.2. ORBIT-PREDICTOR

Orbit Predictor es una librería Python para propagar órbitas de objetos en órbita terrestre (satélites, ISS, Santa Claus, etc) usando TLE (Two-Line Elements set) [8]

Capítulo 3

Planificación

3.1. PLANIFICACIÓN DEL PROYECTO

La estructuración de este proyecto se basa en objetivos, cada uno de esos objetivos se traduce en el desarrollo de una nueva funcionalidad. Como se verá en apartados sucesivos, el código se encuentra desarrollado por clases, cada una de esas clases tiene un objetivo, por ejemplo, la clase **geomag** se encarga de realizar todo lo relativo al cálculo del campo magnético en diferentes situaciones. Por otro lado, hay clases transversales que se utilizan en otras clases para su funcionamiento, como puede ser el caso de **utilities**.

Para la organización del proyecto, en primer lugar se va a realizar un primer borrador de la estructura del código que ayudará a organizar las ideas en la cabeza a un desarrollo más ágil en etapas sucesivas del desarrollo del mismo.

Esta estructura inicial va a ir cambiando hasta el desarrollo final, se puede ver como ha quedado la estructuración del código en la siguiente figura 3.1.

A continuación, se muestra otra de las etapas de la planificación del proyecto. Se ha estructurado de acuerdo a un diagrama Gantt que ha permitido el desarrollo del mismo.

El número de horas aproximado que se ha empleado para el proyecto se encuentra en el diagrama Gantt y ronda un total de 267 horas, incluyendo todas las partes del desarrollo del mismo. Para cada una de las tareas desarrolladas se ha tenido que hacer una fase previa de documentación, la fase de desarrollo donde se ha realizado la implementación numérica y la fase de validación. Las fases de desarrollo numérico y validación en cada uno de los apartados han sido las que más tiempo han tomado para su completo desarrollo.

El número de horas que se ha necesitado para la primera etapa de documentación y configuración del entorno ha sido de 30 horas, para la parte de desarrollo de software, englobando el desarrollo de los scripts de cambio de coordenadas, maniobras interplanetarias, determinación preliminar de órbitas y el desarrollo de propagadores ha conllevado un número de 162 horas de trabajo. El desarrollo del modelo del campo potencial terrestre ha llevado 40 horas, mientras que el magnético ha llevado un total de 30 horas. 25 horas han sido empleadas para el desarrollo de la aplicación y corrección de errores. Para la validación del conjunto de funciones que se han desarrollado, se han empleado 40 horas que han sido realizadas en paralelo al desarrollo de las diferentes funciones.

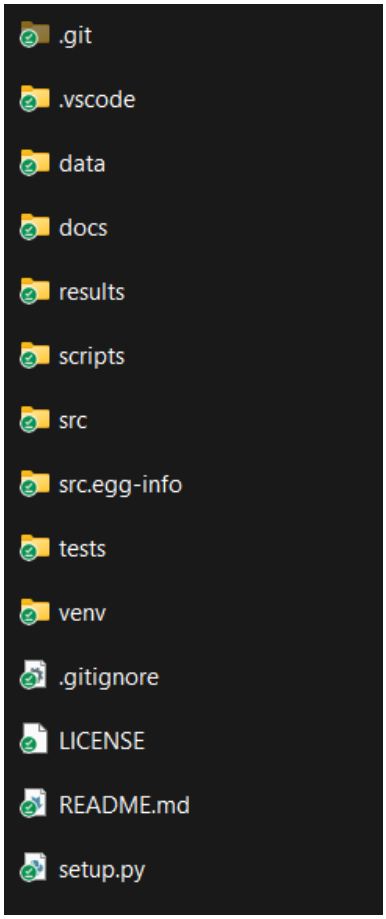


Figura 3.1: Estructura del proyecto

El diagrama Gantt con toda la información de la planificación se encuentra en la página siguiente.

Gestión de proyectos - TFG

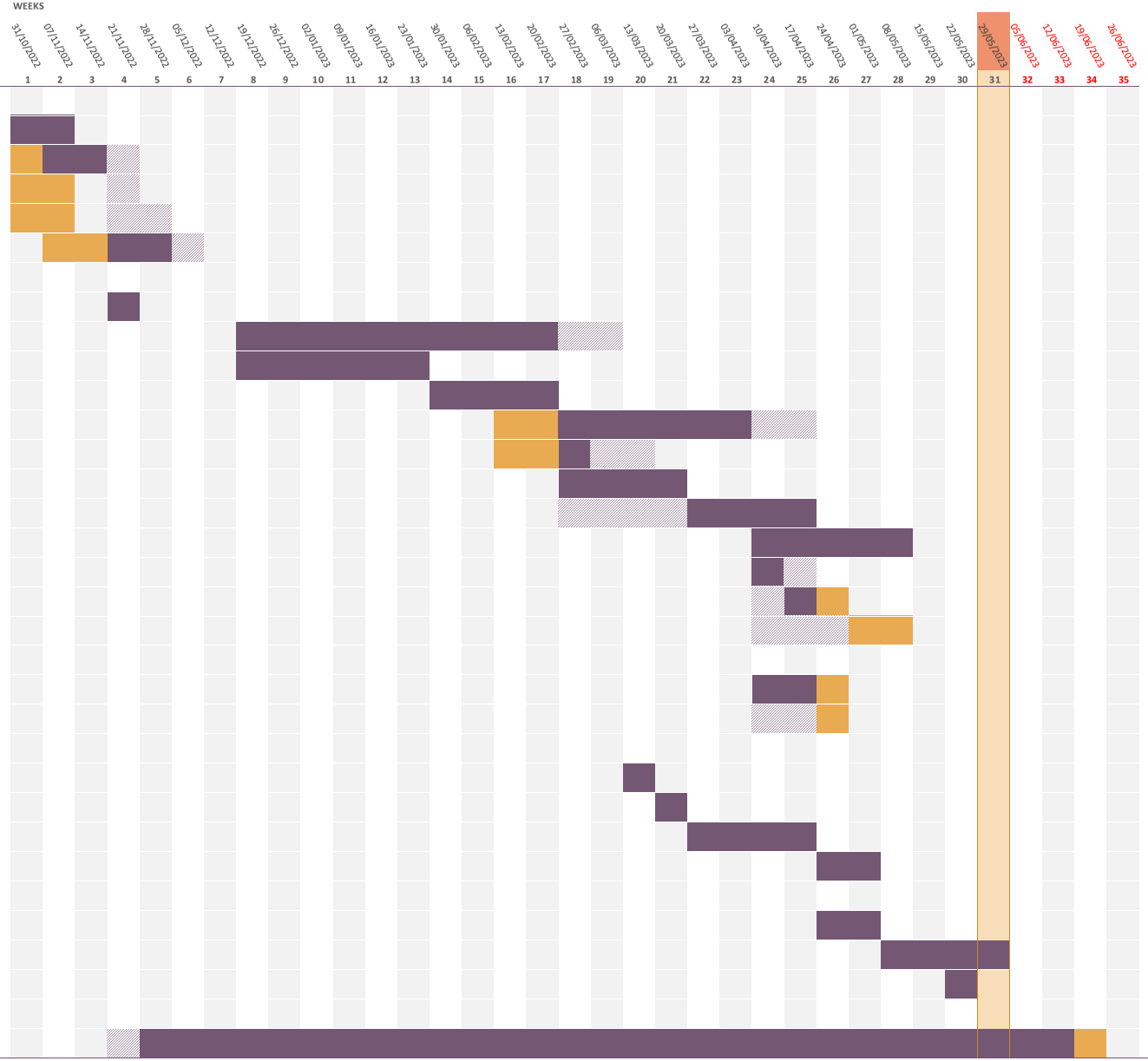
Universidad de León

Horas 267

Period Highlight: 31

Duración planeada | Comienzo real | % Completado | Actual (más allá de lo planeado) | % Completado (más allá de lo planeado)

ACTIVIDAD	COMIENZO	DURACION	COMIENZO REAL	Horas	DURACIÓN REAL	PORCENTAJE COMPLETADO
Busqueda de información						100%
Gestión del proyecto	1	2	1	10	2	100%
Estado del arte	2	3	1	10	3	100%
Busqueda de información VSC, git	4	1	1	2	2	100%
Configuración Python	4	2	1	2	2	100%
Configuración Latex	4	3	2	6	4	100%
Desarrollo software						100%
Desarrollo src.Celestialbodies	4	1	4	2	1	100%
Desarrollo src.Coordinates	8	12	8	20	10	100%
ECI to ECEF	8	6	8	14	6	100%
Funciones adicionales	14	4	14	6	4	100%
Desarrollo propagadores	18	8	16	30	8	100%
Kepler	18	3	16	6	3	100%
KeplerP	18	4	18	10	4	100%
Encke	18	8	22	14	4	100%
Desarrollo de herramientas para el cálculo de maniobras	24	5	24	30	5	100%
Lambert	24	2	24	10	1	100%
Maniobras interplanetarias	24	2	25	15	2	100%
Transferencias interplanetarias	24	3	27	5	2	100%
Campo gravitatorio terrestre						100%
Desarrollo de src.geopot	24	2	24	30	3	100%
Implementación de funciones, para plotear	24	2	26	10	1	100%
Campo magnético terrestre						100%
Web scraping para actualizar las variaciones del campo	20	1	20	5	1	100%
Implemetación en Python	21	1	21	20	1	100%
Representación en diferentes ejes coordenados	22	4	22	20	4	100%
Validación NNOA	26	2	26	5	2	100%
Aplicación						100%
Estructura general	26	2	26	5	2	100%
Implementación de las herramientas desarrolladas	28	4	28	15	4	100%
Révision de la performance	30	1	30	5	1	100%
Test y validación						100%
Validación general	4	30	5	40	30	100%



3.2. ESTRUCTURACIÓN DEL CÓDIGO

- **data**: Esta carpeta contiene archivos fundamentales para el desarrollo del programa, entre otros los coeficientes que permiten calcular el potencial terreste o el campo magnético.
- **docs**: En esta carpeta, se puede encontrar la documentación que se ha utilizado para el desarrollo del programa.
- **results**: Contiene archivos que son resultados de la ejecución del código de los diferentes apartados.
- **scripts**: Desarrollo de la aplicación en consola de comandos. Integra parte de las funcionalidades del simulador.
- **src**: Es la carpeta más importante de todas, pues contiene todas las clases que dan plena funcionalidad el simulador.
  - **CelestialBodies**: Clase que permite crear objetos de cuerpos celestes como planetas o satélites como la Luna para su posterior uso.
  - **Coordinates**: Repositorio de funciones que permiten realizar cambios de coordenadas.
  - **geomag**: Permite calcular el campo magnético terrestre, así como algunas representaciones gráficas.
  - **geopot**: Permite calcular el campo gravitatorio terrestre, así como algunas representaciones gráficas.
  - **interplanetary**: Contiene un conjunto de funciones que permiten realizar problemas interplanetarios como la resolución del problema de Lambert.
  - **KeplerPropagator**: Perturbaciones J2.
  - **Lambert**: Resolución del problema de Lambert desde diferentes modelos.
  - **ord2rv**: Cambio de parámetros entre parámetros orbitales y vectores posición y velocidad.
  - **Orbit**: Propagar de Kepler básico.
  - **Propagator**: Perturbaciones de un tercer cuerpo.
  - **rv2orb**: Cambio de parámetros entre vectores posición y velocidad y parámetros orbitales.
  - **time_conv**: Funciones para convertir tiempos.
  - **Transferts**: Funciones básicas de transferencias clásicas.
  - **utilites**: Clase con funciones que sirven de soporte al resto del código.
- **tests**: Se pueden encontrar diversas pruebas que se han realizado para la validación del código,
- **venv**: Entorno virtual que asegura no tener problemas con el cambio de versiones de los diferentes paquetes utilizados.

Capítulo 4

Fundamentos teóricos del simulador

En este capítulo se pueden observar los principios físicos básicos que servirán para comprender los capítulos sucesivos. El proyecto se basa en *Orbital Mechanics for engeneering students* [9], *An Introduction to the Mathematics and Methods of Astrodynamics* [10] y *Orbital Mechanics* [11].

4.1. PARÁMETROS ORBITALES

Una órbita viene definida generalmente por los siguientes parámetros que se pueden apreciar en la figura 4.1. Un cuerpo en el espacio queda definido por 6 variables que hacen referencia a sus grados de libertad. En el espacio hay 6 grados de libertad, 3 asociados a los movimientos de traslación, y 3 asociados a los movimientos de rotación.

Para la definición de una órbita en un plano, se necesitan únicamente dos variables, el momento angular h (Normal al plano de la órbita) y la excentricidad e, el resto de parámetros pueden ser deducidos de estos dos. Partimos de un sistema coordenado inercial. Para localizar un punto en dicha órbita se requiere un tercer parámetro que es la anomalía verdadera (Desde el perigeo). Si se quiere ubicar el plano de la órbita en tres direcciones, necesitaremos 3 ángulos más, conocidos como ángulos de Euler.

La intersección del plano de la órbita con el plano ecuatorial proporciona la línea nodos. El punto donde esta línea intersecta con la órbita y pasa de la parte inferior a la superior del plano ecuatorial se conoce como el nodo ascendente. El ángulo entre la dirección de referencia y la línea de nodos (Con el nodo ascendente) proporciona el primer ángulo de Euler que se conoce como la longitud del nodo ascendente Ω o por sus siglas en inglés RAAN (Right Ascension of Ascending Node). El ángulo diedro entre el plano orbital y el plano ecuatorial es la inclinación i. Por último queda definir el perigeo de la órbita, para ello se necesita un ángulo a mayores, el ángulo entre la línea de nodos y el perigeo se conoce como argumento del perigeo ω y es el último de los 3 ángulos de Euler que quedaba por definir[9].

Normalmente, el momento angular se suele cambiar por el semieje mayor, a que es la semisuma del eje mayor y menos de la órbita elíptica.

Para resumir, a continuación se muestran los 6 parámetros orbitales básicos.

- h Momento angular específico.
- i Inclinación.
- Ω Longitud del nodo ascendente (RAAN).
- e Excentricidad.

- ω Argumento del perigeo.
- θ Anomalía verdadera.

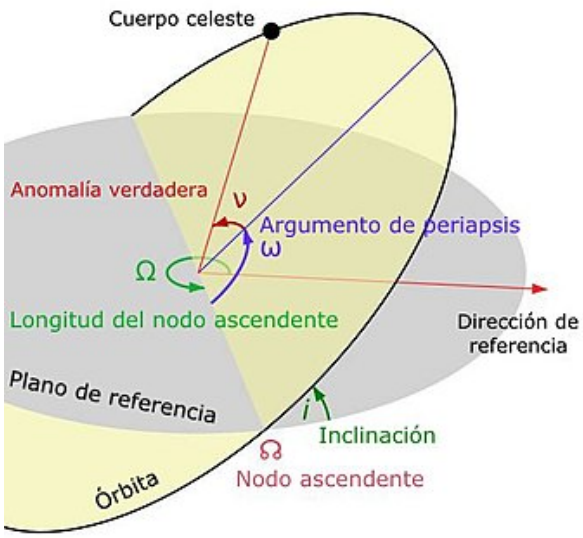


Figura 4.1: Parámetros orbitales [12]

4.2. KEPLER

Kepler enunció sus tres leyes en torno a principios del siglo XVII, estas se pueden resumir en las siguientes:

1. Los planetas tienes movimientos planos y sus órbitas son cónicas con el Sol en uno de los focos. Para ello se suponen que los cuerpos son homogéneos y solo hay una sola fuerza de atracción

$$\mathbf{F} = -m\frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} \Longrightarrow \ddot{\mathbf{r}} - \mu\frac{\mathbf{r}}{\|\mathbf{r}\|^3}, \tag{4.1}$$

donde $\mu = \frac{G}{\mathbf{r}}$ es la constante de gravitación del cuerpo central, $\mathbf{r}$ es el vector posición del cuerpo en rotación con respecto a la masa puntual central y m es la masa del cuerpo que está orbitando.
De la conservación del momento cinético C se tiene:

$$\frac{d\mathbf{C}}{dt} = \frac{d}{dt}(\mathbf{r}\times\dot{\mathbf{r}}) = \mathbf{r}\times\ddot{\mathbf{r}} + \dot{\mathbf{r}}\times\dot{\mathbf{r}} = \mathbf{0}. \tag{4.2}$$

De la evaluación de la conservación de la energía cinética y en función del valor de la constante se puede establecer el tipo de cónica

$$\frac{V^2}{2} - \frac{\mu}{r} = K. \tag{4.3}$$

2. Las áreas barridas en tiempos iguales por el radio vector que une el cuerpo central

con el cuerpo que está orbitando son iguales. Se puede hacer una breve demostración partiendo del área diferencial dA recorrida en un dt:

$$dA = \frac{1}{2}r^2 d\theta, \tag{4.4}$$

y la conservación del momento cinético.

$$\mathbf{C} = r^2 \frac{d\theta}{dt} = cte. \tag{4.5}$$

de donde deducimos:

$$\frac{dA}{dt} = \frac{1}{2}r^2\dot{\theta} = \frac{1}{2}\|\mathbf{C}\| = cte. \tag{4.6}$$

3. En el caso de una elipse, el periodo de revolución T puede expresarse en función de a y μ como:

$$T = 2\pi\sqrt{\frac{a^3}{\mu}}. \tag{4.7}$$

4.3. PROBLEMA DE LOS 2 CUERPOS

El problema de los dos cuerpos es un problema clásico de la mecánica celeste que consiste en determinar las trayectorias de dos cuerpos que interactúan gravitacionalmente entre sí. El problema se puede resolver mediante la solución de las ecuaciones diferenciales que describen el movimiento de los cuerpos. Las ecuaciones del problema de los dos cuerpos son las siguientes:

$$\frac{d^2 2\vec{r_1}}{dt^2} = -\frac{Gm_2}{r_{12}^3}\vec{r_{12}}, \tag{4.8}$$

$$\frac{d^2\vec{r_2}}{dt^2} = -\frac{Gm_1}{r_{12}^3}\vec{r_{12}}, \tag{4.9}$$

donde $\vec{r_1}$ y $\vec{r_2}$ son los vectores posición de los cuerpos 1 y 2 respectivamente, m_1 y m_2 son sus masas, G es la constante gravitatoria y $r_{12} = \vec{r_2} - \vec{r_1}$ es el vector posición relativa entre los dos cuerpos.

Si se generaliza el problema a n cuerpos, se encuentra otro problema clásico de la mecánica celeste que consiste en determinar las trayectorias de n cuerpos que interactúan gravitacionalmente entre sí. El problema se puede resolver mediante la solución de las ecuaciones diferenciales que describen el movimiento de los cuerpos. Las ecuaciones del problema de los n cuerpos son las siguientes:

$$\frac{d^2\vec{r_i}}{dt^2} = -G\sum_{j=1,j\neq i} n\frac{m_j}{r_{ij}^3}\vec{r_{ij}}, \tag{4.10}$$

donde $\vec{r_i}$ y $\vec{r_j}$ son los vectores posición de los cuerpos 1 y 2 respectivamente, m_i y m_j son sus masas, G es la constante gravitatoria y $r_{ij} = \vec{r_j} - \vec{r_i}$ es el vector posición relativa entre los dos cuerpos.

4.4. PERTURBACIONES DE LA ORBITA

Se conocen como perturbaciones a las fuerzas de atracción que se aplican sobre el centro de masas del cuerpo que orbita. Estas se pueden clasificar en tres apartados principales.

1. Debidas a la Tierra.
   - Potencial Terrestre.
   - Rozamiento atmosférico.
   - Mareas terrestres y oceánicas.
2. Debidas a otros cuerpos.
   - Atracción del Sol.
   - Atracción de la Luna.
   - Presión de radiación solar.
   - Atracción de otros planetas.
   - Efectos relativistas.
3. Debidos al satélite.
   - Maniobras orbitales.

Potencial solar y lunar

Un satélite orbitando en una órbita de tipo geoestacionario (GEO), se sentirá atraído hacia el plano eclíptico (Plano definido por la órbita de la Tierra alrededor del Sol)

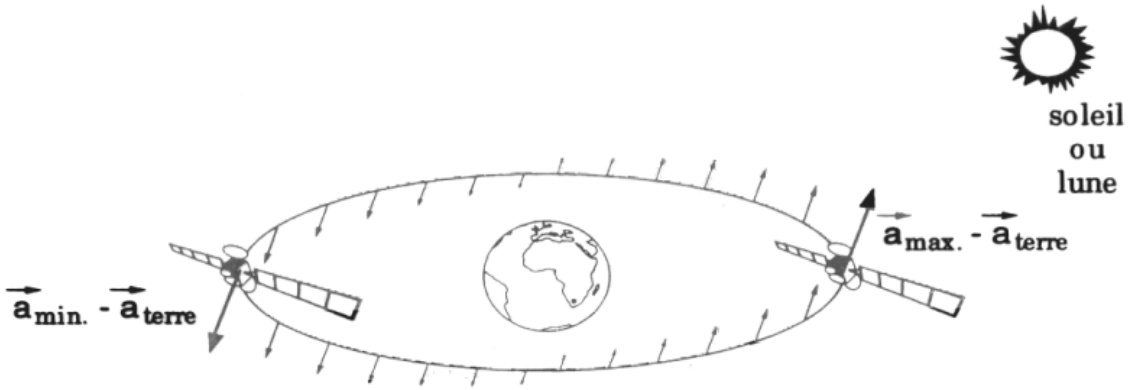


Figura 4.2: Potencial luni-solar (CNES/ESTACA)

De igual manera, un satélite en una órbita baja (LEO), y en el resto de órbitas, sufrirán una ligera modificación de su inclinación i

Rozamiento atmosférico

$$\overrightarrow{\gamma_p} = -\frac{1}{2}\rho\frac{S}{M}C_D V\overrightarrow{V}. \tag{4.11}$$

donde

- $\overrightarrow{\gamma_p}$ Aceleración perturbadora
- ρ Densidad atmosférica
- S Superficie del satélite
- M Masa del satélite
- C_D Coeficiente aerodinámico del satélite
- V Velocidad del satélite

Como consecuencias del rozamiento con la atmósfera se tiene la disminución del semieje mayor a en la ecuación 4.12 y la circularización de la órbita $e \longrightarrow 0$

$$\frac{da}{dt} = -\rho\left(\frac{SC_x}{m}\right)\sqrt{\mu a}. \tag{4.12}$$

Como resultados se obtiene que en función de la altitud la duración de vida de un satélite es muy variable

- $h < 200$ km Duración de algunos días.
- $200 < h < 500$ km Se necesita control del orbita (altitud).
- $h > 500$ km Efecto despreciable de la duración en órbita.

Presión de radiación solar

Esta perturbación solo influye en los satélites que poseen grandes paneles solares

$$\overrightarrow{\gamma_p} = \epsilon\frac{S}{M}C_P P_0\overrightarrow{U}, \tag{4.13}$$

donde

- $\overrightarrow{\gamma_p}$ Aceleración perturbadora
- S Superficie perpendicular al flujo
- M Masa del satélite
- C_P Presión de radiación solar por unidad de superficie
- U Vector unitario en la dirección Sol-Tierra
- ϵ Coeficiente de eclipse (0 o 1)

Como consecuencia principal se tiene la modificación de la excentricidad y la orien-

tación del perigeo

4.5. ÓRBITAS OSCULANTES

Las órbitas oscultoras u osculantes son las órbitas que tendría un objeto en el espacio en un momento dado en el tiempo si no hubiera perturbaciones. Es decir, es la órbita que coincide con su estado vectorial orbital actual (posición y velocidad). Una vez pasado un Δt la posición y la velocidad del objeto a raíz de las perturbaciones van a cambiar ligeramente. La posición y la velocidad se pueden describir completamente con los seis elementos orbitales estándar de Kepler (elementos de osculantación), que son fáciles de calcular siempre que se conozca la posición y la velocidad del objeto con respecto al cuerpo central 4.3

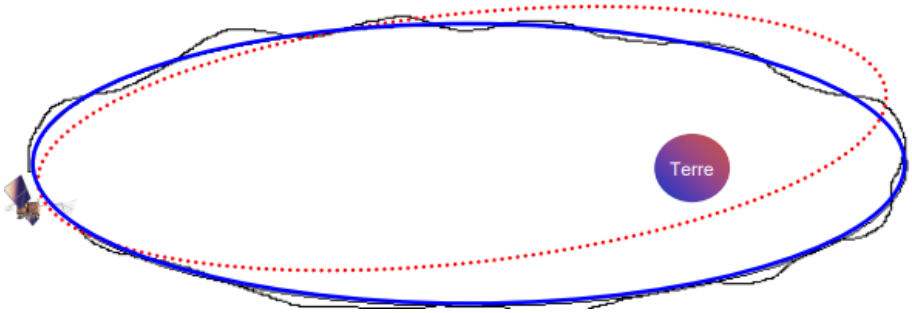
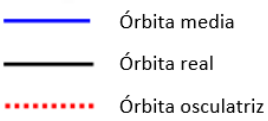


Figura 4.3: Órbita osculante (ESTACA)



Para expresar las perturbaciones hay dos aproximaciones principales, la de Lagrange y la de Gauss

Ecuaciones de Lagrange

$\overrightarrow{\gamma} = -\overrightarrow{grad}(R)$ donde R es el potencial terrestre.

Como se puede apreciar, las perturbaciones se derivan de un potencial. En esta aproximación de Lagrange se tienen en cuenta las irregularidades de del potencial terrestre, así como la atracción de la Luna y el Sol.

$$\begin{pmatrix} \frac{da}{dt} \\ \frac{de}{dt} \\ \frac{di}{dt} \\ \frac{d\Omega}{dt} \\ \frac{d\Omega}{dt} \\ \frac{dM}{dt} - n \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \frac{2}{na} \\ 0 & 0 & 0 & 0 & -\frac{\sqrt{1-e^2}}{na^2 e} & \frac{1-e^2}{na^2 e} \\ 0 & 0 & 0 & -\frac{1}{na^2\sqrt{1-e^2}\sin(i)} & \frac{\cos(i)}{na^2\sqrt{1-e^2}\sin(i)} & 0 \\ 0 & 0 & \frac{1}{na^2\sqrt{1-e^2}\sin(i)} & 0 & 0 & 0 \\ 0 & \frac{\sqrt{1-e^2}}{na^2 e} & \frac{-\cos(i)}{na^2\sqrt{1-e^2}\sin(i)} & 0 & 0 & 0 \\ -\frac{2}{na} & -\frac{1-e^2}{na^2 e} & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{dR}{da} \\ \frac{dR}{de} \\ \frac{dR}{di} \\ \frac{dR}{d\Omega} \\ \frac{dR}{d\omega} \\ \frac{dR}{dM} \end{pmatrix}, \tag{4.14}$$

donde $n = \frac{2\pi}{T}$ y $n^2 a^3 = cste$

Ecuaciones de Gauss

En esta aproximación de Gauss se muestra un caso más general que el anterior. La aceleración se encuentra expresada en un sistema de referencia local (ligado al centro de gravedad del satélite), están expresadas en $\overline{t}\ \overline{n}\ \overline{w}$

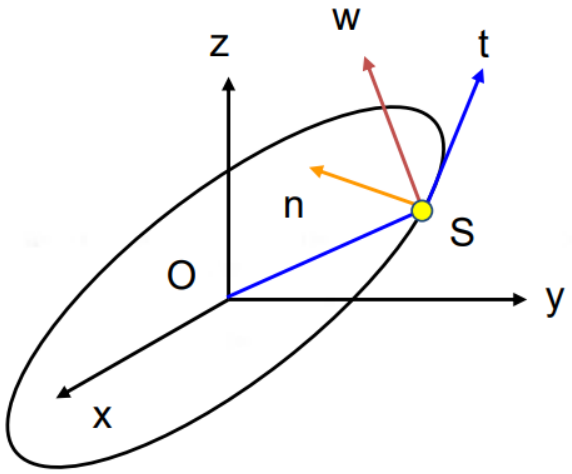


Figura 4.4: Sistema de referencia local (ESTACA/CNES)

En este caso se tiene en cuenta principalmente el rozamiento atmosférico y las maniobras orbitales.

$$\begin{cases} \frac{da}{dt} = \frac{2}{nt} V\mathbf{T} \\ \frac{de}{dt} = \frac{1}{V}\left[2(e+\cos\nu)\mathbf{T} - \sin\nu\frac{r}{a}\mathbf{N}\right] \\ \frac{di}{dt} = \frac{r\cos(\omega+\nu)}{na^2(1-e^2)^{1/2}}\mathbf{W} \\ \frac{d\omega}{dt} = \frac{1}{Ve}\left[2\mathbf{T}\sin\nu + \mathbf{N}\frac{2e+\cos\nu+e^2\cos\nu}{1+e\cos\nu}\right] - \frac{r\cos i\sin(\omega+\nu)}{na^2(1-e^2)^{1/2}\sin i}\mathbf{W} \\ \frac{d\Omega}{dt} = \frac{r\sin(\omega+\nu)}{na^2(1-e^2)^{1/2}\sin i}\mathbf{W} \\ \frac{dM}{dt} = n - \frac{\sqrt{1-e^2}}{eV}\left[2\mathbf{T}\sin\nu\left(1+\frac{e^2}{1+e\cos\nu}\right) + \mathbf{N}\cos\nu\frac{1-e^2}{1+e\cos\nu}\right] \end{cases} \tag{4.15}$$

Capítulo 5

Implementación del modelo IGRF13

En este capítulo se puede ver como se modela el campo magnético terrestre, así como su programación numérica.

5.1. FUNDAMENTOS MATEMÁTICOS

El campo magnético terrestre es de vital importancia para el control de actitud de cualquier tipo de vehículo espacial. Es por ello que existe numerosos modelos que permiten realizar simulaciones del mismo, y el hecho de conocer el campo magnético puede permitir una optimización y un ahorro energético en el subsistema de control de actitud del satélite (ACS). Es de especial interés el uso del campo magnético en órbitas LEO, ya que es en estas donde se puede aprovechar la mayor intensidad del mismo para el control de actitud de una manera eficiente. Una de las principales problemáticas que se encuentran es la variación del mismo a lo largo del tiempo [13].

La influencia del campo geomagnético en el torque que afecta a un satélite se puede expresar como [14]:

$$\mathbf{T} = \dot{\mathbf{h}}_I = \dot{\mathbf{h}}_B + \omega_T \times \mathbf{h}_B, \tag{5.1}$$

donde T denota el vector de momentos externos, h es el vector de momento angular, y ω muestra el vector de velocidad angular los subíndices I y B se refieren al marco de inercia y al marco del cuerpo, respectivamente.

$$\mathbf{T} = \mathbf{T}_c + \mathbf{T}_d, \tag{5.2}$$

donde los momentos externos T se denotan como suma de T_d siendo el torque de las perturbaciones y T_c el torque debido al control.

$$\mathbf{T}_c = \mathbf{T}_{mag} = \mathbf{m} \times \mathbf{B}, \tag{5.3}$$

se produce por el producto vectorial del dipolo magnético m y el vector de campo magnético $\mathbf{B}$.

El campo magnético puede expresarse como el gradiente negativo de una función potencial V:

$$\mathbf{B} = -\nabla V. \tag{5.4}$$

Para su modelización se va a proceder de manera análoga al modelado y simulación del campo gravitatorio terrestre. Para ello se expresa este potencial V como una serie de harmónicos.

$$V(r,\phi,\theta) = R_e \sum_{n=1}^{N} \left(\frac{R_e}{r}\right)^{n+1} \sum_{m=0}^{n} \left[g_n^m \cos(m\phi) + h_n^m \sin(m\phi)\right] P_n^m(\theta). \tag{5.5}$$

donde R_e es el radio ecuatorial, r es la distancia desde el centro de la Tierra y ϕ y θ son latitud y longitud respectivamente. En cuanto a g_n^m y h_n^m, estos son el conjunto de coeficientes de Gauss que se utilizan en los modelos analíticos que describen el campo magnético terrestre se denomina Campo Geomagnético Internacional de Referencia (IGRF). Los coeficientes de Gauss y pueden obtenerse actualizados de [15]. Estos datos solo son válidos durante un periodo de tiempo de unos 5 años, luego será necesario cada vez que se ejecute el programa descargar los datos más recientes del momento. Para tener más detalles de la estructura de los datos y sus mecanismos de obtención, se puede consultar [16]. Las unidades del campo son nT. Al tratarse de un campo conservativo ($\nabla \times \mathbf{B}$), se puede expresar el campo magnético en coordenadas esféricas de la siguiente manera [13].

$$B_r = -\frac{\partial V}{\partial r} = \sum_{n=1}^{N} \left(\frac{R_e}{r}\right)^{n+2} (n+1) \sum_{m=0}^{n} \left[g_n^m \cos(m\phi) + h_n^m \sin(m\phi)\right] P_n^m(\theta), \tag{5.6}$$

$$B_\theta = -\frac{1}{r}\frac{\partial V}{\partial \theta} = -\sum_{n=1}^{N} \left(\frac{R_e}{r}\right)^{n+2} \sum_{m=0}^{n} \left[g_n^m \cos(m\phi) + h_n^m \sin(m\phi)\right] \frac{\partial P_n^m(\theta)}{\partial \theta}, \tag{5.7}$$

$$B_\phi = \frac{-1}{r\sin(\theta)}\frac{\partial V}{\partial \phi} = \frac{-1}{\sin(\theta)} \sum_{n=1}^{N} \left(\frac{R_e}{r}\right)^{n+2} \sum_{m=0}^{n} \left[-m g_n^m \sin(m\phi) + m h_n^m \cos(m\phi)\right] P_n^m(\theta). \tag{5.8}$$

Estas ecuaciones serán implementadas y sus resultados se muestran en 5.2.5.

5.2. DESARROLLO NUMÉRICO

Una vez que se ha visto la base matemática del modelado del campo magnético, se puede apreciar su semejanza con el campo potencial terrestre de la sección 6. Para modelar dicho campo se van a elaborar diferentes modelos con diferentes grados de

aproximaciones que se irán explicando en los siguientes apartados.

Antes de nada, como se mencionó en párrafos anteriores, es necesario actualizar los datos para poder tener un modelo válido. Para lo anterior, se van a extraer los datos de la web NCEI donde se pueden obtener los datos actualizados.

En primer lugar, se van a extraer todos los archivos *.txt* o *.dat* que se encuentran en

https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field,

una vez que se tienen todos los archivos, se debe considerar si estos son los que son necesarios, para ello se va a mirar si el nombre del campo magnético **IGRF** se encuentra en los nombres de los archivos. Llegados hasta este punto, y observando que el nombre de los archivos sigue siempre el mismo patrón, el nombre del modelo y el número de la versión del mismo, se va a evaluar cuál es la versión más reciente y se va a cargar el fichero, el cual se leerá a posteriori. En el caso en el que el fichero ya se tenga precargado, para evitar una pérdida de tiempo y de recursos innecesarios, se van a leer los datos precargados que se habían descargado anteriormente, es decir, si hay un nuevo modelo, este último será descargado, en caso contrario se utilizarán los datos anteriores. Lo actualmente mencionado se puede ver en la función **get_all_txt_from_url** A.4.

Tras inspeccionar la web, se extraen todos los archivos que se encuentran en la misma con formato *.txt* como se muestra a continuación.

['https://www.ngdc.noaa.gov/IAGA/vmod/coeffs/igrf13coeffs.txt',
'https://ngdc.noaa.gov/IAGA/vmod/coeffs/igrf9coeffs.txt',
'https://ngdc.noaa.gov/IAGA/vmod/coeffs/igrf10coeffs.txt',
'https://ngdc.noaa.gov/IAGA/vmod/coeffs/igrf11coeffs.txt',
'https://ngdc.noaa.gov/IAGA/vmod/coeffs/igrf12coeffs.txt',
'https://ngdc.noaa.gov/IAGA/vmod/coeffs/igrf13coeffs.txt']
igrf13 'https://www.ngdc.noaa.gov/IAGA/vmod/coeffs/igrf13coeffs.txt'

Para el procesado de los datos se utiliza otra función que ha sido desarrollada, **get_gh_data** estos datos se corresponden con [16], como se puede apreciar, se tienen los coeficientes h y g del campo magnético ordenados por orden y grado, así como por fechas. En el caso del orden y el grado, se observa un límite de 13, por otro lado, las fechas, están distribuidas de 5 en 5 años, comenzando en 1900 y acabando en la actualidad. Todo esto lo se puede apreciar en la tabla siguiente, donde la última columna hace referencia a las variaciones seculares que permiten calcular un valor del campo magnético aproximado a 5 años vista desde la última actualización de los datos.

g/h	**n**	m	2000.0	2005.0	2010.0	2015.0	2020.0	2020-25
g	1	0	-29619.4	-29554.63	-29496.57	-29441.46	-29404.8	5.7
g	1	1	-1728.2	-1669.05	-1586.42	-1501.77	-1450.9	7.4
h	1	1	5186.1	5077.99	4944.26	4795.99	4652.5	-25.9
g	2	0	-2267.7	-2337.24	-2396.06	-2445.88	-2499.6	-11.0
g	2	1	3068.4	3047.69	3026.34	3012.20	2982.0	-7.0
h	2	1	-2481.6	-2594.50	-2708.54	-2845.41	-2991.6	-30.2
g	2	2	1670.9	1657.76	1668.17	1676.35	1677.0	-2.1
h	2	2	-458.0	-515.43	-575.73	-642.17	-734.6	-22.4
g	3	0	1339.6	1336.30	1339.85	1350.33	1363.2	2.2
g	3	1	-2288.0	-2305.83	-2326.54	-2352.26	-2381.2	-5.9

Tabla 5.1: Ejemplo de los coeficientes h y g por año y las variaciones seculares

Una vez que se tienen los coeficientes **g** y **h**, hace falta obtener los polinomios asociados normalizados de Legendre, para ello se utilizará un método recursivo que permite su cálculo. Este método da directamente los coeficientes normalizados con la cuasinormalización de Schmidt:

$$\begin{aligned} P^{0,0} &= 1, \\ P^{n,n} &= \sin\theta P^{n-1,n-1}, \\ P^{n,m} &= \cos\theta P^{n-1,m} - K^{n,m} P^{n-2,m}, \\ K^{n,m} &= 0 \qquad n = 1, \\ K^{n,m} &= \frac{(n-1)^2 - m^2}{(2n-1)(2n-3)} \qquad n > 1, \end{aligned} \tag{5.9}$$

De manera análoga se obtiene las derivadas de los polinomios normalizados de Legendre.

$$\begin{aligned} \frac{\partial P^{0,0}}{\partial\theta} &= 0, \\ \frac{\partial P^{n,n}}{\partial\theta} &= \sin\theta \frac{\partial P^{n-1,n-1}}{\partial\theta} + \cos\theta P^{n-1,n-1}, \\ \frac{\partial P^{n,m}}{\partial\theta} &= \cos\theta \frac{\partial P^{n-1,m}}{\partial\theta} + \sin\theta P^{n-1,m} - K^{n,m} \frac{\partial P^{n-2,m}}{\partial\theta}, \end{aligned} \tag{5.10}$$

Los coeficientes anteriores están ya normalizados, pero en el caso de los coeficientes que se extraen de la web, estos no lo están. Es por ello que se va a calcular el factor de normalización que posteriormente se aplicará a los coeficientes h y g.

$$S_{n,m} = \left[\frac{(2-\delta_m^0)(n-m)!}{(n+m)!}\right]^{1/2} \frac{(2n-1)!}{(n-m)!}. \tag{5.11}$$

.

Análogamente, a la obtención de los polinomios asociados normalizados de Legendre se puede utilizar la siguiente expresión recursiva para el cálculo del coeficiente de normalización $S_{n,m}$.

$$\begin{aligned} S_{0,0} &= 1, \\ S_{n,0} &= S_{n-1,0}\left(\frac{2n-1}{n}\right) \qquad n >= 1, \\ S_{n,m} &= S_{n,m-1}\sqrt{\frac{(n-m+1)(\delta_m^1+1}{n+m}} \qquad m >= 1, \end{aligned} \tag{5.12}$$

Si se normalizan los coeficientes anteriormente extraídos de la base de datos, se tiene:

$$\begin{aligned} g^{n,m} &= S_{n,m} g_n^m, \\ h^{n,m} &= S_{n,m} h_n^m, \end{aligned} \tag{5.13}$$

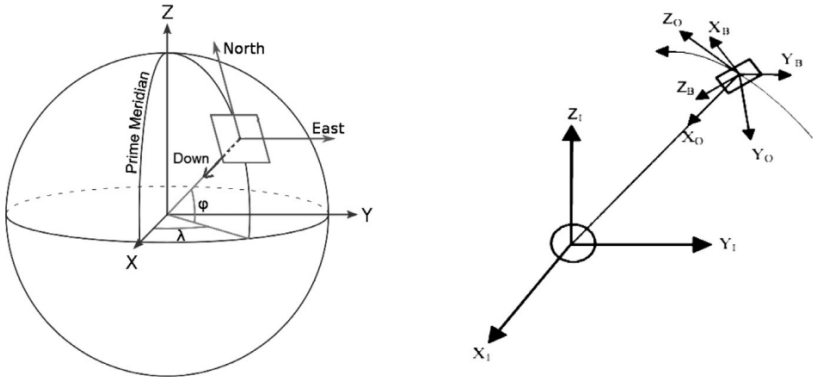


Figura 5.1: Coordenadas [17]

A continuación, se desarrolla basándose en lo anteriormente visto una serie de aproximaciones cuyo coste computacional es significativamente menor, por lo que se van a mostrar 3 aproximaciones con diferente grado de precisión.

5.2.1. APROXIMACIÓN 1: DIPOLO

Este es uno de los modelos más simples cuya formulación se puede ver a continuación en las ecuaciones 5.14.

$$\begin{aligned} B_r &= 2\left(\tfrac{R_e}{r}\right)^3 \left[g^{1,0}\cos\phi + (g^{1,1}\cos\phi + h^{1,1}\sin\phi)\sin\theta\right], \\ B_\theta &= \left(\tfrac{R_e}{r}\right)^3 \left[g^{1,0}\sin\phi - (g^{1,1}\cos\phi + h^{1,1}\sin\phi)\cos\theta\right], \\ B_\phi &= \left(\tfrac{R_e}{r}\right)^3 (g^{1,1}\sin\phi - h^{1,1}\cos\phi). \end{aligned} \tag{5.14}$$

En la siguiente figura 5.2 se puede apreciar la evolución del campo magnético desde 1900 hasta 2025 en un punto del espacio cualquiera. para este ejemplo se ha utilizado una co-elevación de 10° así como una longitud este de 10°, por otro lado, $r = 7000$ km.

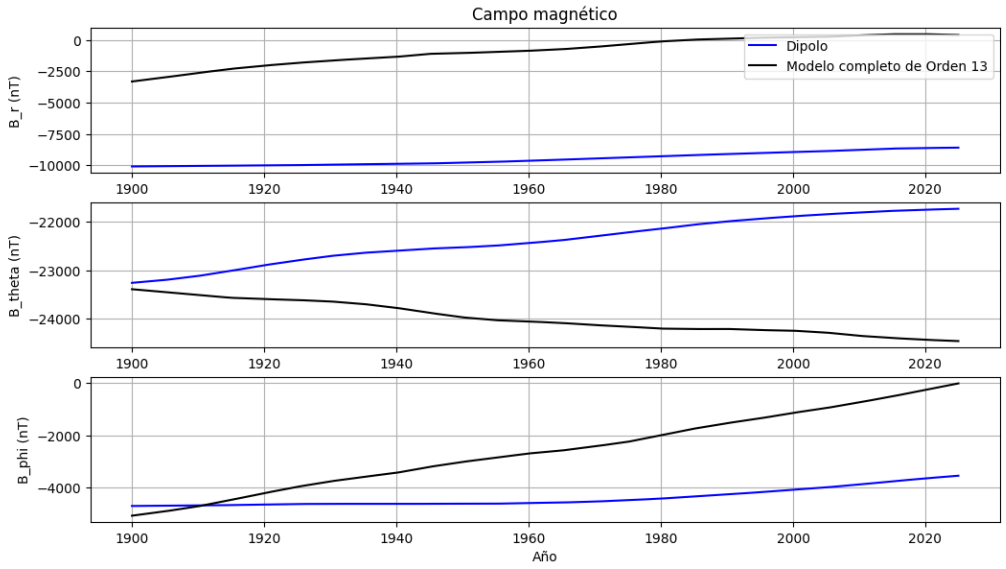


Figura 5.2: Modelo dipolo

5.2.2. APROXIMACIÓN 2: DIPOLO CENTRADO

Este es uno de los modelos más simples cuya formulación se puede ver a continuación en las ecuaciones 5.15.

$$\begin{aligned} B_r &= 2\left(\tfrac{R_e}{r}\right)^3 (g^{1,0}\cos\phi), \\ B_\theta &= \left(\tfrac{R_e}{r}\right)^3 (g^{1,0}\sin\phi - (g^{1,1}), \\ B_\phi &= 0. \end{aligned} \tag{5.15}$$

En la figura 5.3 se puede ver el modelo dipolo centrado comparado con el modelo dipolo.

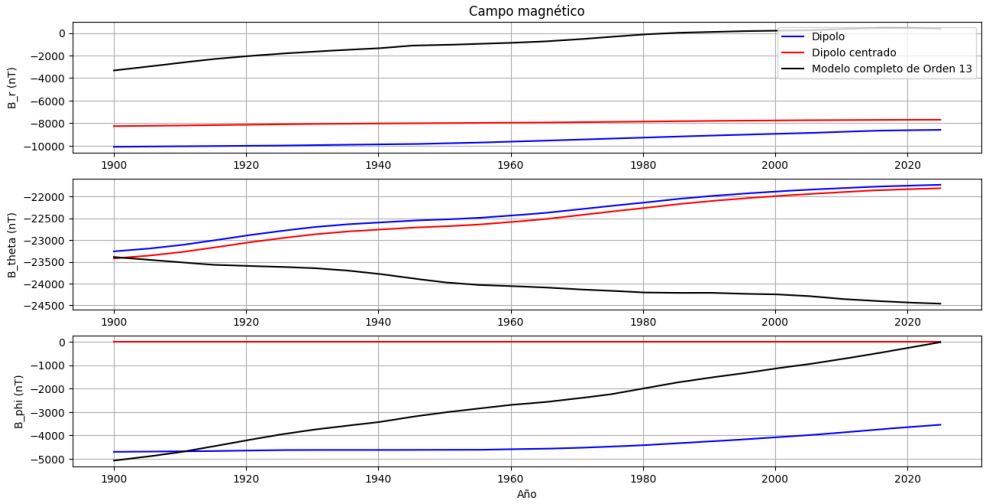


Figura 5.3: Modelo dipolo centrado

5.2.3. APROXIMACIÓN 3: CUADRUPOLO

Este es uno de los modelos más simples de grado N=2 cuya formulación se puede ver a continuación en las ecuaciones 5.16.

$$\begin{aligned}
B_r &= B_r^{dipole} + 3\left(\tfrac{R_e}{r}\right)^4 \left[\tfrac{1}{2}g^{2,0}(\cos 2\phi + \tfrac{1}{3}) + \tfrac{1}{2}(g^{2,1}\cos\phi + h^{2,1}\sin\phi)\sin 2\theta + \right. \\
&\quad \left. \tfrac{1}{2}(g^{2,2}\cos 2\phi + h^{2,2}\sin 2\phi)(1-\cos 2\theta)\right], \\
B_\theta &= B_\theta^{dipole} + \left(\tfrac{R_e}{r}\right)^4 \left[g^{2,0}\sin 2\phi + (g^{2,1}\cos\phi + h^{2,1}\sin\phi)\cos 2\theta - \right. \\
&\quad \left. (g^{2,2}\cos 2\phi + h^{2,2}\sin 2\phi)\sin 2\theta\right], \\
B_\phi &= B_\phi^{dipole} + \left(\tfrac{R_e}{r}\right)^4 \left[(g^{2,1}\sin\phi - h^{2,1}\cos\phi)\cos\theta - 2(g^{2,2}\sin 2\phi - h^{2,2}\cos 2\phi)\sin\theta\right].
\end{aligned} \tag{5.16}$$

En la figura 5.4 se puede ver el modelo cuadrupolo comparado con el modelo dipolo, y se aprecian tendencias similares, pero también diferencias como en la componente B_θ.

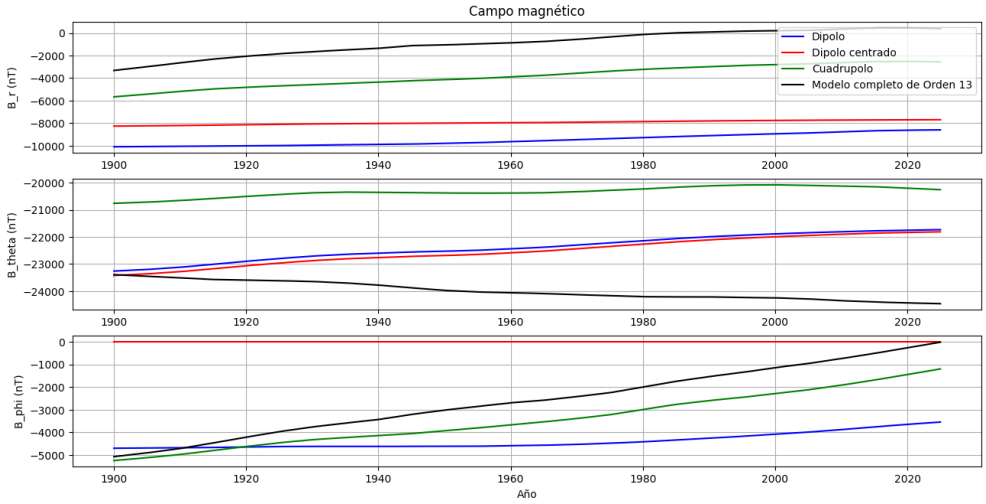


Figura 5.4: Modelo cuadrupolo

5.2.4. APROXIMACIÓN 4: OCTUPOLO

Un modelo un poco más complejo que los anteriores se presenta en esta sección. Cabe resaltar que para su construcción se utiliza de base B_r^{quad}, B_θ^{quad} y B_ϕ^{quad}. Este modelo se considera de grado 3, ya que llega a utilizar coeficientes g y h de grado y orden 3. Para su modelado se ha utilizado la siguiente formulación matemática 5.17.

$$
\begin{aligned}
B_r = B_r^{quad} + 4\left(\tfrac{R_e}{r}\right)^5 \big[&\tfrac{1}{2}g^{3,0}\cos\theta(\cos 2\phi - \tfrac{1}{5}) + \\
&\tfrac{1}{2}(g^{3,1}\cos\phi + h^{3,1}\sin\phi)\sin\theta\left(\cos 2\theta - \tfrac{3}{5}\right) + \\
&\tfrac{1}{2}(g^{3,2}\cos 2\phi + h^{3,2}\sin 2\phi)\cos\theta(1-\cos 2\theta) + \\
&\tfrac{1}{2}(g^{3,3}\cos 3\phi + h^{3,3}\sin 3\phi)\sin\theta(1-\cos 2\theta)\big] , \\
B_\theta = B_\theta^{quad} - 3\left(\tfrac{R_e}{r}\right)^5 \big[&\tfrac{1}{2}g^{3,0}\sin\theta(\cos 2\phi - \tfrac{3}{5}) + \\
&\tfrac{1}{2}(g^{3,1}\cos\phi + h^{3,1}\sin\phi)\cos\theta\left(\cos 2\theta - \tfrac{2}{5}\right) + \\
&\tfrac{1}{2}(g^{3,2}\cos 2\phi + h^{3,2}\sin 2\phi)\sin\theta(\cos 2\theta - \tfrac{1}{3}) + \\
&\tfrac{1}{2}(g^{3,3}\cos 3\phi + h^{3,3}\sin 3\phi)\sin\theta(1-\cos 2\theta)\big] , \\
B_\phi = B_\phi^{quad} + \left(\tfrac{R_e}{r}\right)^5 \big[&\tfrac{1}{2}(g^{3,1}\sin\phi - h^{3,1}\cos\phi)\left(\cos 2\theta + \tfrac{3}{5}\right) + \\
&(g^{3,2}\sin 2\phi - h^{3,2}\cos 2\phi)\sin 2\theta + \tfrac{3}{2}(g^{3,3}\sin 3\phi + h^{3,3}\cos 3\phi)(1-\cos 2\theta)\big] .
\end{aligned}
\tag{5.17}
$$

Tras realizar simulaciones del modelo en el mismo punto que los ejemplos anteriores con modelos de orden inferior, se obtiene la siguiente gráfica 5.5.

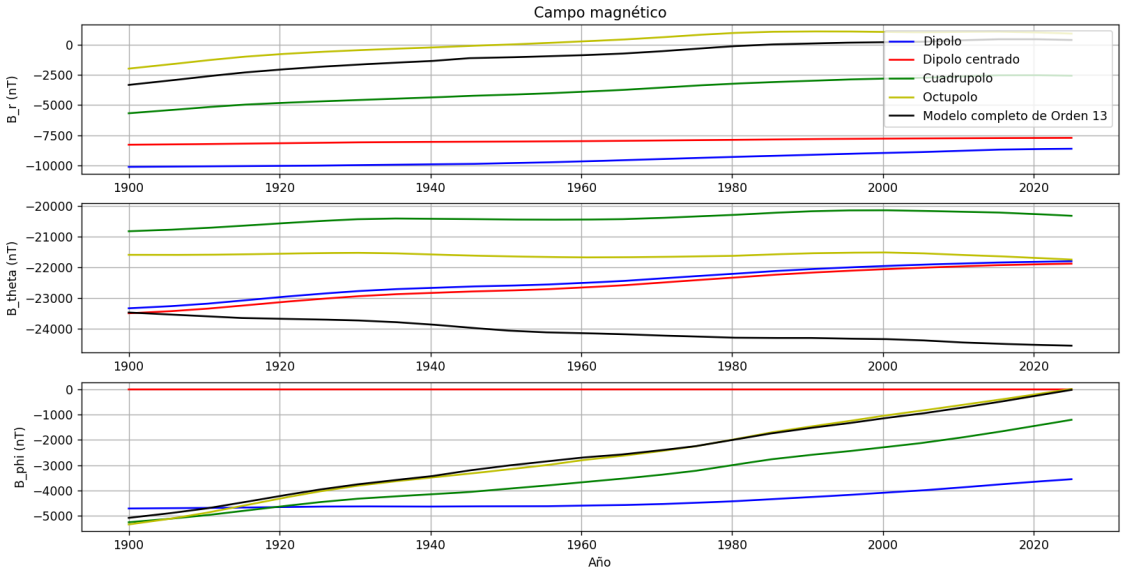


Figura 5.5: Modelo cuadrupolo

Se puede apreciar como en este caso las curvas de los modelos cuadrupolo y octupolo son muy similares con la misma forma de la curva pero con cierto desfase.

5.2.5. MODELOS DE ORDEN SUPERIOR

En esta sección se presentan las simulaciones del campo magnético teniendo en cuenta un número elevado de órdenes del modelo. Para ellos su utilizan las siguientes ecuaciones 5.6, 5.7 y 5.8. Se pueden apreciar los resultados en la figura 5.6.

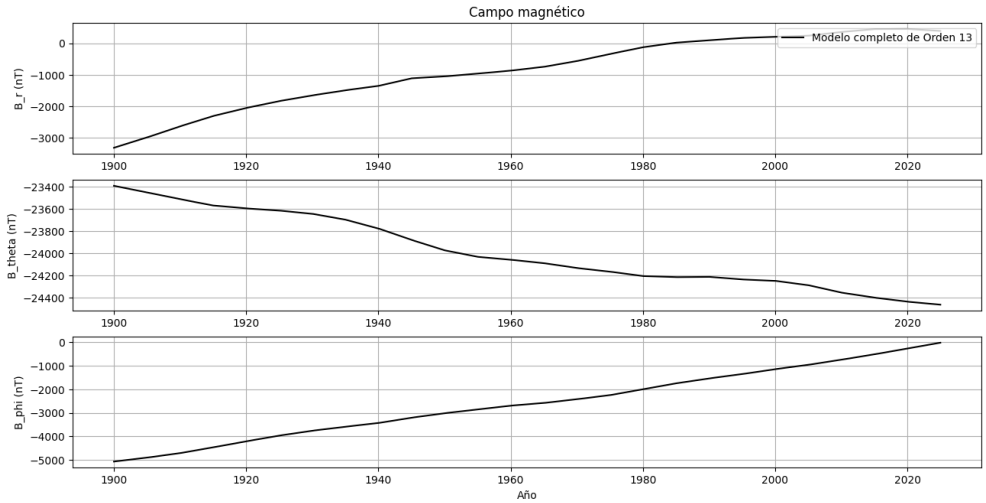


Figura 5.6: Modelo completo

El código se puede apreciar en el anexo A.5

5.2.6. CAMBIOS DE COORDENADAS

En esta sección se van a tratar de realizar unos cambios de coordenadas para poder representar el campo magnético en sistema de referencia que sea de mayor utilidad. Se trata un primer cambio de coordenadas que permite pasar de coordenadas tangenciales locales (coordenadas esféricas) a coordenadas cartesianas tangenciales locales. A continuación se pasará a coordenadas inerciales, posteriormente a coordenadas orbitales y finalmente a ejes cuerpo.

Coordenadas cartesianas tangenciales locales

Las coordenadas cartesianas tangenciales locales NED (North-East-Down) son un sistema de coordenadas geográficas basado en el plano tangente definido por la dirección vertical local y el eje de rotación de la Tierra. Este sistema se forma a partir de un plano tangente a la superficie de la Tierra fijada a una ubicación específica y, por lo tanto, a veces se lo conoce como plano "Tangente local" o "geodésico local". Por convención, el eje este está etiquetado como "Norte", el norte como "Este" y el arriba como "Abajo".

$$\begin{aligned} X &= -B_\theta \cos\epsilon - B_r \sin\epsilon, \\ Y &= B_\phi, \\ Z &= B_\theta \sin\theta - B_r \cos\epsilon, \end{aligned} \tag{5.18}$$

donde ϵ es un término usado para corregir el achatamiento de la Tierra. En los programas en línea, este término se establece en cero [17].

$$\epsilon = \lambda - \delta < 0{,}2, \tag{5.19}$$

donde λ es la latitud geodésica.

En el siguiente esquema de la figura 5.7 se puede ver gráficamente el significado de estas coordenadas.

Su cálculo se encuentra en **def transformation2NED(self, r, theta, phi, lambdaa, Bvector)** dentro de la clase **geomag**

HFID

En esta sección se presenta la obtención de algunos valores significativos que sirven para representar el campo magnético, esto son los siguientes:

- **H:** Componente horizontal del campo magnético.

$$H = \sqrt{X^2 + Y^2}. \tag{5.20}$$

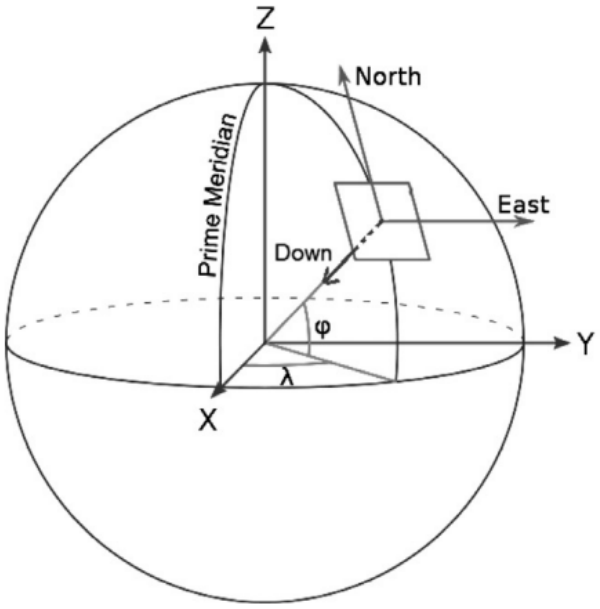


Figura 5.7: Coordenadas NED [17]

- **F:** Intensidad total del campo magnético.

$$F = \sqrt{X^2 + Y^2 + Z^2}. \tag{5.21}$$

- **I:** Inclinación.

$$\arctan\left(\frac{Z}{H}\right). \tag{5.22}$$

- **D:** Declinación.

$$\arctan\left(\frac{Y}{X}\right). \tag{5.23}$$

Su cálculo se encuentra en **def xyz2dhif(x, y, z)** dentro de la clase **geomag**.

Sistema de coordenadas inercial

Para que los resultados de la Ecuación (3) sean efectivos en el trabajo satelital, deben convertirse a componentes inerciales geocéntricos, usando las siguientes ecuaciones:

$$\begin{aligned} B_x &= (B_r \cos\delta + B_\theta \sin\delta)\cos\alpha - B_\phi \sin\alpha, \\ B_y &= (B_r \cos\delta + B_\theta \sin\delta)\sin\alpha + B_\phi \cos\alpha, \\ B_z &= (B_r \sin\delta + B_\theta \cos\delta), \end{aligned} \tag{5.24}$$

donde δ es la latitud medida al norte positivo desde el ecuador (declinación), y α es el tiempo sideral local de la ubicación en cuestión [17].

$$\alpha = long - \theta_g \qquad\qquad \delta = 90 - lat, \tag{5.25}$$

donde θ_g es el tiempo celestial en el meridiano de Greenwich.

Su cálculo se encuentra en **def transformation2inertial(self, Bvector, theta, phi, thetag = 0)** dentro de la clase **geomag**.

Sistema de coordenadas orbital

La función **transformation2orb** realiza una transformación del campo magnético terrestre desde coordenadas inerciales a coordenadas orbitales. Acepta los siguientes parámetros:

- `Bvector`: un vector que representa el campo magnético en coordenadas r, θ, ϕ.
- `orbparam`: un arreglo que contiene los parámetros orbitales, que incluyen la verdadera anomalía, la ascensión recta del nodo ascendente y la inclinación, todos en grados.
- `phi`: la co-elevación del punto donde se calcula el campo magnético, en grados.
- `theta`: la longitud este del punto donde se calcula el campo magnético, en grados.
- `thetag`: la declinación del meridiano de Greenwich, en grados (valor predeterminado: 0).

La función devuelve el campo magnético en la ubicación dada en coordenadas orbitales. Las ecuaciones utilizadas para realizar la transformación son bastante extensas y complejas. A continuación se muestra la representación en LaTeX de las ecuaciones utilizadas en la función `transformation2orb`:

$$\begin{aligned}
B_{xO} = {} & (-\sin(t_a)\cos(RAAN) - \cos(RAAN)\sin(t_a)\cos(i))(Br\cos(\delta)\cos(\alpha) + \\
& B_\theta \sin(\delta)\cos(\alpha) - B_\phi \sin(\alpha)) + (-\sin(t_a)\sin(RAAN) \\
& \cos(t_a)\cos(RAAN)\cos(i))(Br\cos(\delta)\sin(\alpha) + B_\theta \sin(\delta)\sin(\alpha) + B_\phi \cos(\alpha)) \\
& + \cos(t_a)\sin(i)(Br\sin(\delta) - B_\theta \cos(\delta)), \\
B_{yO} = {} & -\sin(i)\sin(RAAN)(Br\cos(\delta)\cos(\alpha) + B_\theta \sin(\delta)\cos(\alpha) - B_\phi \sin(\alpha)) \\
& + \sin(i)\cos(RAAN)(Br\cos(\delta)\sin(\alpha) + B_\theta \sin(\delta)\sin(\alpha) + B_\phi \cos(\alpha)) \\
& - \cos(i)(Br\sin(\delta) - B_\theta \cos(\delta)), \\
B_{zO} = {} & (-\cos(t_a)\cos(RAAN) + \sin(t_a)\sin(RAAN)\cos(i))(Br\cos(\delta)\cos(\alpha) + \\
& B_\theta \sin(\delta)\cos(\alpha) - B_\phi \sin(\alpha)) + (-\cos(t_a)\sin(RAAN) - \sin(t_a)\cos(RAAN)\cos(i)) \\
& (Br\cos(\delta)\sin(\alpha) + B_\theta \sin(\delta)\sin(\alpha) + B_\phi \cos(\alpha)) \\
& - \sin(t_a)\sin(i)(Br\sin(\delta) - B_\theta \cos(\delta)).
\end{aligned} \tag{5.26}$$

Estas ecuaciones describen cómo se realiza la transformación del campo magnético desde las coordenadas inerciales a las coordenadas orbitales.

Sistema de coordenadas ejes cuerpo

La transformación del campo magnético terrestre de coordenadas inerciales a ejes cuerpo es un proceso fundamental en la navegación espacial. Esta transformación se realiza utilizando los parámetros de actitud de la nave espacial, que incluyen el cabeceo, el alabeo y la guiñada en grados. Además, se requiere el vector del campo magnético en coordenadas orbitales. El resultado de esta transformación es el campo magnético en el marco de referencia del cuerpo de la nave espacial, proporcionando información crucial para la navegación y la orientación.

$$\begin{aligned} B_{xB} &= B_{xO} \cdot \cos(\phi) \cdot \cos(\theta) + B_{yO} \cdot (\cos(\phi) \cdot \sin(\theta) \cdot \sin(\beta) - \sin(\phi) \cdot \cos(\beta)) + \\ &\quad B_{zO} \cdot (\cos(\phi) \cdot \sin(\theta) \cdot \cos(\beta) + \sin(\phi) \cdot \sin(\beta)), \\ B_{yB} &= B_{xO} \cdot \sin(\phi) \cdot \cos(\theta) + B_{yO} \cdot (\sin(\phi) \cdot \sin(\theta) \cdot \sin(\beta) + \cos(\phi) \cdot \cos(\beta)) + \\ &\quad B_{zO} \cdot (\sin(\phi) \cdot \sin(\theta) \cdot \cos(\beta) - \cos(\phi) \cdot \sin(\beta)), \\ B_{zB} &= -B_{xO} \cdot \sin(\theta) + B_{yO} \cdot \cos(\theta) \cdot \sin(\beta) + B_{zO} \cdot \cos(\theta) \cdot \cos(\beta), \end{aligned} \tag{5.27}$$

donde θ es el pitch o cabezada, ϕ es el yaw o guiñada, y β es el roll o alabeo.

Estas ecuaciones representan la transformación del campo magnético desde las coordenadas orbitales a las coordenadas del cuerpo en función de los ángulos de orientación (yaw, pitch, roll) del cuerpo.

Sistema de coordenadas ejes cuerpo reducido

Si al sistema de coordenadas ejes cuerpo anterior, se le aplican ciertas hipótesis como la suposición de ángulos de Euler pequeños, se tiene que:

$$\begin{aligned} \cos(x) &\approx 1, \\ \sin(x) &\approx 0. \end{aligned} \tag{5.28}$$

Si se añade esta condiciones al sistema de ecuaciones anterior se obtienen las componentes del campo magnético en el sistema de ejes cuerpo reducido.

$$\begin{aligned} B_{xB} &= B_{xO} + B_{yO}\phi + B_{zO}\theta, \\ B_{yB} &= B_{xO}\phi + B_{yO} - \beta B_{zO}, \\ B_{zB} &= -B_{xO}\theta + B_{yO}\beta + B_{zO}. \end{aligned} \tag{5.29}$$

5.2.7. OTRAS HERRAMIENTAS

Para poder utilizar este modelo, se han desarrollado algunas funciones extras.

gg_to_geo(self, h, gdcolat) y geo_to_gg(self, radius, theta)

Esta función permite pasar de la co-latitud geocéntrica y la distancia desde el centro de la Tierra a coordenadas geodésicas. Este código ha sigo extraído de Github de un repositorio de la NNOA con su correspondiente citación dentro del propio código.

A continuación se muestra, la formulación de Heikkinen que se programa en estas funciones.

En primer lugar, se va a calcular la colatitud geodésica y la altura vertical sobre el elipsoide a partir del radio geocéntrico y la colatitud. Se va a utilizar el WGS84 como elipsoide de referencia, donde el radio ecuatorial es $a = 6378{,}137$, el radio polar $b = 6356{,}752$. El algoritmo a utilizar es el desarrollador por Heikkinen, quién estableció una ecuación cuártica y encontró una solución sin singularidades en el plano ecuatorial [18].

$$
\begin{gathered}
e^2 = \tfrac{a^2-b^2}{a^2}, \\
e_p^2 = \tfrac{a^2-b^2}{b^2}, \\
r = R\sin(\theta), \\
z = R\cos(\theta), \\
F = 54b^2z^2, \\
G = r^2 + (1-e^2)z^2 - e^2(a^2-b^2), \\
c = \tfrac{e^4Fr^2}{G^3}, \\
s = \left(1 + c + \sqrt{c^2+2c}\right)^{\frac{1}{3}}, \\
P = \tfrac{F}{3(s+\frac{1}{s}+1)^2G^2}, \\
Q = \sqrt{1+2e^4P}, \\
r_0 = -\tfrac{Pe^2r}{1+Q} + \sqrt{\tfrac{1}{2}a^2\left(1+\tfrac{1}{Q}\right) - \tfrac{P(1-e^2)z^2}{Q(1+Q)} - \tfrac{1}{2}Pr^2}, \\
U = \sqrt{(r-e^2r_0)^2 + z^2}, \\
V = \sqrt{(r-e^2r_0)^2 + (1-e2)z^2}, \\
z_0 = \tfrac{b^2z}{aV}, \\
altura = U(1-\tfrac{b^2}{aV}), \\
\beta = 90 - \arccos(\tfrac{z+e_p^2z_0}{r}).
\end{gathered}
\tag{5.30}
$$

Esta función calcula la altura y la colatitud geodésica de un satélite a partir de su radio y su colatitud geocéntrica. Los parámetros de entrada y salida son:

Los parámetros de entrada son:

- R: radio geocéntrico del satélite en kilómetros.
- θ: colatitud geocéntrica del satélite en grados.

Los parámetros de salida son:

- altura (h): altura del satélite en kilómetros.

- gdcolat (β): colatitud geodésica del satélite en grados.

De manera análoga, para calcular la colatitud geocéntrica y el radio a partir de la colatitud geodésica y la altura se utilizan los algoritmos que se encuentra en la referencia [19]. Para ello, se usa un factor de achatamiento $f = \frac{1}{298{,}257223563}$, un radio polar de $pl_{rad} = r_{ecuatorial} \cdot (1 - f)$.

$$\begin{gathered} c^2 = \cos(cl_{geo})^2, \\ s^2 = 1 - c^2, \\ \rho = \sqrt{r^2_{ecuatorial} s^2 + pl^2_{rad} c^2}, \\ rad = \sqrt{h(h+2\rho) + (r^4_{ecuatorial} s^2 + pl^4_{rad} c^2)/\rho^2}, \\ cd = \tfrac{h+\rho}{rad}, \\ sd = \tfrac{(r^2_{ecuatorial} - pl^2_{rad}) \cos(cl_{geo}) \sin(cl_{geo})}{\rho rad}, \\ cthc = \cos(cl_{geo}) cd - \sin(cl_{geo}) sd, \\ thc = \arccos(cthc). \end{gathered} \tag{5.31}$$

Esta función calcula el radio geocéntrico y la colatitud geocéntrica de un satélite a partir de su altura y su colatitud geodésica. También calcula los factores de rotación para transformar el campo magnético de un sistema de coordenadas geodésico a uno geocéntrico.

Los parámetros de entrada son:

- h: altura del satélite en kilómetros.
- gdcolat: colatitud geodésica del satélite en grados.

Los parámetros de salida son:

- rad: radio geocéntrico del satélite en kilómetros.
- thc: colatitud geocéntrica del satélite en grados.
- sd: factor de rotación para el componente B_X del campo magnético.
- cd: factor de rotación para el componente B_Z del campo magnético.

arrayofpoints(self, r, theta, phi, year=2020, N=3, savedata=False)

Esta función de la clase **geomag** permite obtener y guardar en un fichero los datos del campo magnético en un conjunto de localizaciones para una fecha determinada. Todos estos valores se extraen en un fichero *.csv* para su posterior utilización para otras aplicaciones.

r (km)	theta (deg)	phi (deg)	Br (nT)	Btheta (nT)	Bphi (nT)	BN (nT)	BE (nT)	BD (nT)
7000	-90	-180	39066,1	9317,5	6368,7	-9317,5	6368,71	-39066,4
7111,1	-70	-140	38334,9	-5640,9	9320,2	5640,9	9320,28	-38334,9
7222,2	-50	-100	21881,4	-13703,9	5972,6	13703,9	5972,63	-21881,4
7333,3	-30	-60	9316,4	-12987,3	-1810,2	12987,3	-1810,22	-9316,4
7444,4	-10	-20	9912,9	-13118,2	-3598,0	13118,2	-3598,01	-9912,9
7555,5	10	20	-827,75	-19232,9	52,47	19232,9	52,47	827,7
7666,6	30	60	-18319,2	-17781,4	493,91	17781,4	493,91	18319,2
7777,8	50	100	-28709,0	-11378,0	-341,64	11378,0	-341,64	28709,0
7888,8	70	140	-30492,3	-5515,1	-583,98	5515,1	-583,98	30492,3
8000	90	180	-29925,8	418,14	410,16	-418,14	410,16	29925,8

Tabla 5.2: Resultados de una simulación

arrayDatesAtLocation(self, val, years, N=3, savedata=False)

Esta función de la clase **geomag** permite obtener y guardar en un fichero los datos del campo magnético en una localización concreta para un conjunto de fechas. De manera análoga al apartado anterior se pueden guardar los resultados en un *.csv* para su posterior utilización.

plotMagneticField(self, val, year=2020, N=13, modelos=[1, 0, 0, 0, 1], timeRange=[1900, 2025], absolute_value=False)

Esta función de la clase **geomag** permite obtener y guardar en un fichero los datos del campo magnético en una localización concreta para un conjunto de fechas. Además de hacer una representación gráfica de los datos. Se pueden elegir los modelos que queremos representar, así como el orden de nuestro modelo.

5.3. VALIDACIÓN

En este apartado se va a validar nuestro software con herramientas online que propone la NNOA que se pueden consultar en [15]. Los datos simulados tanto con nuestro programa como con los softwares propuestos se muestran en las dos tablas siguientes 5.3 5.4. En la tabla 5.5 se muestra el error relativo entre las dos herramientas.

year	BN (nT)	BE (nT)	BD (nT)	D (deg)	H (nT)	I (deg)	F (nT)
2020.0	24433.62	-268.9	-453.81	-0.63	24435.1	-1.06	24439.31
2021.0	24438.97	-219.41	-441.07	-0.51	24439.96	-1.03	24443.94
2022.0	24444.33	-169.92	-428.32	-0.4	24444.92	-1.0	24448.67
2023.0	24449.68	-120.43	-415.58	-0.28	24449.98	-0.97	24453.51
2024.0	24455.04	-70.94	-402.83	-0.17	24455.14	-0.94	24458.46
2025.0	24460.39	-21.45	-390.08	-0.05	24460.4	-0.91	24463.51

Tabla 5.3: Valores obtenidos en el simulador

year	BN (nT)	BE (nT)	BD (nT)	D (deg)	H (nT)	I (deg)	F (nT)
2020.0	24433.7	-268.70	-453.81	-0.63	24435.2	-1.06	24439.4
2021.0	24439.	-219.40	-441.07	-0.51	24440.0	-1.03	24444.0
2022.0	24444.3	-170.00	-428.32	-0.4	24444.9	-1.0	24448.6
2023.0	24449.6	-120.80	-415.58	-0.28	24449.9	-0.97	24453.4
2024.0	24454.9	-71.40	-402.83	-0.17	24455.0	-0.94	24458.3
2025.0	24460.2	-22.10	-390.08	-0.05	24460.2	-0.91	24463.3

Tabla 5.4: Valores verificados según un software de la NNOA

year	BN (%)	BE (%)	BD (%)	D (%)	H (%)	I (%)	F (%)
2020.0	-3.27E-04	7.44E-02	-4.19E-02	0	-4.09E-04	0	-3.68E-04
2021.0	-1.23E-04	4.56E-03	-6.80E-03	0	-1.64E-04	0	-2.45E-04
2022.0	1.23E-04	-4.71E-02	2.80E-02	0	8.18E-05	0	2.86E-04
2023.0	3.27E-04	-3.06E-01	9.15E-02	0	3.27E-04	0	4.50E-04
2024.0	5.72E-04	-6.44E-01	1.32E-01	0	5.72E-04	0	6.54E-04
2025.0	7.77E-04	-2.94E+00	-5.64E-02	0	8.18E-04	0	8.58E-04

Tabla 5.5: Errores de los parámetros anteriores en %

Por otro lado, se va a probar para otros puntos geográficos y se validará el modelo como se hizo en el apartado anterior. A la vista de las tablas 5.6 y 5.7, se constata que los valores son prácticamente idénticos. El error entre los valores se puede ver expresado en forma de porcentaje en la tabla 5.8

R [km]	Lat [grados]	Long [grados]	Bx [nT]	By [nT]	Bz [nT]
7000	-90	-180	-9317.58	6368.71	-39066.41
7000	-80	-160	-2071.02	9247.5	-42504.81
7000	-70	-140	5882.87	9883.43	-40281.61
7000	-60	-120	11811.27	8931.31	-33323.17
7000	-50	-100	14989.92	6816.01	-23858.05
7000	-40	-80	15644.96	2912.86	-14869.86
7000	-30	-60	14515.5	-2193.24	-10416.76
7000	-20	-40	13529.73	-5025.05	-11225.52
7000	-10	-20	15226.69	-4483	-12967.79
7000	0	0	20448.15	-1859.71	-9890.32
7000	10	20	24950.14	452.21	365.73
7000	20	40	25672.06	1073.72	13761.19
7000	30	60	23636.58	911.18	25089.91
7000	40	80	20052.68	848.51	34481.94
7000	50	100	15148.77	-476.92	40998.33
7000	60	120	10756.88	-1855.13	43288.93
7000	70	140	7257.16	-1439.48	43837.72
7000	80	160	3335.43	-188.21	44125.24

Tabla 5.6: Valores obtenidos en el simulador

R [km]	Lat [grados]	Long [grados]	Bx [nT]	By [nT]	Bz [nT]
7000	-90	-180	-9317.6	6368.7	-39066.4
7000	-80	-160	-2071	9247.5	-42504.8
7000	-70	-140	5882.9	9883.4	-40281.6
7000	-60	-120	11811.3	8931.3	-33323.2
7000	-50	-100	14989.9	6816	-23858
7000	-40	-80	15645	2912.9	-14869.9
7000	-30	-60	14515.5	-2193.2	-10416.8
7000	-20	-40	13529.7	-5025.1	-11225.5
7000	-10	-20	15226.7	-4483	-12967.8
7000	0	0	20448.2	-1859.7	-9890.3
7000	10	20	24950.1	452.2	365.7
7000	20	40	25672.1	1073.7	13761.2
7000	30	60	23636.6	911.2	25089.9
7000	40	80	20052.7	848.5	34481.9
7000	50	100	15148.8	-476.9	40998.3
7000	60	120	10756.9	-1855.1	43288.9
7000	70	140	7257.2	-1439.5	43837.7
7000	80	160	3335.4	-188.2	44125.2

Tabla 5.7: Valores verificados según un software de la NNOA

R [km]	Lat [grados]	Long [grados]	Bx [%]	By [%]	Bz [%]
7000	-90	-180	2.15E-04	1.57E-04	2.56E-05
7000	-80	-160	9.66E-04	0.00E+00	2.35E-05
7000	-70	-140	5.10E-04	3.04E-04	2.48E-05
7000	-60	-120	2.54E-04	1.12E-04	9.00E-05
7000	-50	-100	1.33E-04	1.47E-04	2.10E-04
7000	-40	-80	2.56E-04	1.37E-03	2.69E-04
7000	-30	-60	0.00E+00	1.82E-03	3.84E-04
7000	-20	-40	2.22E-04	9.95E-04	1.78E-04
7000	-10	-20	6.57E-05	0.00E+00	7.71E-05
7000	0	0	2.45E-04	5.38E-04	2.02E-04
7000	10	20	1.60E-04	2.21E-03	8.20E-03
7000	20	40	1.56E-04	1.86E-03	7.27E-05
7000	30	60	8.46E-05	2.19E-03	3.99E-05
7000	40	80	9.97E-05	1.18E-03	1.16E-04
7000	50	100	1.98E-04	4.19E-03	7.32E-05
7000	60	120	1.86E-04	1.62E-03	6.93E-05
7000	70	140	5.51E-04	1.39E-03	4.56E-05
7000	80	160	8.99E-04	5.31E-03	9.07E-05

Tabla 5.8: Errores de los parámetros anteriores en %

A la vista de los resultados anteriores, se observa como los errores entre el modelo desarrollado y el modelo oficial son muy bajos, teniendo en cuenta que se trabaja en nT, y siendo los errores del orden de 10^{-4}, se observa como estos son ínfimos y el modelo es perfectamente funcional.

Capítulo 6

Implementación del modelo EGM96

El potencial terrestre también es un valor de interés para las aplicaciones espaciales, es por ello que este capítulo se dedica íntegramente al modelado del mismo. Existen diferentes modelos para su cálculo, en este proyecto el modelo que se va a desarrollar es el EGM96.

6.1. FUNDAMENTOS MATEMÁTICOS

Para la obtención del potencial terrestre y su posterior uso para el cálculo de las perturbaciones orbitales, se sigue el siguiente desarrollo matemático.

El modelado del campo gravitatorio terrestre se realiza de manera análoga al modelado del campo magnético.

$$U(r,\theta,\varphi) = \frac{\mu}{r}\left(1 + \sum_{n=1}^{\infty}\left(\frac{a}{r}\right)^n \sum_{m=0}^{\infty} Y_n^m(\sin(\theta))(C_n^m\cos(m\varphi) + S_n^m\sin(m\varphi))\right). \quad (6.1)$$

Es necesario normalizar los polinomios asociados de Legendre $P_n^m(x)$, para ello se utiliza la siguiente ecuación:

$$Y_n^m = \sqrt{\frac{(2-\delta_{0m})(2n+1)(n-m)!}{(n+m)!}}P_n^m(x), \quad (6.2)$$

con

$$\delta_{0m} = \begin{cases} 1 & \text{si} \quad m = 0 \\ 0 & \text{otro caso} \end{cases} \quad (6.3)$$

Asumiendo $C_1^0 = C_1^1 = S_0^1 = S_1^1 = 0$ y para $m = 0$ se tiene $\sin(m\varphi) = 0$ se puede reescribir la expresión del potencial como la ecuación siguiente:

$$U(r,\theta,\varphi) = \frac{\mu}{r}\left(1 + \sum_{n=2}^{\infty}\left(\frac{a}{r}\right)^n J_n Y_n^0(\sin(\theta))\right) + \frac{\mu}{r}\left(\sum_{n=2}^{\infty}\sum_{m=1}^{\infty} Y_n^m(\sin(\theta))(C_n^m\cos(m\varphi) + S_n^m\sin(m\varphi))\right). \quad (6.4)$$

Gravedad

Para el cálculo de la gravedad se pueden utilizar las siguientes expresiones:

$$g_r = -\frac{\mu}{r^2} - \sum_{n=1}^{\infty} \frac{\mu(n+1)a^n}{r^{n+2}} \sum_{m=0}^{\infty} Y_n^m(\sin(\theta))(C_n^m \cos(m\varphi) + S_n^m \sin(m\varphi)), \tag{6.5}$$

$$g_\theta = \sum_{n=1}^{\infty} \frac{\mu a^n}{r^{n+2}} \sum_{m=0}^{\infty} \frac{d}{d\theta} Y_n^m(\sin(\theta)) \cos\theta (C_n^m \cos(m\varphi) + S_n^m \sin(m\varphi)), \tag{6.6}$$

$$g_\phi = \sum_{n=1}^{\infty} \frac{\mu a^n}{r^{n+2} \sin\theta} \sum_{m=0}^{\infty} Y_n^m m(\sin(\theta))(S_n^m \cos(m\varphi) - C_n^m \sin(m\varphi)). \tag{6.7}$$

6.1.1. ONDULACIÓN

La ondulación del modelo EGM96 hace referencia a la altura con respecto al nivel del elipsoide de referencia. Para calcular la ondulación es necesario conocer primeramente el potencial V del elipsoide de referencia que se calcula con la siguiente fórmula.

$$V(r,\theta,\varphi) = -\frac{\mu}{r}\left(1 + \sum_{n\ even}^{\infty} \left(\frac{a}{r}\right)^n J_n Y_n^0(\sin(\theta))\right). \tag{6.8}$$

Para su implementación numérica, se necesita obtener primeramente el potencial perturbador. La idea es calcular la diferencia entre el modelo del potencial terrestre que tiene en cuenta que la tierra no es completamente esférica con su modelo ideal de potencial de tierra esférica.

$$T(r,\theta,\varphi) = U(r,\theta,\varphi) - V(r,\theta,\varphi). \tag{6.9}$$

A continuación, se utiliza la llamada fórmula de Nerts para calcular la ondulación.

$$N = \frac{T}{\gamma} \tag{6.10}$$

En este caso, γ es la magnitud de la gravedad normal en la superficie del elipsoide. Asumiendo $r = a$, se obtiene la siguiente expresión simplificada de la ondulación terrestre.

$$N(a,\theta,\varphi) = \frac{a^2}{\mu} T(r,\theta,\varphi) \tag{6.11}$$

Se puede seguir el desarrollo matemático en [20], [21], [10] y los datos de los coeficientes S_n^m y C_n^m se pueden obtener en [22].

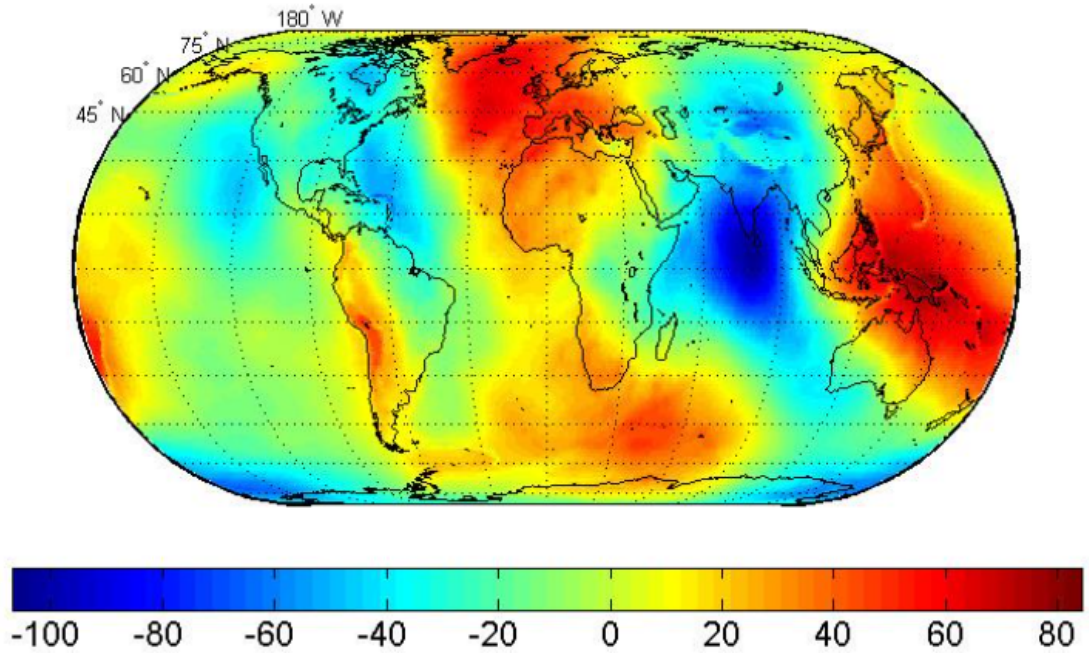


Figura 6.1: Representación de la ondulación en metros [23]

En figura 6.1 se muestra la elevación en metros con respecto al elipsoide de referencia del modelo EGM96.

6.2. DESARROLLO NUMÉRICO

En primer lugar, lo que se va a hacer, es obtener los datos de los parámetros que permitirán ajustar el grado de precisión del modelo. Estos datos de los coeficientes S_n^m y C_n^m se pueden obtener en [22]. En la tabla 6.1 se pueden ver los primeros coeficientes S_n^m y C_n^m, así como sus desviaciones estándar.

m	n	S	C	s (sd)	c (sd)
2	0	-4.84E-04	0.00E+00	3.56E-11	0.00E+00
2	1	-1.87E-10	1.20E-09	1.00E-30	1.00E-30
2	2	2.44E-06	-1.40E-06	5.37E-11	5.44E-11
3	0	9.57E-07	0.00E+00	1.81E-11	0.00E+00
3	1	2.03E-06	2.49E-07	1.40E-10	1.36E-10
3	2	9.05E-07	-6.19E-07	1.10E-10	1.12E-10
3	3	7.21E-07	1.41E-06	9.52E-11	9.33E-11
4	0	5.40E-07	0.00E+00	1.04E-10	0.00E+00
4	1	-5.36E-07	-4.73E-07	8.57E-11	8.24E-11
4	2	3.51E-07	6.63E-07	1.60E-10	1.64E-10
4	3	9.91E-07	-2.01E-07	8.47E-11	8.27E-11
4	4	-1.89E-07	3.09E-07	8.73E-11	8.79E-11

Tabla 6.1: Valores de S_n^m y C_n^m

El modelo desarrollado tiene la capacidad de usarse con el grado de precisión que se desee, siendo en valor máximo 360, coincidiendo con el valor máximo en cuando a orden y grado de los coeficientes S_n^m y C_n^m.

6.2.1. HERRAMIENTAS

En este apartado se presentan algunas de las funcionalidades de la clase **geopot**.

Obtención del campo gravitatorio y la gravedad

En este apartado se puede calcular el campo potencial terrestre tanto en un punto como en un *array* de puntos. Se pueden apreciar los resultados en la tabla 6.2.

r km	Lat º	Long º	Potencial m^2/s^2	g1 m/s^2	g2 m/s^2	g3 m/s^2	g m/s^2
7000	-90	-180	56891685.42	-8.11274	0.00000	0.00000	8.11274
7000	-80	-160	56893879.30	-7.50913	-2.73310	-1.40522	8.11366
7000	-70	-140	56900509.49	-5.84450	-4.90409	-2.76936	8.11650
7000	-60	-120	56910777.94	-3.51887	-6.09480	-4.05222	8.12093
7000	-50	-100	56923395.45	-1.08223	-6.13740	-5.21524	8.12635
7000	-40	-80	56936815.11	0.90910	-5.15596	-6.22261	8.13211
7000	-30	-60	56949381.10	2.03850	-3.53075	-7.04252	8.13749
7000	-20	-40	56959562.56	2.13829	-1.79418	-7.64840	8.14183
7000	-10	-20	56966231.82	1.33246	-0.48510	-8.02028	8.14467
7000	0	0	56968638.15	-	-	-	0.00000
7000	10	20	56966318.42	-1.33231	-0.48536	-8.02036	8.14474
7000	20	40	56959423.99	-2.13796	-1.79449	-7.64837	8.14178
7000	30	60	56948965.28	-2.03829	-3.53076	-7.04232	8.13727
7000	40	80	56936408.93	-0.90913	-5.15582	-6.22241	8.13187
7000	50	100	56923221.35	1.08211	-6.13733	-5.21515	8.12622
7000	60	120	56910858.37	3.51880	-6.09483	-4.05227	8.12095
7000	70	140	56900766.60	5.84456	-4.90419	-2.76948	8.11665
7000	80	160	56894176.17	7.50929	-2.73316	-1.40530	8.11385
7000	90	180	56891919.86	8.11288	0.00000	0.00000	8.11288

Tabla 6.2: Ejemplo de cálculo del Potencial y la gravedad para distintos puntos

Representaciones gráficas

Se pueden ver dos gráficas obtenidas 6.2.

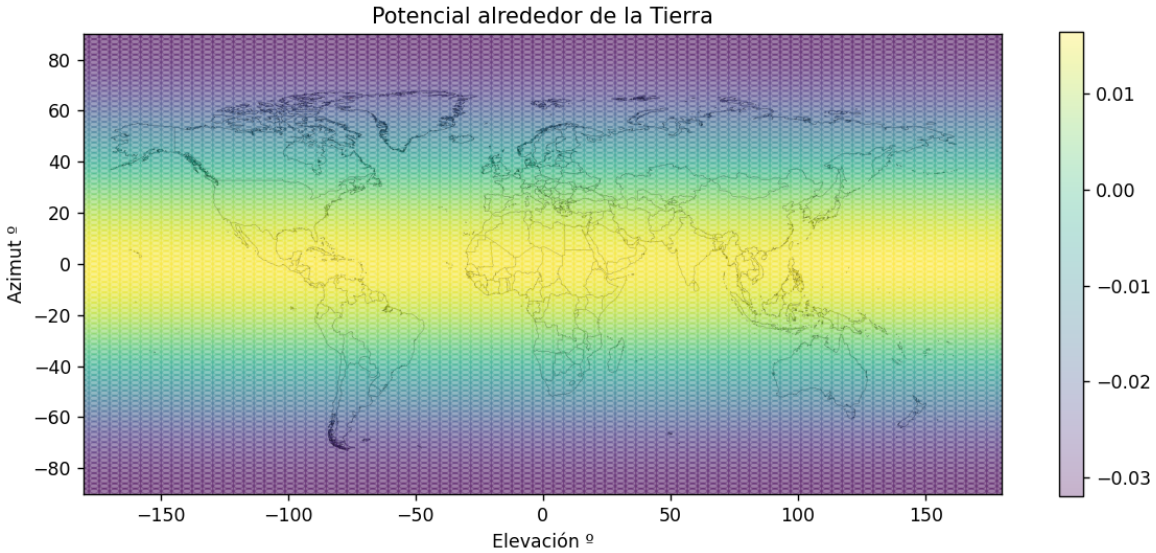


Figura 6.2: Potencial terrestre (Diferencia entre el valor de tierra esférica y el del modelo en J/kg)

Las gráficas están tanto en 2D como en 3D. En 3D se tiene la tabla 6.3.

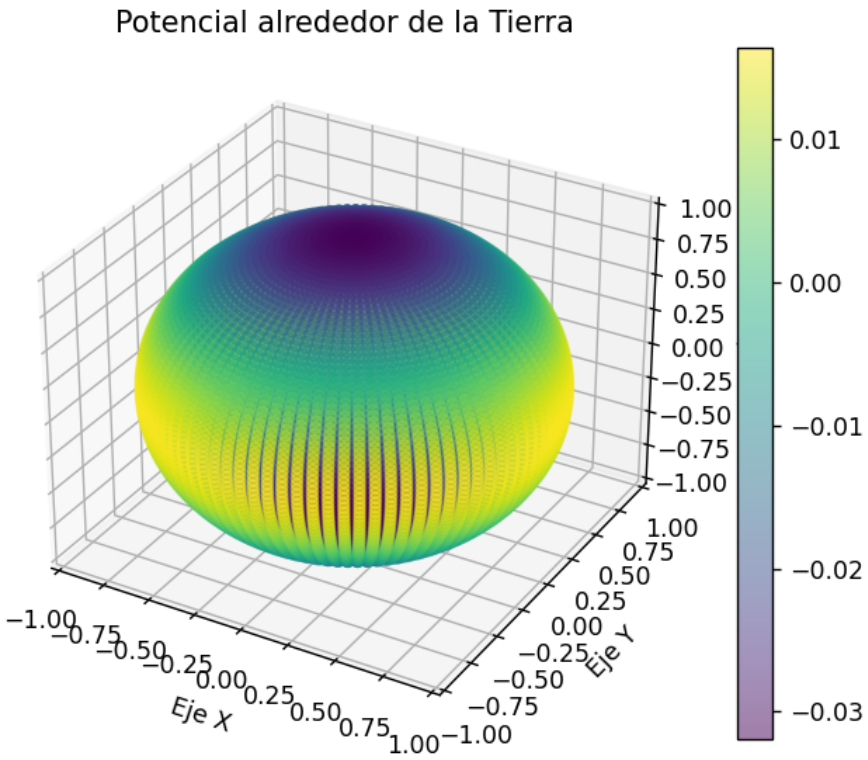


Figura 6.3: Potencial terrestre 3D (Diferencia entre el valor de tierra esférica y el del modelo en J/kg)

6.3. VALIDACIÓN

En este apartado se pretende validar el modelo EGM96 desarrollado, para ello se va a utilizar como referencia el modelo que se encuentra en [20] donde se utilizará su modelo para validar el nuestro.

Para ello se va a realizar una tabla comparativa donde se muestran los errores en porcentaje para diferentes puntos alrededor de la Tierra. Los resultados obtenido se encuentran en la siguiente tabla 6.3.

R [km]	Lat [grados]	Long [grados]	Potencial [%]	g1 [%]	g2 [%]	g3 [%]
7000	-90	-180	2.99E-05	-	-	-
7000	-80	-160	1.03E-04	7.09E-04	9.33E-04	1.74E-03
7000	-70	-140	5.55E-05	1.32E-04	6.34E-04	2.05E-03
7000	-60	-120	7.67E-05	6.72E-04	5.80E-04	2.57E-04
7000	-50	-100	4.41E-05	7.96E-04	4.77E-05	5.81E-04
7000	-40	-80	5.27E-06	5.52E-03	5.36E-04	2.29E-04
7000	-30	-60	3.16E-05	4.53E-03	9.12E-04	3.86E-04
7000	-20	-40	8.41E-05	2.35E-03	3.81E-03	6.11E-04
7000	-10	-20	1.16E-04	6.22E-03	5.78E-02	5.75E-04
7000	0	0	7.82E-05	-	-	-
7000	10	20	1.16E-04	6.79E-04	8.17E-03	7.35E-04
7000	20	40	1.71E-04	4.79E-03	6.19E-03	1.43E-03
7000	30	60	2.72E-04	1.05E-03	1.38E-03	2.30E-03
7000	40	80	2.36E-05	7.55E-03	1.76E-03	7.55E-04
7000	50	100	1.57E-04	9.28E-04	5.72E-04	1.58E-03
7000	60	120	1.32E-05	1.14E-03	4.49E-04	9.26E-04
7000	70	140	3.93E-05	3.77E-05	4.20E-04	8.78E-05
7000	80	160	3.41E-06	7.39E-05	7.40E-05	1.72E-04
7000	90	180	1.53E-05	-	-	-

Tabla 6.3: Validación EGM96

En los polos y en el ecuador encontramos una serie de singularidades, a la vista de la expresión 6.7 donde se observa un $\sin\theta$ dividiendo la expresión.

Capítulo 7

Propagador de órbitas con Python

El fundamento de un propagador de órbitas no es más que resolver la ecuación diferencial que describe el movimiento. La ecuación que se muestra a continuación es la base sobre la cual se irán desarrollando modelos más complejos que incluyan las diferentes perturbaciones.

En los siguientes apartados, se irán desarrollando tanto matemáticamente como su implementación numérica de las partes fundamentales que conforman un simulador de mecánica espacial. También se incluirá en cada sección la validación de las diferentes partes desarrolladas.

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{7.1}$$

7.1. PROPAGADOR DE ÓRBITAS

Esta clase que se ha desarrollado siguiendo la formulación matemática del libro "Fundamental of astrodynamics"[21] permite realizar la propagación de la órbita sabiendo el vector posición inicial r_0 y el vector velocidad inicial v_0. En el código se puede encontrar como *Orbit()*.

Para la resolución del problema de Kepler se utilizará la ecuación universal de Kepler, la cual se resolverá de manera iterativa para obtener la anomalía universal [21]

$$\sqrt{\mu}\Delta t = \frac{r_0 v_{r0}}{\sqrt{\mu}}\chi^2 C(\alpha\chi^2) + (1-\alpha r_0)\chi^3 S(\alpha\chi^2) + r_0\chi. \tag{7.2}$$

donde $\alpha = \frac{1}{a}$ y las funciones S y C

$$S(z) = \sum_{k=0}^{\infty}(-1)^k \frac{z^k}{(2k+3)!} \longleftrightarrow C(z) = \sum_{k=0}^{\infty}(-1)^k \frac{z^k}{(2k+2)!}. \tag{7.3}$$

Tras aplicar el método de Newton, se obtiene una de las soluciones a la ecuación para una tolerancia y un número máximo de iteraciones.

Partiendo de la anomalía universal, se pueden obtener los coeficientes de Lagrange y a partir de estos obtener el vector velocidad y posición tras un intervalo de tiempo:

$$f = 1 - \frac{\chi^2}{r_0}C(\alpha\chi^2), \tag{7.4}$$

$$g = \Delta t - \frac{1}{\sqrt{\mu}}\chi^3 S(\alpha\chi^2), \tag{7.5}$$

$$\dot{f} = \frac{\sqrt{\mu}}{r_0 r}\left[\alpha\chi^3 S(\alpha\chi^2) - \chi\right], \tag{7.6}$$

$$\dot{g} = 1 - \frac{\chi^2}{r}C(\alpha\chi^2), \tag{7.7}$$

y finalmente:

$$\mathbf{r} = f\mathbf{r}_0 + g\mathbf{v}_0, \tag{7.8}$$

$$\mathbf{v} = \dot{f}\mathbf{r}_0 + \dot{g}\mathbf{v}_0. \tag{7.9}$$

Se puede ver parte del código en A.1

El desarrollo completo del algoritmo numérico se puede seguir en [21] o [9]. Se trata de un método simple de propagación de órbitas, en el cual no se tienen en cuenta las perturbaciones que vistas en los apartados anteriores. Se puede ver un ejemplo en la figura 7.1 donde se han simulado las órbitas tradicionales de un GPS, circular, una órbita Molnya, característica por su alta excentricidad, y una órbita geoestacionaria.

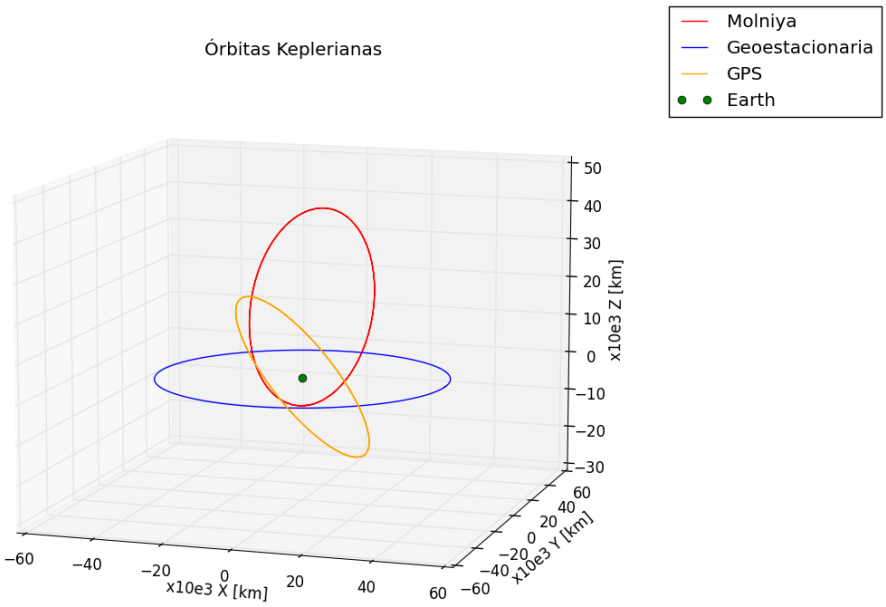


Figura 7.1: Órbitas de algunos satélites representativos

7.2. PERTURBACIONES

Como se vio en apartados anteriores, las perturbaciones son uno de los aspectos más importantes a tener en cuenta a la hora de obtener modelos realista como se puede observar en 4.3.

En este apartado, se estudiarán, mediante el desarrollo de una clase nueva, los principales propagadores que tienen en cuenta los efectos de las perturbaciones.

7.2.1. FORMULACIÓN DE ENCKE

El método de integración de Encke se basa en la derivación instantánea del problema de 2 cuerpos ($\overline{r}(t)$ $\overline{v}(t)$) proyectada hacia delante en el tiempo partiendo de ($r(t_0)$ $v(t_0)$). La solución instantánea se conoce como órbita oscultadora como se representa en la figura 7.2.

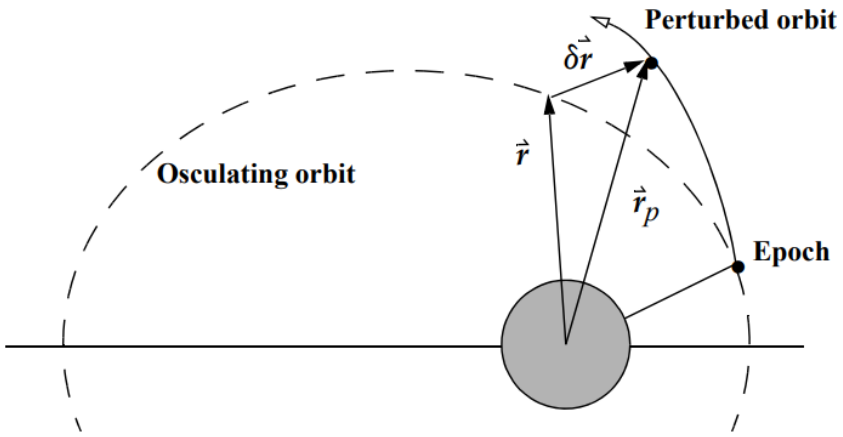


Figura 7.2: Funcionamiento de la formulación de Encke [21]

$$\frac{d^2\overline{\mathbf{r}}}{dt^2} + \frac{\mu}{\overline{r}^3}\overline{\mathbf{r}} = 0, \tag{7.10}$$

de esta manera se tiene:

$$\begin{aligned} \overline{\mathbf{r}}(t_0) &= \mathbf{r}(t_0), \\ \overline{\mathbf{v}}(t_0) &= \mathbf{v}(t_0). \end{aligned} \tag{7.11}$$

En un instante de tiempo posterior $t = t_0 + \Delta t$ se tiene:

$$\begin{aligned} \mathbf{r}(t) &= \overline{\mathbf{r}}(t) + \alpha(t), \\ \mathbf{v}(t) &= \overline{\mathbf{v}}(t) + \beta(t), \end{aligned} \tag{7.12}$$

$$\frac{d^2\alpha}{dt^2} + \frac{\mu}{\overline{r}^3}\alpha = \frac{\mu}{\overline{r}^3}\left(1 - \frac{\overline{r}^3}{r^3}\right)\mathbf{r} + \mathbf{a}_d. \tag{7.13}$$

Se tienen las siguientes condiciones iniciales:

$$\begin{aligned} \alpha(t_0) &= 0, \\ \dot{\alpha}(t_0) &= \beta(t_0) = 0. \end{aligned} \tag{7.14}$$

La ecuación 7.10 tiene problemas de estabilidad numérica, es por ello que para evitar este problema se reescribe el problema como:

$$\left(1 - \frac{\overline{r}^3}{r^3}\right) = -x\frac{3 + 3x + x^2}{1 + (1+x)^{3/2}} \Longleftrightarrow x = \frac{\alpha(\alpha - 2\mathbf{r})}{r^2}. \tag{7.15}$$

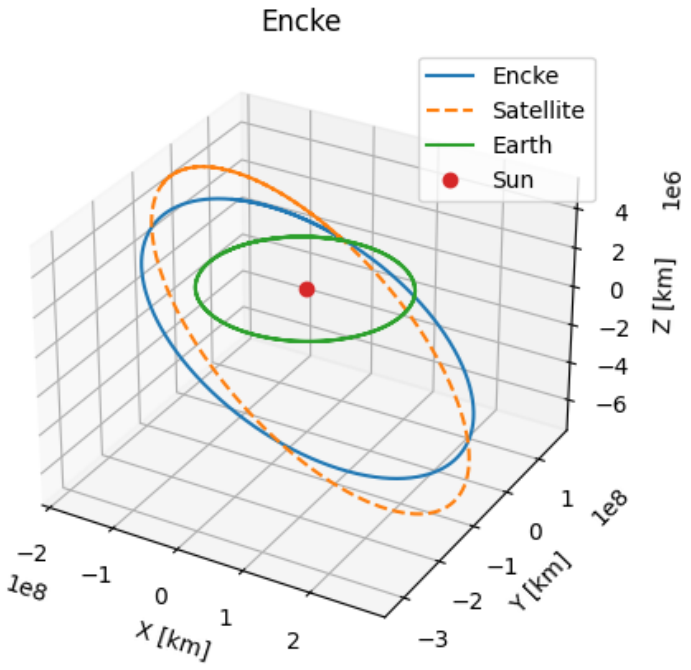


Figura 7.3: Perturbación debida a un 3º cuerpo - Encke

Para más detalles de la formulación matemática se puede consultar el [10]. Por otro lado, para la programación que se van a seguir el **Algoritmo 62** del *Vallado* [21]. Este algoritmo no es más que la implementación de la formulación matemática que se describió en la parte superior.

También se puede consultar [24] para seguir el desarrollo de la formulación de Encke así como su implementación numérica.

7.2.2. COWELL

La formulación de Cowell para la propagación de órbitas es muy sencilla y se basa en la integración directa de las siguientes ecuaciones diferenciales:

$$\overline{X} = \begin{bmatrix} \overrightarrow{r} \\ \overrightarrow{v} \end{bmatrix}, \qquad \dot{\overline{X}} \begin{bmatrix} \overrightarrow{v} \\ -\frac{\mu \overrightarrow{r}}{r^3} + \overrightarrow{a}_p \end{bmatrix}. \tag{7.16}$$

Siguiendo los métodos numéricos que se pueden consultar en [25], se puede realizar la integración a lo largo de la órbita.

Aquí se muestra como ejemplo el método de Runge-Kutta para su resolución:

$$\begin{aligned} k_1 &= hf(x(t), t), \\ k_2 &= hf\left(x(t) + \frac{k_1}{2}, t + \frac{h}{2}\right), \\ k_3 &= hf\left(x(t) + \frac{k_2}{2}, t + \frac{h}{2}\right), \\ k_4 &= hf(x(t) + k_3, t + h), \\ x(t+h) &= x(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{7.17}$$

Capítulo 8

Cambios de coordenadas

8.1. CLASE RV2ORB: PASO DE VECTORES R(T) Y V(T) A PARÁMETROS ORBITALES

El objetivo de esta sección es el cálculo de los parámetros orbitales teniendo en cuenta que numéricamente se trabajan con vectores posición y velocidad.

Partiendo de "Spacecraft Dynamics and Control"[14] se pueden encontrar las relaciones buscadas, las cuales se implementan en la clase de python correspondiente.

En primer lugar, el momento angular específico se define como:

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}. \tag{8.1}$$

El resto de parámetros son los parámetros definidos en la sección 4.1

$$i = \arccos\left(\frac{h_z}{\|h\|}\right). \tag{8.2}$$

$$\mathbf{N} = e_z \times \mathbf{h} \Longrightarrow \Omega = \arccos\left(\frac{N_x}{\|N\|}\right). \tag{8.3}$$

$$\mathbf{e} = \frac{1}{\mu}\left[\mathbf{v} \times \mathbf{h} - \mu\frac{\mathbf{r}}{\|r\|}\right]. \tag{8.4}$$

$$\omega = \arccos\left(\frac{\mathbf{Ne}}{\|N\|\|e\|}\right). \tag{8.5}$$

$$\theta = \arccos\left(\frac{\mathbf{er}}{\|e\|\|r\|}\right). \tag{8.6}$$

El modelo es validado utilizando un ejemplo de la referencia [21]

	x	**y**	**z**
R (km)	-4975.14	3451.24	3869.89
V (km/s)	3.69	-1.92	-6.11

Tabla 8.1: Parámetros iniciales

Obteniendo como resultados:

- Semieje mayor: 7200.470 km
- Excentricidad: 0.0081
- Inclinación: 98.599 deg

- Nodo ascendente: 319.7043 deg
- Argumento del periastro: 70.879 deg
- Anomalía verdadera: 76.119 deg

8.2. CLASE ORB2RV: PASO DE PARÁMETROS ORBITALES A VECTORES R(T) Y V(T)

Otra de las situaciones más habituales es la de tener los parámetros habituales y tener que pasar a los vectores de posición y velocidad. En caso de tener una órbita dada por los parámetros orbitales habituales, se puede obtener su equivalente en vectores posición y velocidad de la siguiente manera:

$$\mathbf{r}_x = \frac{\|h\|^2}{\mu} \frac{1}{1 + e\cos\theta} \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}. \tag{8.7}$$

$$\mathbf{v}_x = \frac{\mu}{\|h\|} \begin{pmatrix} -\sin\theta \\ e + \cos\theta \\ 0 \end{pmatrix}. \tag{8.8}$$

Estos son los vectores posición y velocidad en el sistema de referencia perifocal. Para cambiarlo al sistema de referencia ECI se necesitan realizar tres cambios de referencia como se muestra a continuación:

$$R_3(\Omega) = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8.9}$$

$$R_1(i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}. \tag{8.10}$$

$$R_3(\omega) = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8.11}$$

La matriz de cambio de ejes de referencia es:

$$Q = R_3(\omega)^T R_1(i)^T R_3(\Omega)^T. \tag{8.12}$$

Se va a utilizar un ejemplo del libro [9] para la validación de esta clase, así como la clase del apartado anterior.

Paramétros	Valor	Unidad
h	80000	Km^2/s
e	1.4	-
RA	40	rad
incl	30	rad
w	60	rad
TA	30	rad

Tabla 8.2: Valores de entrada para la validación de la clase

Los resultados obtenidos son los siguientes:

	x	**y**	**z**
R (km)	-4039.9	4814.56	3628.62
V (km/s)	-10.386	-4.77192	1.74388

Tabla 8.3: Valores de salida para la validación de la clase

Si los se comparan con los del libro de referencia vemos que son idénticos.
State vectors:
r (km) = [-4039.9 4814.56 3628.62]
v (km/s) = [-10.386 -4.77192 1.74388]

8.3. LIBRERÍA COORDINATES

En este apartado se van a ver algunas de las funciones más destacadas de la librería Coordinates que incluye funciones de utilidad para el cambio de coordenadas, así como una clase para pasar de ECI a ECEF y viceversa. Se muestra un ejemplo donde se ha pasado de sistema de coordenadas espaciales, a un sistema coordenado conde la Tierra se encuentra girando. Se puede apreciar las diferencias entre las figuras 7.1 y 8.2.

Esta clase se encarga de realizar un cambio de coordenadas, destaca el sistema de coordenadas cuasi-inercial centrado en un cuerpo (sea el Sol para el análisis de misiones interplanetarias, sea la Tierra) fijado en estrellas muy lejanas que permanecen cuasi-invariantes para el periodo de tiempo de las misiones típicas. Por otro lado, el sistema de coordenadas de Tierra rotante es especialmente útil para obtener las trazas de las órbitas en tierra. Para ello hay que tener en cuenta la rotación terrestre $\omega_T = 7{,}2921 \times 10^{-5} \frac{rad}{s}$ y realizar un cambio de coordenadas mediante las matrices de rotación implementadas en esta clase.

$$\overline{X}_{ECEF} = R_{ECI2ECEF} \times \overline{X}_{ECI} \tag{8.13}$$

Se puede observar mediante el cambio de ejes en A.3 y visualmente mediante un ejemplo se tienen las figuras 8.2 y 8.1.

Hay que remarcar que $R_{ECI2ECEF}$ no es más que una rotación en el eje z, y pese a que para efectos académicos pudiera ser suficientemente preciso, se mostrará más adelante como calcular una nueva $R_{ECI2ECEF}$ donde se tengan en cuenta los diversos efectos que entran en juego, como la inclinación de la tierra, el constante cambio de los polos, la nutación, el efecto del tiempo sideral, etc.

8.3.1. FUNCIÓN PLOT_GROUND_TRACK

Utilizando como inputs los valores de longitud y latitud, esta función mediante la librería *folium* representa las coordenadas en un mapa que posteriormente se guarda en formato *.htlm*.

Se presenta un ejemplo en la figura 8.1.

Figura 8.1: Trazas sobre un mapa

8.3.2. FUNCIÓN ECEF2LATLON

En este apartado se sigue la formulación propuesta por el libro de referencia [21], donde el ejemplo 3-3 permite verificar que el código funciona correctamente. Para ello utilizamos un vector posición en coordenadas ECEF

$$R_{ECEF} == 6524{,}834\hat{I} + 6862{,}875\hat{J} + 6448{,}296\hat{J}km$$

El resultado que muestra el libro, es el mismo que el que se ha obtenido tras la programación:

- Longitud geodésica: 34.352º
- Longitud geocéntrica: 34.173º
- Altitud: 5085.219 Km
- Latitud: 46.446 º

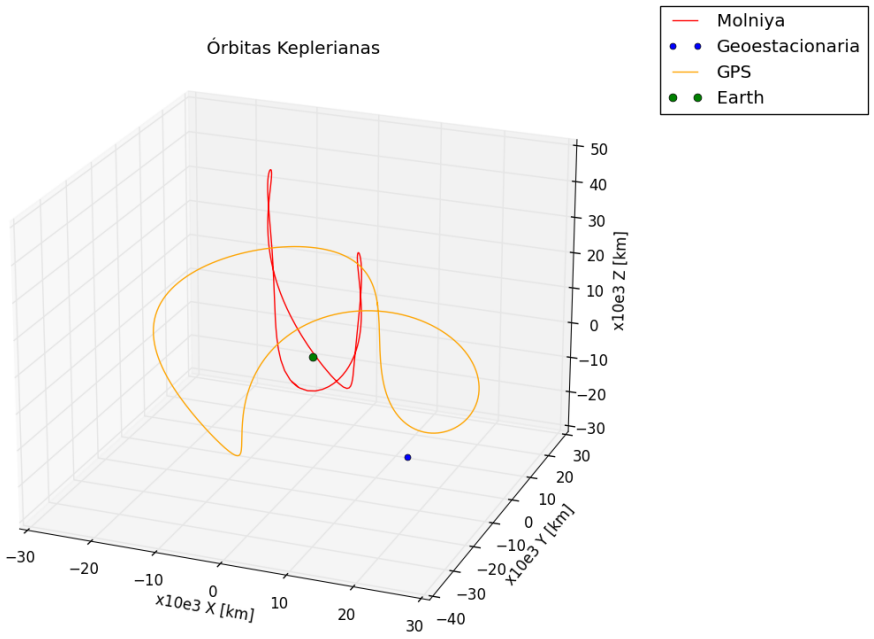


Figura 8.2: Órbitas de algunos satélites representativos en sistema de referencia de Tierra fija

8.3.3. CAMBIOS DE COORDENADAS ENTRE ECI Y ECEF

En esta sección se estudia como realizar el cálculo de la posición y la velocidad de un satélite en el sistema de referencia terrestre fijo (ITRF) a partir de sus datos iniciales en el sistema de referencia inercial (GCRF) y viceversa. Como aproximación, se puede considerar que hay una única rotación a lo largo del eje z.

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.14}$$

Las dimensiones reales del problema del cambio de coordenadas celestes a un sistema de coordenadas terrestre, hay que tener en cuenta diferentes factores como:

- Nutación
- Movimiento de los polos
- Precesión
- Tiempo sideral

Para ver de manera más gráfica este cambio de coordenadas, se necesita obtener las matrices de cambio de coordenadas debidas a los siguientes fenómenos. Para ello se va a seguir la bibliografía de referencia [21] así como este código que será adaptado a nuestro problema [26] .

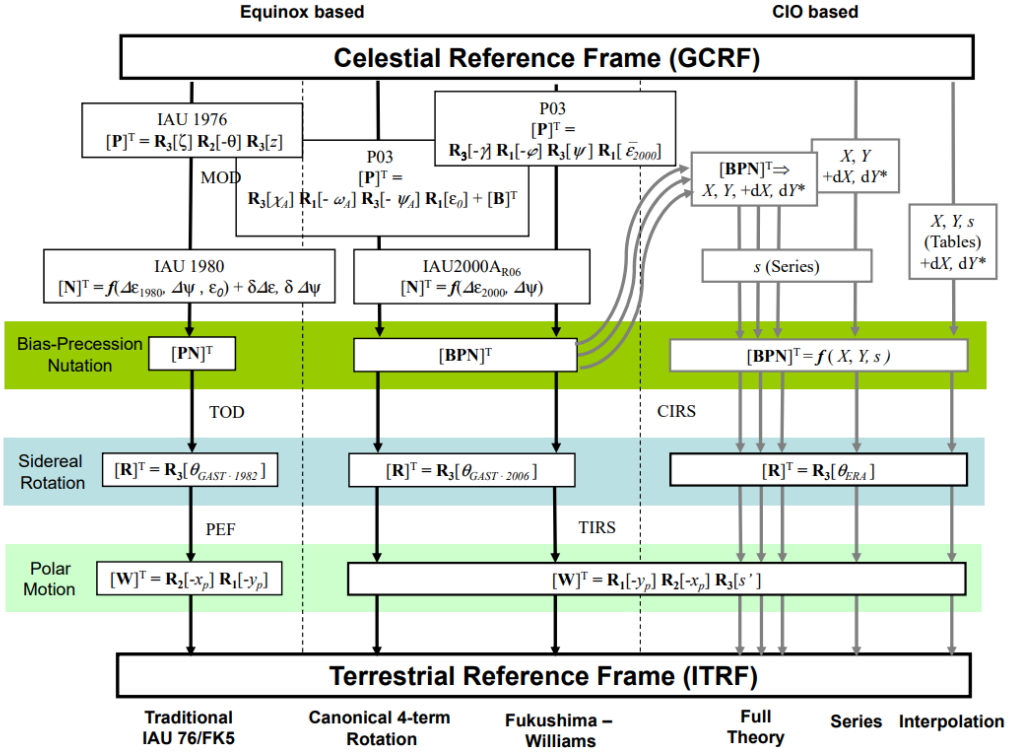


Figura 8.3: Transformación de coordenadas terrestres a celestes [21]

En este caso, se ha creado una clase que permite realizar los cálculos de manera precisa (IAU 76/ FK5), para ello se necesitan unos parámetros iniciales que se pueden consultar en la tabla 8.4.

Parámetro	Descripción	Valor ejemplo	Unidades
ttt	Siglos julianos de tiempo terrestre	0.0426	
jdut1	Fecha juliana de UT1	2.4531e+06	
lod	Duración del día	0.0016	seg
xp	Coeficiente de movimiento polar	-6.8205e-07	arcsec
yp	Coeficiente de movimiento polar	1.6159e-06	arcsec
eqterms	un valor booleano que indica si se usan los términos extra para la nutación (0 o 2)	2	0 ó 2
ddpsi	Corrección a delta psi	-2.5305e-07	arcsec
ddeps	Corrección a delta eps	-1.8787e-08	arcsec

Tabla 8.4: Valores de entrada la de clase ECI2ECEF

Se ha desarrollado como una clase que tiene diferentes métodos que calculan las matrices de rotación para cada caso. Hay métodos de la clase **ECI2ECAF** como **nutation**, **precess**, **sideal**, **polarm**, que permiten obtener las matrices nutación $N(t)$, precesión $P(t)$, rotación sideral $R(t)$ y La matriz de movimiento de los polos $W(t)$.

$$\vec{r}_{GRCF} = [P(t)][N(t)][R(t)][W(t)]\vec{r}_{ITRF} \tag{8.15}$$

Se han utilizado las siguientes aproximaciones para cada modelo de fenómeno físico:

- **nutation** [N(t)]: IAU 1980
- **precess** [P(t)]: IAU 1976
- **sideal** [R(t)]: IAU 1976
- **polarm** [W(t)]: IAU 1976

Se va a validar el código desarrollado con el ejemplo 3-15 del libro [21]. Para ello se utiliza para inicializar la clase los valores de la columna *Valor ejemplo* de la tabla 8.4. Además, se necesita introducir los vectores posición, velocidad y en el caso de que lo hubiera, el vector aceleración.

Los resultados obtenidos se muestran en la siguiente tabla 8.5 donde se compararan con los valores obtenidos en el ejemplo que se utiliza como validación.

	[km]			[km/s]		
Sistema de coordenadas	R1	R2	R3	V1	V2	V3
ECEF (ITRF)	-1033.479	7901.295	6380.357	-3.226	-2.872	5.532
ECI (GCRF) IAU-76/FK5	5102.498	6123.008	6378.149	-4.743	0.790	5.534
			Validación			
ECI (GCRF)	5102.509	6123.011	6378.137	-4.743	0.791	5.534
Error [%]	-2,06E-04	-6,21E-05	1,89E-04	6,32E-08	3,79E-07	5,42E-08

Tabla 8.5: Validación ECI2ECEF

De manera análoga, para validar la función que permite pasar de coordenadas ECI a ECEF, se introducen los valores de salida que se han obtenido, y se comprueba como se obtienen los valores de entrada.

8.3.4. FK4

Esta aproximación se encuentra cada vez más en desuso, pero a día de hoy continúa utilizándose en algún programa de simulación, es por eso que se ha decidido incluir en este apartado. Basándose en la página 235 del libro de referencia [21], se puede escribir la matriz de cambio de coordenadas como:

$$fk4m = \begin{bmatrix} 0{,}999925678612394 & -0{,}011181874556714 & -0{,}004858284812600 \\ 0{,}011181874524964 & 0{,}999937480517880 & -0{,}000027169816135 \\ 0{,}004858284884778 & -0{,}000027156932874 & 0{,}999988198095508 \end{bmatrix}, \quad (8.16)$$

donde el cambio de coordenadas quedaría como:

$$\vec{X}_{J2000} = [fk4m]\vec{X}_{B1950} \quad (8.17)$$

Hay que tener en cuenta que la conversión entre los sistemas FK4 y IAU-76/FK5 introduce dos diferencias importantes, UT1 tiene una tasa secular actualizada en IAU-76/FK5, y sus orígenes no coinciden. [21]

Capítulo 9

Determinación preliminar de órbitas

9.1. CLASE UTILITIES

Esta clase contiene una serie de funciones generales, entre las que se encuentran funciones para la determinación preliminar de órbitas que se resumen en las siguientes:

- Observaciones de 3 vectores posición para obtener el vector velocidad de la observación intermedia. Método de Gibbs.
- Observaciones de 3 vectores posición y sus respectivos. tiempos de observación para obtener el vector velocidad de la observación intermedia. Método de Herrick-Gibbs.
- Observaciones del rango, azimut, elevación, tiempo sideral local, latitud y altura, con sus ratios.
- Observaciones del vector posición geocéntrico y desde la posición del observador, así como los tiempos de medida. Método de Gauss con mejora iterativa.
- Observaciones de 2 vectores posición, se obtiene el vector velocidad. Se conoce como el problema de Lambert. En este caso, se ha desarrollado una clase aparte, donde se desarrollan varias versiones de este problema. Se explica en la sección 10.1 que corresponde a dicha clase.

9.1.1. FUNCIÓN FINDTOF

Esta función es capaz de calcular el tiempo de vuelo conocidos los vectores inicial y final. Se ha utilizado el **Algoritmo 11** del libro [21]

9.1.2. FUNCIÓN GIBBS

$$\overrightarrow{Z}_{12} = \overrightarrow{r}_1 \times \overrightarrow{r}_2, \qquad \overrightarrow{Z}_{23} = \overrightarrow{r}_2 \times \overrightarrow{r}_3, \qquad \overrightarrow{Z}_{31} = \overrightarrow{r}_3 \times \overrightarrow{r}_1, \tag{9.1}$$

$$\alpha_{cop} = \arcsin\left(\frac{\overrightarrow{Z}_{23} \cdot \overrightarrow{r}_1}{|\overrightarrow{Z}_{23}||\overrightarrow{r}_1|}\right), \tag{9.2}$$

$$\cos(\alpha_{12}) = \frac{\overrightarrow{r}_1 \cdot \overrightarrow{r}_2}{|\overrightarrow{r}_1||\overrightarrow{r}_2|}, \qquad \cos(\alpha_{23}) = \frac{\overrightarrow{r}_2 \cdot \overrightarrow{r}_3}{|\overrightarrow{r}_2||\overrightarrow{r}_3|}, \tag{9.3}$$

$$\begin{gathered}\overrightarrow{N} = r_1\overrightarrow{Z}_{23} + r_2\overrightarrow{Z}_{31} + r_3\overrightarrow{Z}_{12},\\ \overrightarrow{D} = \overrightarrow{Z}_{23} + \overrightarrow{Z}_{31} + \overrightarrow{Z}_{12},\\ \overrightarrow{S} = (r_2 - r_3)\overrightarrow{r}_1 + (r_3 - r_1), \overrightarrow{r}_2 + (r_1 - r_2)\overrightarrow{r}_3,\\ \overrightarrow{B} = \overrightarrow{D} \times \overrightarrow{2}_2.\end{gathered} \tag{9.4}$$

$$L_g = \sqrt{\frac{\mu}{ND}}, \tag{9.5}$$

$$\overrightarrow{v}_2 = \frac{L_g}{r_2}\overrightarrow{B} + L_g\overrightarrow{S}. \tag{9.6}$$

9.1.3. FUNCIÓN HERRICK_GIBBS

Se puede observar el código en A.2.

$$\begin{gathered}\Delta t_{31} = JD_3 - JD_1,\\ \Delta t_{32} = JD_3 - JD_2,\\ \Delta t_{21} = JD_2 - JD_1,\end{gathered} \tag{9.7}$$

$$\overrightarrow{Z}_{23} = \overrightarrow{r}_2 \times \overrightarrow{r}_3, \qquad \alpha_{cop} = \pi/2 - \arccos\left(\frac{\overrightarrow{Z}_{23} \cdot \overrightarrow{r}_1}{|\overrightarrow{Z}_{23}||\overrightarrow{r}_1|}\right). \tag{9.8}$$

$$\cos(\alpha_{12}) = \frac{\overrightarrow{r}_1 \cdot \overrightarrow{r}_2}{|\overrightarrow{r}_1||\overrightarrow{r}_2|}, \qquad \cos(\alpha_{23}) = \frac{\overrightarrow{r}_2 \cdot \overrightarrow{r}_3}{|\overrightarrow{r}_2||\overrightarrow{r}_3|}, \tag{9.9}$$

$$\begin{aligned}\overrightarrow{v}_2 = -\Delta t_{32}\left(\frac{1}{\Delta t_{21}\Delta t_{31}} + \frac{\mu}{12r_1^3}\right)\overrightarrow{r_1} + (\Delta t_{32} - \Delta t_{21})\left(\frac{1}{\Delta t_{21}\Delta t_{32}} + \frac{\mu}{12r_2^3}\right)\overrightarrow{r_2}\\ +\Delta t_{21}\left(\frac{1}{\Delta t_{32}\Delta t_{31}} + \frac{\mu}{12r_3^3}\right)\overrightarrow{r_3}.\end{aligned} \tag{9.10}$$

Se van a validar los dos métodos anteriores, para ello se va a utilizar un caso práctico que se encuentra en [21]. Los parámetros de entrada se pueden visualizar en la siguiente tabla 9.1.

Parámetros	Valores		
r0 (km)	3419.85564	6019.82602	2784.60022
t (s)		0	
r1 (km)	2935.91195	6326.18324	2660.59584
t1 (s)		1*60 + 16.48	
r2 (km)	2434.95202	6597.38674	2521.52311
t2 (s)		2*60 + 33.04	

Tabla 9.1: Validación Gibbs y Herrick Gibbs

Tras la ejecución de ambas funciones, se obtienen los siguientes resultados que se muestran en la tabla 9.2.

Parámetros	Valores			Método
V (Km/s)	-6.44163224	3.77762516	-1.72058257	Gibbs
V (Km/s)	-6.44155727	3.77755951	-1.7205676	Herrick Gibbs
Valor referencia V (km/s)	-6.441645	3.7776343	-1.720587	Libro

Tabla 9.2: Resultados Gibbs y Herrick Gibbs

Los métodos presentados a continuación suponen que las mediciones son perfectas y desprecian los términos de orden superior. Visto que las medidas que vamos a realizar no van a ser perfectas, es necesario realizar varias medidas, lo que lleva a nuevos problemas de correcciones diferenciales. Sobre este tema, se puede leer el capítulo 10 de [21] donde se encontrarán diferentes métodos para resolver el problema. En particular con la corrección por mínimos cuadrados o los filtros de Kalman. [27]

Los principios matemáticos de los mínimos cuadrados, ya sean lineales o no lineales, se pueden encontrar en los siguientes libros [21] [27] [25].

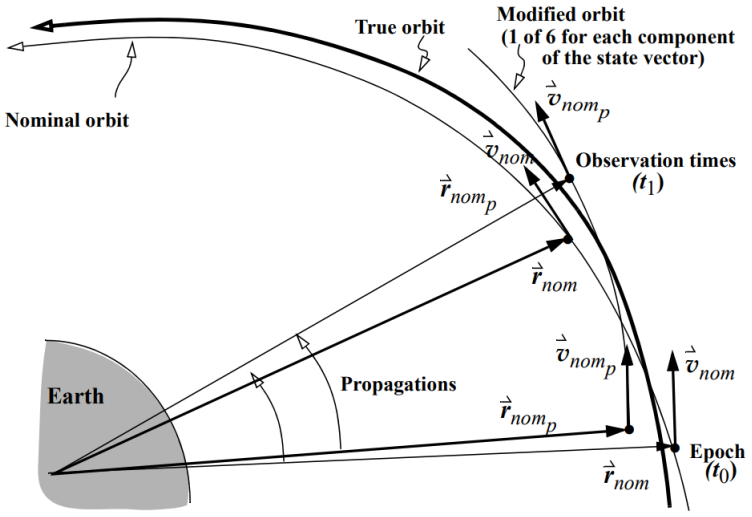


Figura 9.1: Correcciones diferenciales DPO [21]

9.1.4. TLE FILES

En este apartado se descodifica uno de los archivos estándar en el ámbito espacial, los archivos TLE de su nombre en inglés Two Lines Elements. La función recibe la dirección de un archivo **.tle** y devuelve un diccionario con todos y cada uno de los parámetros.

En este enlace se tienen todos los datos actualizados. [28]

También hay otra función que obtiene del archivo **.tle** los parámetros orbitales, haciendo uso de funciones de otros módulos, permite calcular el vector velocidad y el vector posición. Gracias a esto se puede calcular las órbitas de satélites reales haciendo uso de los propagadores desarrollados.

Campo	Columnas	Contenido	Ejemplo
1	01	Número de línea	1
2	03–07	Número de catálogo del satélite	25544
3	08	Clasificación (U: unclassified, C: classified, S: secret)	U
4	10–11	Año de lanzamiento	98
5	12–14	Número de lanzamiento del año	067
6	15–17	Piece of the launch	A
7	19–20	Epoch año (last two digits of year)	08
8	21–32	Epoch (día del año)	264.51782528
9	34–43	Derivada primera del movimiento medio	-.00002182
10	45–52	Derivada segunda del movimiento media	00000-0
11	54–61	*Drag o presión solar*	-11606-4
13	65–68	Número de elemento	292

Tabla 9.3: Línea 1 de un TLE

Campo	Columnas	Contenido	Ejemplo
1	01	Line number	2
3	09–16	inclinación (grados)	51.6416
4	18–25	RAAN (grados	247.4627
5	27–33	Excentricidad	0006703
6	35–42	Argumento del perigeo (grados)	130.5360
7	44–51	Anomalía media (grados)	325.0288
8	53–63	Movimiento medio (rev/días)	15.72125391
9	64–68	Revoluciones	56353

Tabla 9.4: Línea 2 de un TLE

Una vez decodificados se tiene una función que con los parámetros orbitales, se pueden sacar los vectores posición y velocidad inerciales. Con estos dos se puede utilizar cualquiera de las funciones del simulador, por ejemplo, se puede realizar la propagación de la órbita.

Capítulo 10

Maniobras interplanetarias

En este capítulo se exponen algunas de las funciones del simulador, como la resolución de algunas transferencias interplanetarias como las maniobras de Hohmann, transferencias bielípticas, etc. Además, se presenta la resolución del problema de Lambert.

10.1. CLASE LAMBERT: PROBLEMA DE LAMBERT

Conocidas dos posiciones en el espacio, la inicial r_0 y la deseada r_1, y el tiempo que se tarda en llegar de una a otra, el problema de Lambert consiste en determinar la trayectoria que permite llegar de la posición inicial a la final en el tiempo prefijado.

Según Lambert, el tiempo es independiente de la excentricidad de la órbita y depende únicamente de los vectores posición.

The orbital transfer time depends only upon the semimajor axis, the sum of the distances of the initial and final points of the arc from the center of force, and the length of the chord joining these points.

A través de los diferentes algoritmos de Lambert se puede calcular la v_0 que sería necesaria para alcanzar la posición r_1.

10.1.1. FUNCIÓN MINIMUM_ENERGY: MÉTODO DE MÍNIMA ENERGÍA

Este método de la clase Lambert se encarga de resolver el problema de Lambert para el caso de energía mínima.

La programación se ha seguido de acuerdo al **Algoritmo 56** del [21]

10.1.2. FUNCIÓN UNIVERSAL: VARIABLE UNIVERSAL

Siguiendo el desarrollo teórico de [9] se ha programado y comprobado la función con ejemplos del propio libro.

Se van a validar los métodos anteriores basándose en unos ejemplos que se encuentran disponibles en [21]. Estos ejemplos que se usan para la validación se comparan con nuestros resultados en la tabla 10.1

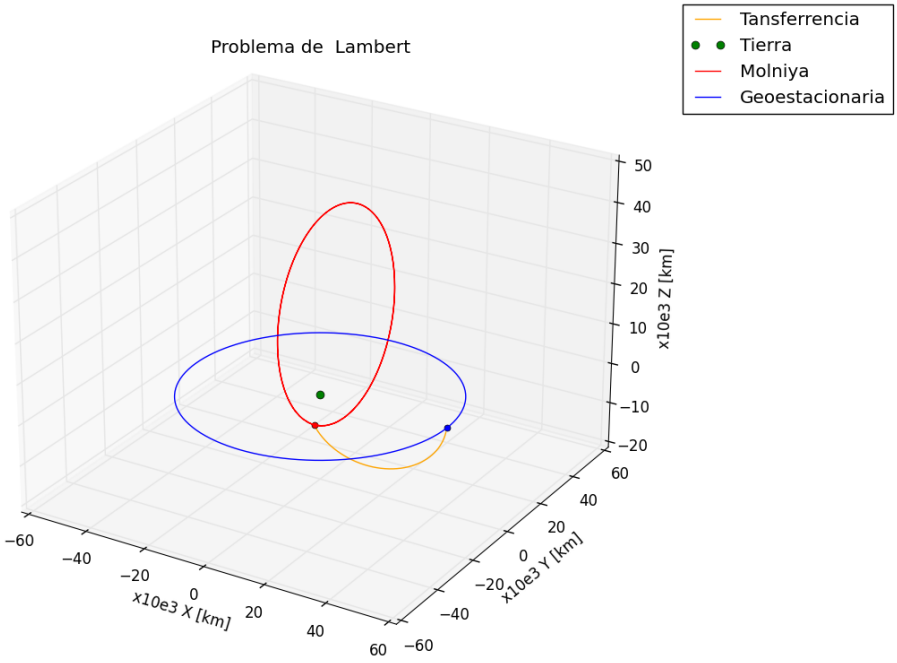


Figura 10.1: Problema de Lambert

	ri	rj	rk
r0 [km]	15,94534	0	0
r1 [km]	12214.84	10249.467	0
	V1	V2	V3
V0 [km/s]	2,058911	2,915964	0
V0 libro [km/s]	2,058925	2,915956	0
Error [%]	6,93E-04	2,66E-04	0
V1 [km/s]	-3,4515625	0,91031547	0
V1 libro [km/s]	-3,451569	0,910301	0
Error [%]	1,89E-04	1,59E-03	0

Tabla 10.1: Validación algoritmos de Lambert

10.2. CLASE TRANSFERTS

La clase `Interplanetary2D` tiene los siguientes atributos:

`mu : La constante gravitacional del cuerpo central.`
`radius : El radio del cuerpo central.`
`r0 : Vector de posición inicial.`
`r1 : Vector de posición final.`

10.2.1. HOHMANN

La función `hohmann` calcula la transferencia de Hohmann entre dos órbitas circulares. Los pasos principales son:

1. Se calcula la magnitud de los vectores de posición inicial y final (`r0` y `r1`).
2. Se calcula el semieje mayor promedio a utilizando la fórmula $(r0 + \texttt{radius})/2$.
3. Se calcula el delta-V (`dv1` y `dv2`) utilizando las fórmulas proporcionadas en el código.
4. Se calcula el tiempo de transferencia utilizando la fórmula proporcionada.
5. Se devuelven los valores de delta-V y tiempo.

La ecuación para calcular el delta-V (`dv1`) y el tiempo de transferencia en una transferencia de Hohmann es:

$$dv1 = \sqrt{\frac{\mu}{r0}} \left(\sqrt{\frac{2 \cdot \text{radius}}{r0 + \text{radius}}} - 1 \right) \tag{10.1}$$

$$dv2 = \sqrt{\frac{\mu}{\text{radius}}} \left(1 - \sqrt{\frac{2 \cdot a}{a + \text{radius}}} \right) \tag{10.2}$$

$$\text{time} = \sqrt{\frac{a^3}{\mu}} \pi \tag{10.3}$$

10.2.2. BIELÍPTICA

La función `bielliptic` calcula la transferencia bielíptica entre dos órbitas circulares. Los pasos principales son:

1. Se calcula la magnitud de los vectores de posición inicial y final (`r0` y `r1`).
2. Se calculan los semiejes mayores promedios $a1$ y $a2$ utilizando las fórmulas proporcionadas.
3. Se calculan los delta-V (`dv1`, `dv2` y `dv3`) utilizando las fórmulas proporcionadas.
4. Se calcula el tiempo de transferencia utilizando la fórmula proporcionada.
5. Se devuelven los valores de delta-V y tiempo.

La ecuación para calcular el delta-V (`dv1`, `dv2` y `dv3`) y el tiempo de transferencia

en una transferencia bielíptica es:

$$dv_1 = \sqrt{\mu\left(\frac{2}{r_0} - \frac{1}{a_1}\right)} - \sqrt{\frac{\mu}{r_0}} \tag{10.4}$$

$$dv_2 = \sqrt{\mu\left(\frac{2}{r_b} - \frac{1}{a_2}\right)} - \sqrt{\mu\left(\frac{2}{r_b} - \frac{1}{a_1}\right)} \tag{10.5}$$

$$dv_3 = \sqrt{\frac{\mu}{r_1}} - \sqrt{\mu\left(\frac{2}{r_1} - \frac{1}{a_2}\right)} \tag{10.6}$$

$$\text{time} = \sqrt{\frac{a_1^3}{\mu}}\pi + \sqrt{\frac{a_2^3}{\mu}}\pi \tag{10.7}$$

10.2.3. IMPULSO ÚNICO

La función `One_target_burn` calcula una transferencia de un solo impulso entre dos órbitas circulares. Los pasos principales son:

1. Se calcula la relación inversa de los vectores de posición inicial y final (`R_inv`).

$$R_{inv} = \frac{r0}{r1} \tag{10.8}$$

2. Dependiendo del lugar de la quema (periapsis o apoapsis), se calcula la excentricidad de la órbita de transferencia y el semieje mayor (`e_trans` y `a_trans`).

$$e_{trans} = \frac{R_{inv} - 1}{\cos(\nu_{transb}) - R_{inv}} \quad \text{(si la quema es en periapsis)} \tag{10.9}$$

$$e_{trans} = \frac{R_{inv} - 1}{\cos(\nu_{transb}) + R_{inv}} \quad \text{(si la quema es en apoapsis)} \tag{10.10}$$

3. Se calculan las velocidades, inicial (`v_ini`), de transferencia en la primera etapa (`v_trans_a`), final (`v_fin`) y de transferencia en la segunda etapa (`v_trans_b`) utilizando las fórmulas proporcionadas.

$$a_{trans} = \frac{r0}{1 - e_{trans}} \tag{10.11}$$

$$v_{ini} = \sqrt{\frac{\mu}{r0}} \tag{10.12}$$

$$v_{transa} = \sqrt{\mu\left(\frac{2}{r_0} - \frac{1}{a_{trans}}\right)} \tag{10.13}$$

$$v_{fin} = \sqrt{\frac{\mu}{r1}} \tag{10.14}$$

$$v_{transb} = \sqrt{\mu \left(\frac{2}{r_1} - \frac{1}{a_{trans}} \right)} \tag{10.15}$$

4. Se calculan los delta-V (`dva` y `dvb`) utilizando las fórmulas proporcionadas.

$$\delta v_a = v_{transa} - vini \tag{10.16}$$

$$\tan(\phi) = \frac{e_{trans} \sin(\nu_{transb})}{1 + e_{trans} \cos(\nu_{transb})} \tag{10.17}$$

$$\phi = \arctan(\tan(\phi)) \tag{10.18}$$

$$\delta v_b = \sqrt{v_{transb}^2 + v_{fin}^2 - 2 v_{transb} v_{fin} \cos(\phi)} \tag{10.19}$$

5. Se calcula el tiempo de transferencia utilizando la fórmula proporcionada.

$$\cos(E) = \frac{e_{trans} \sin(\nu_{transb})}{1 + e_{trans} \cos(\nu_{transb})} \tag{10.20}$$

$$E = \arccos(\cos(E)) \tag{10.21}$$

$$\text{time} = \sqrt{\frac{a_{trans}^3}{\mu}} \left(E - e_{trans} \sin(E) \right) \tag{10.22}$$

6. Se devuelve el valor de delta-V.

$$\delta v_{orb} = |\delta v_a| + |\delta v_b| \tag{10.23}$$

10.2.4. INCLINCACIÓN

Las funciones `inclination`, `change_ascension_node` y `change_ascn_inc` son funciones independientes que calculan diferentes cambios orbitales y devuelven el valor de delta-V.

En resumen, la clase `Interplanetary2D` proporciona métodos para calcular diferentes tipos de transferencias orbitales y cambios en la órbita utilizando las fórmulas y parámetros proporcionados.

Las funciones han sido validadas con ejemplos de uso de los libros [9] y [21].

Capítulo 11

Desarrollo de la aplicación

En esta sección se presenta una sencilla aplicación de consola que permite al usuario trabajar con las diferentes funciones que se han ido desarrollando a lo largo de este informe. Se puede ver la pantalla de inicio a través de la cual se navega para acceder a las diferentes funciones en la imagen 11.1 .

Las diferentes partes del código muestran los resultados por pantalla cuando estos no sean demasiado grandes, así como que permiten guardar los resultados en ficheros *.csv* o *.txt* que se guardan automáticamente en la carpeta *resuts/*. Muchas de las funciones pueden hacer uso de los datos generados por otras funciones.

Se ha de añadir que para obtener todo el potencial del simulador, se recomienda usar directamente el código fuente y realizar modificaciones al mismo según las necesidades del usuario. Al utilizar el simulador como una librería de Python de mecánica espacial, se puede acceder al 100 % de las funciones del mismo, así como la integración con otras librerías que se encuentran disponibles en Python. Todo esto se traduce en una mayor flexibilidad.

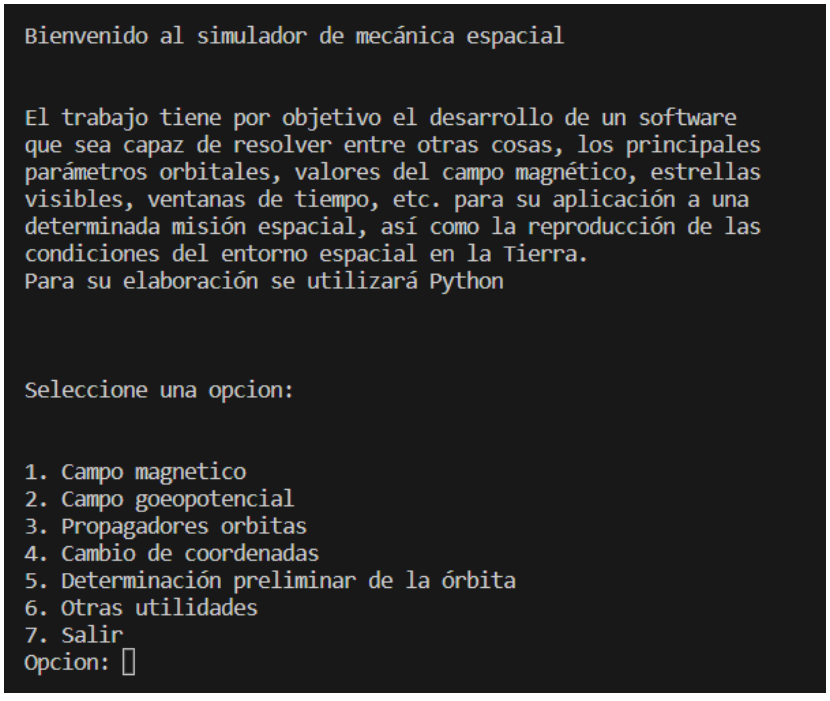


Figura 11.1: Pantalla de bienvenida

A continuación se muestran algunas de las opciones dentro del menú del campo magnético terrestre.

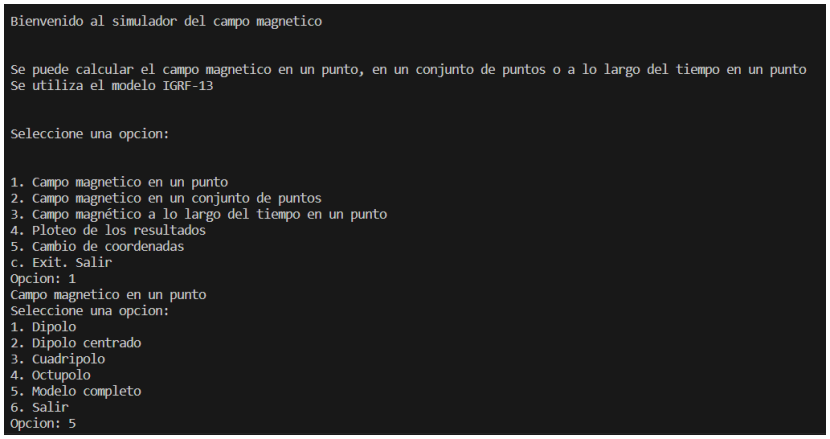


Figura 11.2: Menús

Una vez ejecutadas algunas partes del código se tiene la posibilidad de guardar los datos obtenidos, estos datos son guardados en la carpeta *results*, como se muestra en la figura 11.3.

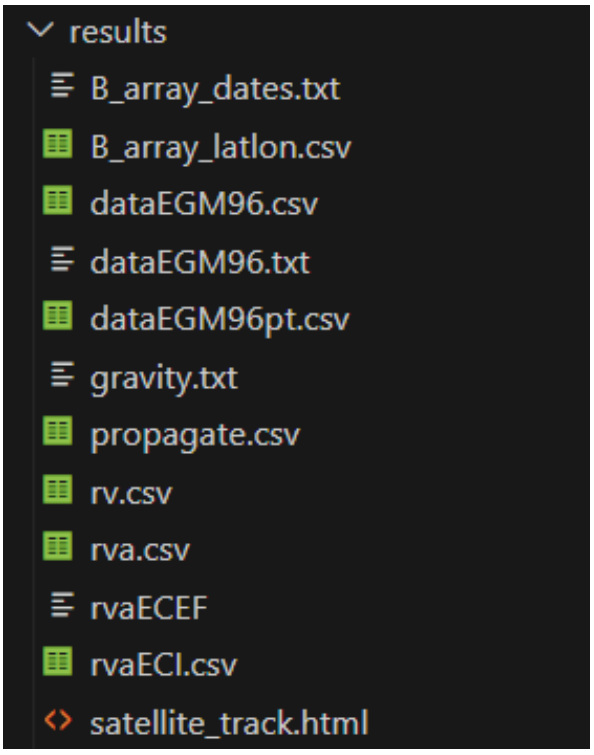


Figura 11.3: Carpeta donde se guardan los resultados

11.1. ESTRUCTURACIÓN DE LA APLICACIÓN

En esta sección se va a comentar brevemente como se ha organizado el conjunto de scripts que permiten hacer uso de la aplicación. En primer lugar, la aplicación está desarrollada dentro de la carpeta ***Simulador/scripts***. Dentro de esta carpeta, como se puede apreciar, hay diferentes archivos:

- **App.py**: Se trata del núcleo de la aplicación, en ella se encuentran los diferentes menús con las diferentes opciones. Cada una de estas opciones se desarrolla en el resto de archivos *.py* que se encuentran en la carpeta donde se cargan las clases y librarías desarrolladas en ***Simulador/src***
- **Coor.py**: Este archivo carga la librería Coordinates y hace uso de las diferentes funciones que se encuentran desarrolladas dentro de la misma. Entre otras funciones que se encuentran disponibles se pueden encontrar:
  - Cambio de coordenadas ECI a ECEF.
  - Cambio de coordenadas ECEF a ECI.
  - RV a COE.
  - COE a RV.
  - ECEF a latitud, longitud y altura.
- **DPO.py**: En este script se encuentran las utilidades relativas a la determinación preliminar de órbitas, como funciones principales se tienen:
  - Observaciones del vector posición, Gibbs y Herrick-Gibbs.
  - Observaciones de latitud, longitud, altura y variación de latitud, longitud y altura.
  - Observaciones desde la Tierra, desde el centro de la Tierra y tiempo. Método iterativo de Gauss
  - Vectores posición y velocidad desde un *.tle*.
- **Geo.py**: Este script se encarga de realizar los cálculos del vector gravedad el cualquier punto de la Tierra así como su potencial, es decir, todas las funciones que se encuentran dentro de ***Simulador/src/geomag.py***.
  - Campo geopotencial en un punto.
  - Campo geopotencial en un conjunto de puntos.
  - Representación del campo geopotencial y de la gravedad.
- **Mag.py**: Esta es una de las partes más importantes del simulador, contiene las funcionalidades de la clase Geomag que se encuentra en ***Simulador/src.geomag.py***, entre sus características principales se tiene:
  - Campo magnético en un punto.
  - Campo magnético en un conjunto de puntos.
  - Campo magnético a lo largo del tiempo en un punto.
  - Ploteo de los resultados.
  - Cambio de coordenadas. En este apartado, se puede obtener los resultados en coordenadas esféricas, en coordenadas NED, en coordenadas inerciales, en coordenadas orbitales e incluso en coordenadas ejes cuerpo. Por otro lado, se puede pasar de coordenadas geodésicas a geocéntricas y viceversa.
- **Prop.py**: Esta clase se encarga de servir como un sencillo propagador de órbitas

que permite obtener a partir de un vector posición y velocidad inicial $\vec{r}_0$ y $\vec{v}_0$ la órbita completa.

- Órbita en un instante de tiempo mediante el método de Kepley o Pkepler del Vallado [21].
- Órbita a lo largo del tiempo.
- Ploteo de los resultados.

Capítulo 12

Caso de estudio

En este apartado se va a realizar una simulación de lo que sería el análisis de una misión espacial utilizando las herramientas que se han desarrollado y validado a lo largo del proyecto.

Se va a realizar un análisis del campo magnético y el potencial terrestre a lo largo de la órbita de un satélite español de tecnología radar destinado, no sólo a cubrir las necesidades de seguridad y defensa, sino también otras de carácter civil, más concretamente el satélite PAZ [29].

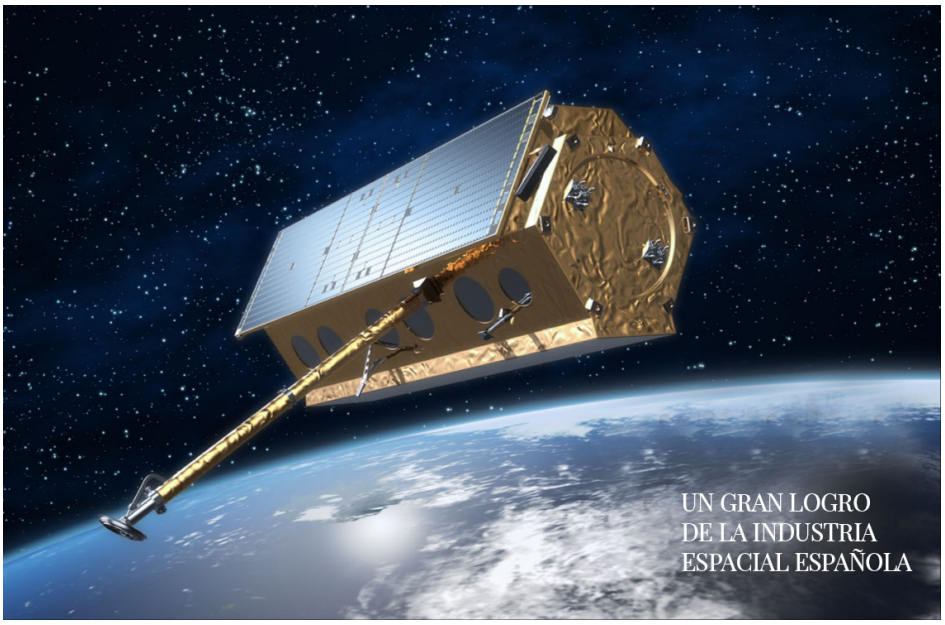


Figura 12.1: Satélite PAZ [29]

12.1. ETAPA 1: OBTENCIÓN DE DATOS

En primer lugar, se va a utilizar la base de datos de *Celestrack* [5] donde se obtiene el fichero *.tle* del satélite de interés, para el caso de estudio el satélite PAZ.

El archivo TLE es el siguiente:

```
1  PAZ
2  1 43215U 18020A   23050.16781453  .00000107  00000+0  82680-5 0  9997
3  2 43215  97.4463  58.9616 0001892  93.7517 337.1362 15.19152901276708
```

Figura 12.2: Archivo paz.tle

Para decodificar el siguiente archivo, se puede usar tanto las funciones que se en-

cuentran disponibles en la clase ***Simulador/src/utilities.py*** o directamente la aplicación desarrollada en la siguiente ruta ***Determinación preliminar de órbitas - r y v desde tle***. Consultar el apartado 9.1.4 para obtener más detalles.

```
Decodificar TLE
Introduzca la ruta del .TLE
Ruta: data/paz.tle
Nombre: PAZ
Inclinacion: 97.4463
RAAN: 58.9616
Excentricidad: 0.0001892
Argumento del parigeo: 93.7517
Anomalía media: 337.1362
Movimiento medio: 15.19152901
Epoch: 2023-02-19 00:00:00
1º derivada del movimiento medio: 1.07e-06
1º derivada del movimiento medio: 0.0
Backend TkAgg is interactive backend. Turning interactive mode on.
Warning: No body selected. Using Earth as default.
Name: PAZ
Posicion:  [ 6419.34362949  5099.74637383 21964.50539044] km
Velocidad:  [-1.8590125  -3.42865408  1.33906034] km/s
```

Figura 12.3: R y V desde tle con aplicación.

Se obtienen los valores de los vectores posición $\vec{r}_0$ y velocidad $\vec{v}_0$ inicial:

	1	2	3
Posición [km]	6419.3436	5099.74637	21964.5053
Velocidad [km/s]	-1.8590125	-3.42865408	1.33906034

Tabla 12.1: Resultados de r y v del satélite PAZ

12.2. ETAPA 2: PROPAGACIÓN DE LA ÓRBITA

Una vez obtenidos los valores de los vectores posición $\vec{r}_0$ y velocidad $\vec{v}_0$ inicial, se puede comenzar la propagación de la órbita a lo largo del tiempo. En el caso de estudio, como demostración se va a propagar la órbita durante una hora con intervalos de cálculo de 3 minutos.

Vamos a propagar la órbita haciendo uso de la aplicación en el menú de propagación de órbitas. Análogamente, se podría utilizar la clase disponible en ***Simulador/src/-Propagator.py*** o ***Simulator/src/kepler_Propagator.py***.

x [km]	y [km]	z [km]	Vx [km/s]	Vy [km/s]	Vz [km/s]
6419.34	5099.75	21964.51	-1.86	-3.43	1.34
6418.57	5098.31	21965.07	-1.86	-3.43	1.34
6080.05	4477.39	22195.46	-1.89	-3.46	1.22
5385.90	3224.24	22588.59	-1.96	-3.50	0.97
4305.57	1322.76	23007.73	-2.04	-3.54	0.58
2806.91	-1227.40	23243.36	-2.12	-3.54	0.07
873.68	-4382.04	23014.64	-2.17	-3.46	-0.58
-1470.31	-8015.66	21982.57	-2.16	-3.25	-1.33
-4131.93	-11881.57	19784.29	-2.05	-2.86	-2.15
-6919.09	-15578.80	16097.21	-1.80	-2.25	-2.95
-9519.67	-18542.15	10737.94	-1.39	-1.39	-3.63
-11503.16	-20077.53	3790.66	-0.80	-0.30	-4.03
-12362.51	-19465.57	-4258.06	-0.06	0.92	-4.02
-11611.47	-16147.34	-12425.10	0.75	2.12	-3.46
-8941.49	-9980.96	-19246.85	1.50	3.08	-2.29
-4416.75	-1513.94	-22983.69	2.03	3.53	-0.62
1350.85	7838.29	-22063.19	2.16	3.26	1.29
7143.46	15860.14	-15729.71	1.77	2.19	3.01
11307.29	19992.06	-4729.13	0.88	0.45	4.00
12221.18	18208.42	8299.46	-0.33	-1.52	3.82
9009.45	10124.45	19131.36	-1.49	-3.06	2.32

Tabla 12.2: Resultados propagación Pkepler

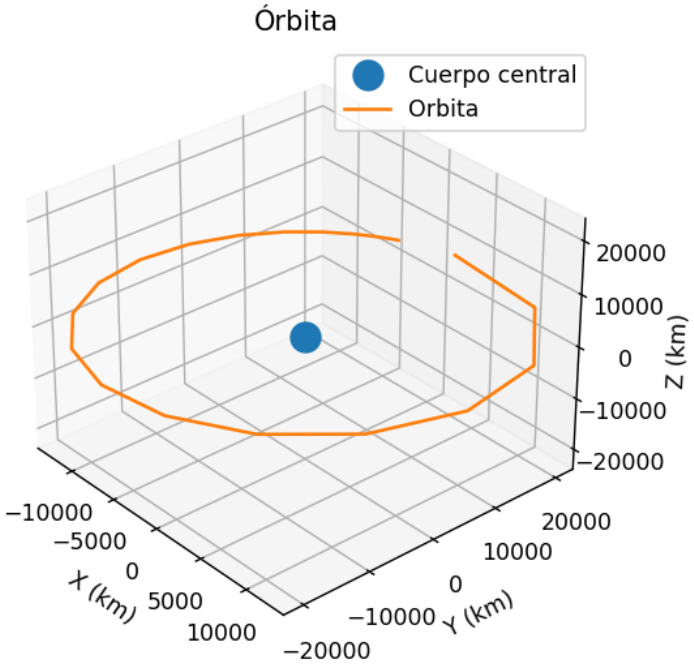


Figura 12.4: Representación órbita satélite PAZ

12.3. ETAPA 3: CAMBIO DE COORDENADAS

En esta etapa, hay que darse cuenta de que se está trabajando en un sistema de coordenadas inerciales (ECI), y se necesita para su uso en los algoritmos de cálculo del campo magnético y el potencial terrestre, los datos en un sistema coordenado ECEF.

Para esta etapa se hace uso de la librería ***Simulador/src/Coordinates.py***

x [km]	y [km]	z [km]	Vx [km/s]	Vy [km/s]	Vz [km/s]
608.08	8168.67	21967.20	1.85	-3.74	1.34
608.61	8167.12	21967.76	1.85	-3.74	1.34
834.66	7496.97	22198.00	1.80	-3.80	1.21
1283.47	6136.44	22590.81	1.69	-3.91	0.97
1946.15	4052.20	23009.46	1.51	-4.04	0.58
2801.18	1220.50	23244.42	1.25	-4.16	0.07
3805.15	-2340.51	23014.82	0.90	-4.22	-0.58
4882.16	-6528.03	21981.70	0.45	-4.15	-1.33
5914.38	-11106.11	19782.24	-0.10	-3.88	-2.15
6737.87	-15661.39	16093.94	-0.71	-3.34	-2.95
7149.58	-19580.94	10733.54	-1.35	-2.48	-3.63
6932.08	-22077.50	3785.43	-1.93	-1.30	-4.03
5901.30	-22290.53	-4263.61	-2.34	0.15	-4.02
3976.67	-19483.88	-12430.23	-2.47	1.70	-3.46
1263.14	-13335.12	-19250.71	-2.21	3.10	-2.29
-1881.26	-4263.63	-22985.47	-1.53	4.03	-0.62
-4833.00	6319.93	-22062.38	-0.47	4.15	1.29
-6792.23	16017.02	-15726.34	0.77	3.28	3.01
-7002.08	21875.95	-4723.98	1.86	1.46	4.00
-5075.13	21332.06	8304.91	2.45	-0.91	3.82
-1322.33	13482.47	19135.25	2.22	-3.08	2.32

Tabla 12.3: Órbita en coordenadas ECEF

Una vez obtenidos los vectores en un sistema de referencia ECEF, se puede continuar el cambio de coordenada. Se procede a cambiar de coordenadas mediante la función **ecef2latlongh** para obtener la latitud geocéntrica, longitud y la altitud. Los resultados se muestran en la siguiente tabla 12.4.

Latitud [grados]	Longitud [grados]	r [Km]
85.7427059	69.458289	17085.358
85.7382432	69.4622175	17085.3588
83.6472352	71.1457447	17085.7019
78.1865745	74.4176573	17086.2939
64.3464048	78.8926149	17086.9298
23.5431355	82.4749976	17087.2713
-31.595266	78.9625476	17086.8558
-53.208046	69.5615995	17085.1902
-61.9631821	57.4140136	17081.9768
-66.7215712	43.2090011	17077.4735
-69.9413779	27.130535	17072.8508
-72.5683016	9.24596833	17070.2201
-75.1714485	-10.4258891	17071.9033
-78.4643412	-31.8831414	17078.8001
-84.5889278	-55.0379269	17088.5819
-66.1913479	-78.484131	17095.5429
52.5940237	-70.0801628	17093.8868
67.0199823	-41.9720457	17083.4953
72.2511148	-11.5665053	17072.7219
76.61749	20.6512033	17072.0432
84.3984674	54.5703968	17081.4373

Tabla 12.4: Latitud, longitud y altura del satélite PAZ

12.4. ETAPA 4: CÁLCULO DE CAMPO MAGNÉTICO (IGRF13)

Utilizando los resultados que se guardaron en la ruta *Simulador/results/propagate_lla.csv*

Se hace uso de la aplicación, la función de leer de un archivo, los datos para calcular el valor del campo magnético en los diferentes puntos.

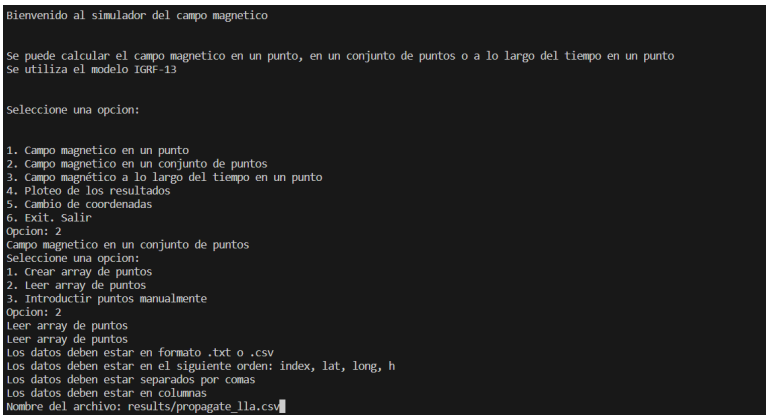


Figura 12.5: Captura de la aplicación

r [km]	Lat [deg]	Long [deg]	Br [nT]	Btheta [nT]	Bphi [nT]	BN [nT]	BE [nT]	BD [nT]
17085.36	85.74	69.46	-3122.56	-242	-55.01	242	-55.01	3122.56
17085.36	85.74	69.46	-3122.53	-242.12	-55	242.12	-55	3122.53
17085.7	83.65	71.15	-3105.75	-298.66	-50.22	298.66	-50.22	3105.75
17086.29	78.19	74.42	-3043.82	-444.65	-41.77	444.65	-41.77	3043.82
17086.93	64.35	78.89	-2766.37	-806.02	-35.43	806.02	-35.43	2766.37
17087.27	23.54	82.47	-966.11	-1601.4	-69.83	1601.4	-69.83	966.11
17086.86	-31.6	78.96	2066.55	-1161.5	-205.39	1161.5	-205.39	-2066.55
17085.19	-53.21	69.56	2646.21	-667.95	-290.63	667.95	-290.63	-2646.21
17081.98	-61.96	57.41	2718.29	-505.2	-341.23	505.2	-341.23	-2718.29
17077.47	-66.72	43.21	2700.03	-467.16	-371.21	467.16	-371.21	-2700.03
17072.85	-69.94	27.13	2661.13	-486.8	-376.79	486.8	-376.79	-2661.13
17070.22	-72.57	9.25	2625.89	-534.32	-350.57	534.32	-350.57	-2625.89
17071.9	-75.17	-10.43	2612.84	-586.05	-284.3	586.05	-284.3	-2612.84
17078.8	-78.46	-31.88	2646.26	-608.38	-171.72	608.38	-171.72	-2646.26
17088.58	-84.59	-55.04	2772.77	-524.57	-14.84	524.57	-14.84	-2772.77
17095.54	-66.19	-78.48	2316.2	-868.37	117.6	868.37	117.6	-2316.2
17093.89	52.59	-70.08	-2657.36	-784.3	-83.69	784.3	-83.69	2657.36
17083.5	67.02	-41.97	-2936.93	-520.35	-138.77	520.35	-138.77	2936.93
17072.72	72.25	-11.57	-2992.39	-470.58	-146.92	470.58	-146.92	2992.39
17072.04	76.62	20.65	-3039.67	-422.26	-121.33	422.26	-121.33	3039.67
17081.44	84.4	54.57	-3115.68	-261.4	-82.16	261.4	-82.16	3115.68

Tabla 12.5: Resultados del campo magnético en la órbita del satélite PAZ

Los valores se han guardado en **Simulador/results/B_array_latlon.csv** para su posterior uso en otras aplicaciones.

12.5. ETAPA 5: CÁLCULO DE POTENCIAL TERRESTRE (EGM96)

Análogamente, al apartado anterior, se usa la aplicación de manera análoga al apartado anterior, pero en este caso para el cálculo de del potencial terrestre y el vector gravedad $\vec{g}$.

r [km]	Lat [deg]	Long [km]	Potencial [mˆ2/sˆ2]	g1 [m/sˆ2]	g2 [m/sˆ2]	g3 [m/sˆ2]
17085.36	85.74	69.46	23326456.90	-0.48	-1.27	-0.10
17085.36	85.74	69.46	23326455.83	-0.48	-1.27	-0.10
17085.70	83.65	71.15	23326022.97	-0.44	-1.28	-0.15
17086.29	78.19	74.42	23325370.68	-0.36	-1.29	-0.28
17086.93	64.35	78.89	23325265.54	-0.24	-1.21	-0.59
17087.27	23.54	82.47	23328227.49	-0.07	-0.54	-1.25
17086.86	-31.60	78.96	23328198.08	0.14	0.70	-1.16
17085.19	-53.21	69.56	23328545.23	0.38	1.02	-0.82
17081.98	-61.96	57.41	23332208.61	0.65	1.02	-0.64
17077.47	-66.72	43.21	23338019.34	0.91	0.86	-0.54
17072.85	-69.94	27.13	23344133.59	1.14	0.59	-0.47
17070.22	-72.57	9.25	23347582.42	1.29	0.21	-0.41
17071.90	-75.17	-10.43	23345151.56	1.30	-0.24	-0.35
17078.80	-78.46	-31.88	23335590.07	1.14	-0.71	-0.27
17088.58	-84.59	-55.04	23322069.73	0.78	-1.11	-0.13
17095.54	-66.19	-78.48	23313387.40	0.25	-1.22	-0.55
17093.89	52.59	-70.08	23316731.67	-0.37	1.02	-0.83
17083.50	67.02	-41.97	23329778.56	-0.93	0.84	-0.53
17072.72	72.25	-11.57	23344183.95	-1.28	0.26	-0.42
17072.04	76.62	20.65	23344901.21	-1.24	-0.47	-0.32
17081.44	84.40	54.57	23331830.91	-0.79	-1.11	-0.13

Tabla 12.6: Resultados del campo potencial terrestre en la órbita del satélite PAZ

Capítulo 13

Conclusión

Este proyecto permite hacer uso de las bases de la mecánica espacial y a través de las mismas obtener una serie de valores de interés. Los dos grandes ejes de este proyecto han sido el campo magnético y el campo gravitatorio, alrededor de los cuales se han ido creando una serie de funciones que permiten transformar los valores obtenidos por los diferentes modelos a nuestras necesidades. Alrededor de estos ejes principales, se han ido contrayendo una serie de funcionalidades que complementan la actual librería de mecánica espacial. Algunas de las funcionalidades extra son la propagación de órbitas, el cambio de coordenadas, etc.

Finalmente, se ha desarrollado una sencilla aplicación de consola de comandos que permite acceder a alguna de las funcionalidades principales. Sin embargo, para obtener una mayor flexibilidad se ha de usar como una librería en Python, donde se podrá complementar su uso con otras herramientas disponible. Todo esto hace que Python y su comunidad hagan de este proyecto, un simulador de mecánica espacial complete, versátil y flexible.

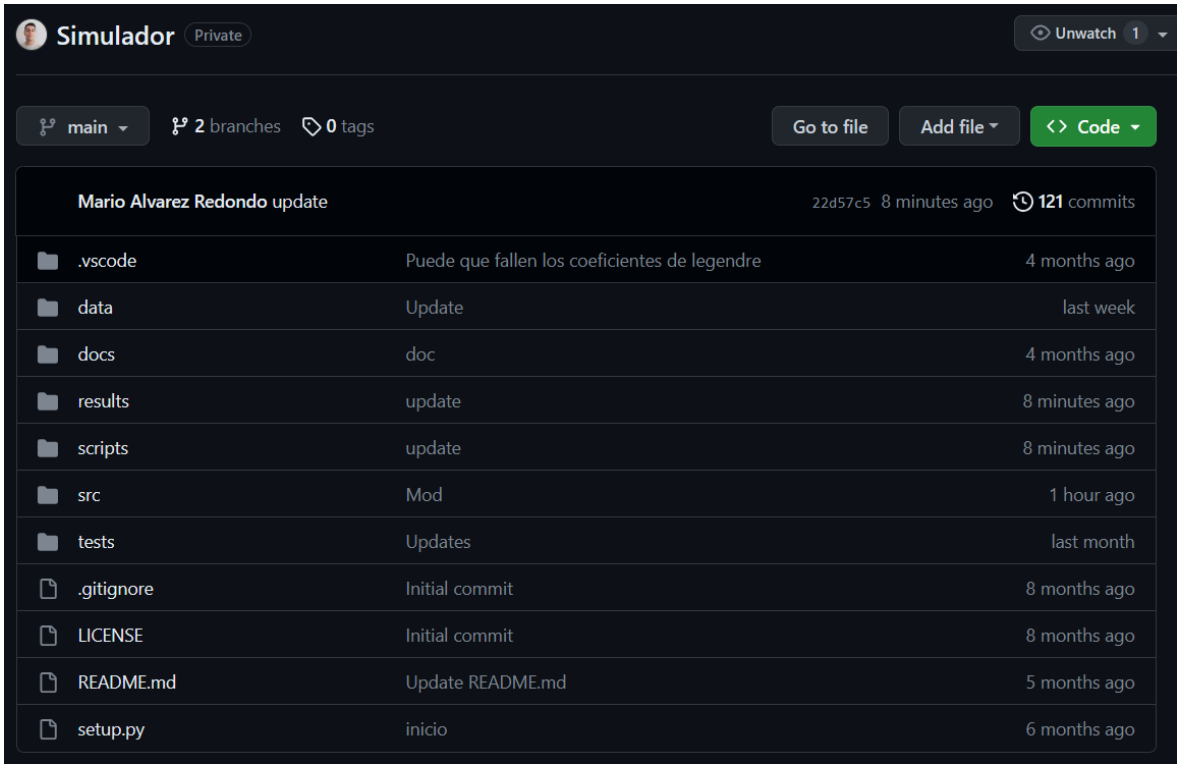


Figura 13.1: Github

Se han aplicado una serie de metodología de programación estructurada siguiendo

las indicaciones del manual goodCode [30], así como un control de versiones con **git** y el uso continuo de **Github** para tener siempre una copia en la nube del proyecto.

El código completo del programa puede obtenerse a través del siguiente repositorio de *Github* `mario-alvarez-redondo/Simulador`

13.1. FUTURAS MEJORAS

En la siguiente tabla se muestran una serie de posibles mejoras que serán implementadas en un futuro desarrollo.

Área de majora	Estado actual	Mejora a desarrollar	Tiempo estimado de desarrollo
Interfaz	Consola de comandos	Desarrollo GUI	>2 semanas
Entorno magnético	IGRF13	Añadir modelo WWM	<1 semanas
Entorno geopotencial	EGM96	Añadir modelos EGM08	>2 semanas
Sistema de ejes coordenados	Desarrollado	Implementar de actualización automática de los valores de nutación, variación de los polos...	2 semanas
Propagadores de órbitas	Implementación de modelos ideales y con perturbaciones J2, 3^{o} cuerpo...	Añadir más perturbaciones para obtener un propagador más realista	>2 semanas
Maniobras interplanetarias	Desarrolladas	Iterconexión de las funciones dentro de la interfaz	>2 semanas
Determinación preliminar de órbitas	Desarrolladas funciones en base a observaciones instantáneas	Implementación de correcciones diferenciales mediante el método de mínimos cuadrados y filtros de Kalman	>3 semanas

Tabla 13.1: Futuras mejoras

Apéndice A

Código Python destacado

```
1
2  def r0v02rv(self, r0, v0, dt):
3      '''Final position and velocity for elliptic, parabolic and hyperbolic
       orbits
4      from initial position, velocity and time since periapsis
5
6      Parameters
7      ----------
8      r0 : float
9          Initial position vector in km
10     v0 : float
11         Initial velocity vector in km/s
12     dt : float
13         Time since periapsis in seconds
14     Returns
15     -------
16     r : float
17         Final position vector in km
18     v : float
19         Final velocity vector in km/s
20     '''
21
22     # Normalization
23     r0n = np.linalg.norm(np.array(r0))
24     v0n = np.linalg.norm(np.array(v0))
25
26     # Radial component of initial velocity
27     vr0 = np.dot(r0, v0) / r0n
28
29     # alpha
30     alpha = (2 / r0n - v0n**2 / self.mu)
31
32     # The sign of alpha determines the type of orbit
33     # alpha > 0 : elliptic orbit
34     # alpha = 0 : parabolic orbit
35     # alpha < 0 : hyperbolic orbit
36
37     chi = self.Kepler_universal(r0n, vr0, dt, alpha)
38
39     f, g = self.lagrange_coeff(chi, dt, r0n, alpha)
```

```
40
41      r = f * r0 + g * v0
42
43      rn = np.linalg.norm(np.array(r))
44
45      df, dg = self.d_lagrange_coeff(chi, rn, r0n, alpha)
46
47      v = df * r0 + dg * v0
48
49      return r, v
```

Extracto de código A.1: Propagador de usando la anomalía universal

```
1
2  def HERRICK_GIBBS(self, R0, R1, R2, t0, t1, t2):
3      '''This method is used to find the velocity vectors at
4      the intermediate position like the Gibbs method, but it
5      uses the time of flight to find the velocity vectors.It
6      performs better than the Gibbs method when the position
7      vectors are close to each other (<1  ).
8
9      Parameters
10     ----------
11     R0 : numpy array
12         Initial position vector
13     R1 : numpy array
14         Intermediate position vector
15     R2 : numpy array
16         Final position vector
17     t0 : float
18         Initial time
19     t1 : float
20         Intermediate time
21     t2 : float
22         Final time
23     Returns
24     ----------
25     V1 : numpy array
26         Intermediate velocity vector
27     '''
28
29     # Parameters
30     dt01 = np.abs(t1 - t0)
31     dt12 = np.abs(t2 - t1)
32     dt02 = np.abs(t2 - t0)
33
34     r0 = np.linalg.norm(R0)
```

```
    r1 = np.linalg.norm(R1)
    r2 = np.linalg.norm(R2)

    Z12 = np.cross(R1, R2)

    alpha_cop = np.pi / 2 - np.arccos(np.dot(Z12, R0) / np.linalg.norm(
    Z12) / np.linalg.norm(R0))

    # Angles between the vectors
    alpha01 = np.arccos(np.dot(R0, R1) / r0 / r1)
    alpha12 = np.arccos(np.dot(R1, R2) / r1 / r2)

    # Show yhe angles in degrees
    print('The angles between the vectors are: ', np.rad2deg(alpha01), np
    .rad2deg(alpha12))

    A = np.dot(-dt12 * (1/(dt01 * dt02) + self.mu/(12*r0**3)), R0)
    B = np.dot((dt12 - dt01) * (1/(dt01 * dt12) + self.mu/(12*r2**3)), R1
    )
    C = np.dot(dt01 * (1/(dt12 * dt02) + self.mu/(12*r2**3)), R2)

    v2 = A + B  + C

    return v2
```

Extracto de código A.2: Función de Herrick Gibbs

```

def cart2efix(r, v, t, dt, t0):
    ''' Space fixed coordinate system to Earth fixed coordinate system
    Parameters
    ----------
    r : array_like
        Position vector in km
    v : array_like
        Velocity vector in km/s
    t : float
        Time in seconds after t0
    dt : float
        Time step in seconds
    t0 : float
        Time of reference in seconds
    Returns
    -------
    r : array_like
        Position vector in km
    v : array_like
```

```
        Velocity vector in km/s
    '''

    # Earth's rotation rate
    omega = 7.2921158553e-5

    hour = t0 / 3600
    sid = hour * 15 * np.pi / 180

    t = np.arange(0, t+dt, dt)

    theta0 = omega * t + sid

    for i in range(len(theta0)):
        R = np.transpose(rotation_matrix_3(theta0[i]))
        r[i, :] = np.dot(R, r[i, :])
        v[i, :] = np.dot(R, v[i, :])

    return r, v
```

Extracto de código A.3: Función cambio de coordenadas

```

def get_all_txt_from_url(self, url="https://www.ncei.noaa.gov/products/
    international-geomagnetic-reference-field"):

        # Get the content of the web page
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "html.parser")

        # Find all the links to .txt files
        txt_files = soup.find_all("a", href=re.compile("\.txt$"))

        # Find the substring in the content of the .txt files
        substring = "igrf"

        urls = []

        for txt_file in txt_files:
            txt_url = txt_file["href"] # Get the url of the .txt file
            urls.append(txt_url) # Add the url to the list

        print(urls) # Print the list of urls

        num_max = 0
        for cadena in urls:
            idx = cadena.index('igrf') + 4
```

```
            idx_f = idx

            for ii in cadena[idx:]:
                idx_f += 1
                if ii.isdigit() == False:
                    idx_f -= 1
                    break

            valor = int(cadena[idx:idx_f])
            if valor > num_max:
                num_max = valor

        print('igrf' + str(num_max))

        # Find the num_max between the urls

        for url_txt in urls:
            if 'igrf' + str(num_max) in url_txt:
                return url_txt
            else:
                return None
```

Extracto de código A.4: Función de la clase Geomag

```
def magnetic_field(self, r, theta, phi, year=2021, N=3):

        '''This function calculates the magnetic field at a given location
        Parameters
        ----------
        phi : float
            Longitude measured in degrees positive east from Greenwich (in degrees)
        theta : float
            The latitude measured in degrees positive from equator (in degrees)
        r : float
            The distance from the center of the Earth to the point where to calculate the magnetic field (in km)
        N : int
            The order of the magnetic field to calculate
        Returns
        -------
        B : float
            The magnetic field at the given location
        '''
        # Checks to see if located at either pole to avoid singularities
```

```
20          theta = 90 - theta
21
22          if -0.00000001 < theta < 0.00000001:
23              theta = 0.00000001
24          elif 179.99999999 < theta < 180.00000001:
25              theta = 179.99999999
26
27
28          # Convert to radians
29          phi = np.radians(phi)
30          theta = np.radians(theta)
31
32          # Verify the limits of the input
33          if r < 0:
34              raise ValueError('The distance must be positive')
35          if r < self.Re:
36              raise ValueError('The distance must be higher than the radius
     of the Earth')
37          if phi < -np.pi or phi > np.pi:
38              raise ValueError('The co-elevation must be between -180 and
    180')
39          if theta < 0 or theta > np.pi:
40              raise ValueError('The longitude must be between -90 and 90')
41
42          Snm = self.Snm(nm_max=N)
43          Pnm, dPnm = self.gauss_norm_ass_leg_poly(nm_max=N, theta=theta)
44          # Pnm = self.legendre_poly(N, theta)
45
46
47          if not 0<N<=13:
48              raise ValueError('The order must be between 1 and 13')
49
50          Br = 0
51          Btheta = 0
52          Bphi = 0
53
54          for m in range(N+1):
55              for n in range(1, N+1):
56                  if m <= n:
57                      gh = self.get_gh_norm(n, m, year, 'b')
58                      g = gh['g']
59
60                      if m != 0:
61                          h = gh['h']
62                      else:
63                          h = 0
```

```
64
65                  Br += (self.Re / r)**(n+2)* (n + 1) * Pnm[n, m] * (g
    * np.cos(m*phi) + h*np.sin(m*phi))
66                  Btheta += -(self.Re/r)**(n+2) * dPnm[n, m] * (g * np.
    cos(m*phi) + h * np.sin(m*phi))
67
68                  # Pnm[m, n+1] dPnm[n, m]
69
70                  if theta == 0 or theta == np.pi:
71                      Bphi = np.inf
72                  else:
73                      Bphi -= (self.Re/r)**(n+2) * (-m*g*np.sin(m*phi)
    + h*m*np.cos(m*phi)) * Pnm[n, m] / np.sin(theta)
74
75                  # print(n, m, Br, Btheta, Bphi)
76
77      Bvector = np.array([Br, Btheta, Bphi])
78      Bmodule = np.linalg.norm(Bvector)
79
80      return Bmodule, Bvector
```

Extracto de código A.5: Función de la clase Geomag modelo

Bibliografía

[1] K. Bootcamps. «Ventajas y Desventajas de Python.» (2023), dirección: https://keepcoding.io/blog/ventajas-y-desventajas-de-python/.

[2] NASA. «GMAT.» (2023), dirección: https://software.nasa.gov/software/GSC-18094-1.

[3] AGI. «Ansys STK.» (jul. de 2022), dirección: https://www.ansys.com/products/missions/%20ansys-stk%5C#tab1-2.

[4] Wikipedia. «Orbiter (simulador).» (2023), dirección: https://es.wikipedia.org/wiki/Orbiter_(simulador).

[5] D. T. Kelso. «Celestrack.» (2023), dirección: http://celestrak.org/.

[6] C. by CNES. «SIMU-CIC.» (2022), dirección: https://www.connectbycnes.fr/en/simu-cic.

[7] J. L. C. Rodríguez, Y. Gondhalekar, A. Hidalgo et al. «poliastro/poliastro: poliastro 0.17.0 (SciPy US '22 edition).» ver. v0.17.0. (jul. de 2022), dirección: https://doi.org/10.5281/zenodo.6817189.

[8] J. L. Cano. «Orbit-predictor.» (2023), dirección: https://pypi.org/project/orbit-predictor/#:~:text=Orbit%5C%20Predictor%5C%20is%5C%20a%5C%20Python%5C%20library%5C%20to%5C%20propagate,a%5C%20%5C%E2%5C%80%5C%9Cwrapper%5C%E2%5C%80%5C%9D%5C%20for%5C%20the%5C%20python%5C%20implementation%5C%20of%5C%20SGP4.

[9] H. Curtis, «Orbital Mechanics for engeneering students,» *Elsevier Aerospace Engeneering series*, págs. 1-101, 2005.

[10] R. H. Battin, «An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition,» *American Institute of Aeronautics and Astronautics, Inc.*, págs. 107-341, 1999.

[11] B. A. C. John E. Prussing, «Orbital Mechanics,» *Oxford University Press, Inc.*, 1993.

[12] Wikipedia. «Parámtros orbitales.» (2023), dirección: https://es.wikipedia.org/wiki/Elementos_orbitales.

[13] M. B. M. Navabi, «Mathematical modeling and simulation of the earth's magnetic field: A comparative study of the models on the spacecraft attitude control application,» *Applied Mathematical Modelling*, 2017.

[14] M. J. SIDI, «Spacecraft Dynamics and Control,» *Cambridge University Press*, págs. 26-28, 1997.

[15] NOAA. «International Geomagnetic Reference Field (IGRF).» (2023), dirección: https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field.

[16] C. B. e. a. P. Alken E. Thébault, «International Geomagnetic Reference Field: the thirteenth generation.,» *Earth Planets Space 73, 49*, 2021.

[17] J. Davis, «Mathematical Modeling of Earth's Magnetic Field,» *Virginia Tech, Blacksburg, VA 24061*, 2004.

[18] J. Zhu, «Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates,» *IEEE Transactions on Aerospace and Electronic Systems, vol. 30, no. 3, pp. 957-961*, jul. de 1994.

[19] L. R. A, «The main field,» *Geomagnetism, Volume 1, Jacobs, J. A., Academic Press*, 1987.

[20] L. Bystricky. «EGM96.» (2022), dirección: https://people.sc.fsu.edu/~lb13f/projects/space_environment/egm96.php.

[21] D. A. Vallado, «Fundamentals of Astrodynamics and Applications,» *Microcosm Press and Springer*, 2013.

[22] L. Bystricky. «How to Get The Spherical Harmonic Coefficients and Other Solution Products.» (2022), dirección: https://cddis.nasa.gov/926/egm96/getit.html.

[23] Wikipedia. «Earth Gravitational Model.» (2023), dirección: https://en.wikipedia.org/wiki/Earth_Gravitational_Model.

[24] A. Tewari, «Atmospheric and Space Flight Dynamics (Modeling and Simulation with MATLAB and Simulink),» *Department of Aerospace Engineering Indian Institute of Technology*, págs. 117-191, 2007.

[25] A. M. B. Richard L. Burden Douglas J. Faires, «Análisis numérico,» *Cengage Learning*, págs. 193-366, 2016.

[26] CelesTrak. «Astrodynamics Software.» (2023), dirección: hhttps://celestrak.org/software/vallado-sw.php.

[27] O. M. E. Gill, «Satellite Orbits Models, Methods, and Applications,» *Springer*, 2001.

[28] CelesTrak. «NORAD GP Element Sets Current Data.» (2023), dirección: https://celestrak.org/NORAD/elements/.

[29] Hisdesat. «Satélite PAZ.» (2023), dirección: https://www.hisdesat.es/paz/.

[30] P. Mineault, «Good research code,» 2022.